UDC 004.7

COMPARATIVE EFFICIENCY OF DATA SHARDING STRATEGIES IN DISTRIBUTED LEDGER SYSTEMS

Goryunova E.T.
specialist degree, Novosibirsk state university (Novosibirsk, Russia)

Krestov S.A.
specialist degree, Novosibirsk state university (Novosibirsk, Russia)

СРАВНИТЕЛЬНАЯ ЭФФЕКТИВНОСТЬ СТРАТЕГИЙ ШАРДИНГА ДАННЫХ В РАСПРЕДЕЛЁННЫХ РЕЕСТРОВЫХ СИСТЕМАХ

Горюнова Е.Т.
специалист, Новосибирский государственный университет (Новосибирск, Россия)

Крестов С.А.
специалист, Новосибирский государственный университет (Новосибирск, Россия)

Abstract

This paper presents a comparative evaluation of data sharding strategies used in distributed ledger systems. The analysis explores partitioning methods, cross-shard transaction protocols, storage architectures, and security implications associated with fragmenting ledger state. Particular attention is given to performance trade-offs, consistency management, and resistance to targeted attacks in partitioned networks. The findings offer practical insight into how different sharding approaches influence system scalability, responsiveness, and reliability. The study serves as a foundation for future design choices in high-performance ledger infrastructures.

Keywords: data sharding, distributed ledger, partitioned systems, cross-shard transactions, ledger architecture, scalability, performance, blockchain security.

Аннотация

В работе проведён сравнительный анализ стратегий шардинга данных в распределённых реестровых системах. Рассматриваются методы разделения состояния, протоколы обработки межшардовых транзакций, архитектурные решения хранения и аспекты безопасности, возникающие при фрагментации реестра. Отдельное внимание уделено балансу между производительностью, управлением согласованностью и устойчивостью к целевым атакам в условиях раздельных подсетей. Представленные результаты дают практическое понимание влияния различных подходов к шардингу на масштабируемость, отклик и надёжность систем. Исследование формирует основу для выбора архитектурных решений в высоконагруженных реестровых платформах.

Ключевые слова: шардинг данных, распределённый реестр, фрагментированные системы, межшардовые транзакции, архитектура реестра, масштабируемость, производительность, безопасность блокчейна.

Introduction

The rapid expansion of distributed ledger technologies (DLTs) in recent years has been driven by the increasing demand for secure, transparent, and decentralized data management. However, as

transaction volumes grow and participation scales globally, the limitations of monolithic ledger architectures become more pronounced. Bottlenecks in throughput, rising latency, and inefficiencies in state replication challenge the practical deployment of DLT-based platforms in real-world, high-frequency environments. To address these constraints, architectural paradigms based on data partitioning have emerged as a viable approach to enhance performance without sacrificing decentralization.

Data sharding-an approach rooted in distributed database design-has gained attention as a scalable solution for distributing ledger state across multiple parallel processing units or network nodes. Sharding allows segments of the ledger to be processed independently, reducing the computational and communication burden on individual participants. Various strategies have been proposed, differing in how they allocate data, route transactions, and handle cross-shard communication. These differences directly affect throughput, fault tolerance, and consistency models, making it essential to evaluate sharding methods through both theoretical analysis and empirical validation.

This study provides a structured examination of the efficiency of multiple data sharding strategies within the context of DLT platforms. The investigation encompasses static and dynamic partitioning schemes, load-balancing techniques, and transaction routing models. Special focus is placed on how these strategies perform under heterogeneous conditions, including variable network topologies and workload distributions. By comparing the practical implications of different approaches, the paper contributes to the development of performance-aware design principles for scalable and reliable distributed ledger infrastructures.

Main part

Structural classification of sharding techniques in distributed ledgers

Sharding strategies in distributed ledger systems are developed to address the problem of limited scalability by distributing storage and processing responsibilities among multiple nodes or sub-networks. These strategies differ significantly in how data is partitioned and accessed, how inter-shard communication is managed, and what consistency guarantees can be provided across partitions. In practice, the design of a sharding model must balance simplicity of implementation, efficiency of cross-shard operations, and the ability to adapt to varying workloads or deployment conditions.

To provide a comparative overview, several representative approaches to data sharding in DLTs have been analyzed and summarized [1]. These include static key-based partitioning, dynamic schemes that adjust based on observed activity, region-aware placement strategies, hash-based random distribution, and explicit routing mechanisms for inter-shard coordination. The following table 1 presents a structural comparison of these methods, focusing on five critical characteristics: data distribution logic, scalability potential, inter-shard communication overhead, and the complexity of consistency enforcement.

Table 1

Comparison of data sharding strategies in distributed ledger technologies

Sharding strategy	**Data distribution method**	**Scalability**	**Cross-shard communication overhead**	**Consistency complexity**
Static key-range partitioning	Fixed value ranges assigned to shards	Moderate; requires pre-analysis	Low	Simple within fixed ranges
Dynamic workload-aware sharding	Partitions adjusted based on access patterns	High; adapts to usage load	Moderate	Complex due to dynamic changes
Geographic region-based sharding	Allocation by physical or network location	Contextual; depends on topology	Low to moderate	Moderate; regionally consistent

Sharding strategy	Data distribution method	Scalability	Cross-shard communication overhead	Consistency complexity
Random hash-based partitioning	Keys mapped using hash functions	High; evenly balanced in theory	High	Requires global coordination
Cross-shard transaction routing	Routing layer manages inter-shard transfers	Variable; limited by routing overhead	High	High; requires transaction-level tracking

The comparative analysis shows that while static and geographically-aware sharding offer lower communication overhead, their adaptability to dynamic load is limited. In contrast, dynamic and hash-based approaches provide improved scalability at the cost of increased coordination complexity. Cross-shard routing introduces further overhead, particularly in systems where transactional atomicity must be preserved. Selecting a sharding strategy thus requires a careful trade-off between performance, network structure, and operational guarantees [2].

Beyond structural classification, the practical implications of each sharding strategy vary depending on deployment context and system objectives. For example, static key-range partitioning may suit systems with predictable access patterns, such as supply chain tracking or digital identity registries, but lacks flexibility when transaction distribution shifts over time. Conversely, dynamic workload-aware sharding introduces adaptability but demands real-time monitoring, rebalancing logic, and robust metadata tracking, increasing the overall operational overhead.

Geographic or network-based sharding introduces physical locality into data placement, which can significantly improve performance in latency-sensitive environments. However, this advantage may diminish in cloud-based or virtualized networks where physical proximity does not guarantee consistent communication quality [3]. Randomized hash-based strategies are often appealing for their simplicity and load balancing properties but struggle with maintaining atomicity and consistency during inter-shard transactions, especially in public blockchain environments.

As illustrated in figure 1, inter-shard routing mechanisms-while introduced to support operations across partitioned ledger spaces-are accompanied by significant architectural challenges. These include ambiguity in transaction ordering, difficulties in rollback handling during failure scenarios, and elevated risks of double-spending or inconsistencies in partial states. Therefore, the selection and implementation of such strategies require careful evaluation not only of their theoretical soundness but also of their operational resilience, edge-case handling, and integration with the broader governance and consensus structures of the distributed ledger environment.

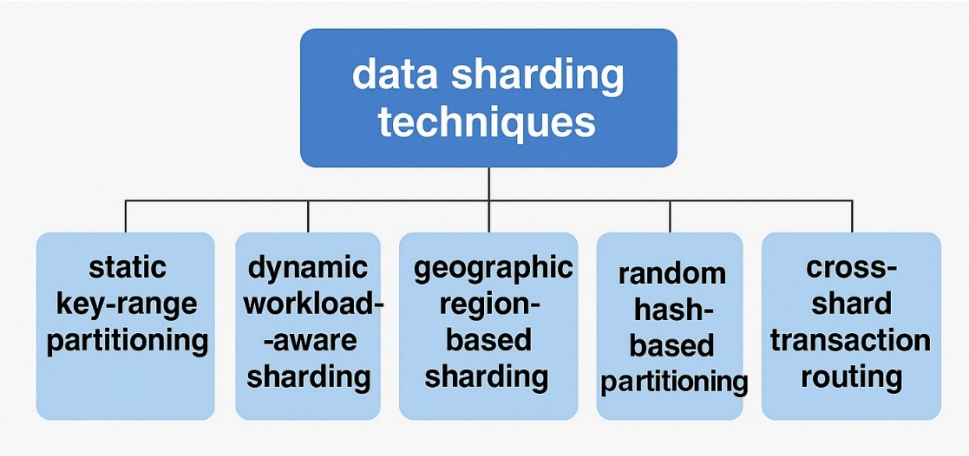

Figure 1. Structural classification of data sharding techniques in distributed ledgers

The figure provides a visual overview of the main data sharding strategies examined in this section. It highlights the logical distinctions and structural placement of each method, offering a clear reference for comparative analysis. This schematic reinforces the classification framework used to analyze performance trade-offs across varying ledger architectures.

Throughput and latency considerations in partitioned ledger systems

Performance metrics such as throughput and latency serve as primary indicators of the operational efficiency of distributed ledger systems employing sharding [4]. Throughput is generally defined as the number of transactions successfully processed per unit time, while latency refers to the delay between transaction submission and its final confirmation. In sharded environments, these metrics are influenced by a variety of factors, including the shard assignment algorithm, inter-shard message propagation delay, and the complexity of consensus synchronization across partitions.

In systems with static sharding, throughput tends to scale linearly with the number of shards, assuming an even distribution of workload and minimal inter-shard interaction. However, real-world usage often introduces workload imbalance and state contention, resulting in bottlenecks that offset the expected gains [5]. Dynamic sharding mechanisms attempt to address this by reallocating data or workload based on observed metrics, yet such adaptivity introduces overhead that may reduce net throughput, particularly in networks with frequent reconfiguration cycles or unstable connectivity.

Latency is particularly sensitive to the coordination model. Systems that require strong consistency guarantees-especially during cross-shard transactions-must perform multiple rounds of verification and commit procedures, increasing the time to finality. To mitigate this, some architectures adopt relaxed consistency models or delayed finality, sacrificing determinism for performance. Ultimately, the selection of a sharding strategy must be aligned with application-level tolerances: real-time systems demand minimal latency, while archival or batch-processing applications may prioritize throughput and state integrity.

Cross-shard transaction processing and consistency management

One of the defining challenges in sharded distributed ledger architectures is the processing of transactions that span multiple shards. Unlike single-shard operations, cross-shard transactions require coordinated execution across independent partitions, each maintaining its own subset of the global state [6]. This coordination must ensure atomicity, consistency, and isolation despite the absence of centralized control. Achieving these properties necessitates the design of robust communication protocols, transaction staging mechanisms, and conflict resolution strategies.

Several models have been proposed to manage cross-shard operations, including two-phase commit (2PC), optimistic concurrency control, and asynchronous messaging with eventual reconciliation. The 2PC approach ensures strong consistency by having all involved shards prepare and confirm the transaction before committing it globally. While effective, this method introduces significant latency and is vulnerable to deadlock in the presence of node failure. Optimistic concurrency models, by contrast, allow tentative execution followed by validation, reducing latency but risking rollback when conflicts arise. Asynchronous designs prioritize throughput and scalability by deferring coordination, accepting the risk of temporary inconsistencies.

In practice, the choice of model depends on the application's tolerance to temporary divergence and its need for fast finality. Financial ledgers, for instance, often require strict consistency and cannot afford conflicting transaction states, necessitating stronger coordination. In contrast, supply chain traceability systems may tolerate short-term inconsistencies in favor of performance. Additional mechanisms such as versioned state tracking, deterministic ordering, and cryptographic proofs (e.g., Merkle proofs for inter-shard state inclusion) are integrated to support secure reconciliation and verification across partitions.

Maintaining consistency across shards is further complicated by network dynamics, such as variable message delays and asynchronous node participation. To address this, some systems implement consistency layers that monitor state divergence and trigger reconciliation cycles or fallback consensus procedures. Others integrate coordination metadata directly into the transaction payloads, enabling context-aware validation at the shard level. These architectural choices influence not only correctness but also the resource efficiency and resilience of the overall system.

Ultimately, the effective processing of cross-shard transactions is a trade-off between protocol complexity, consistency guarantees, and system responsiveness. As applications demand both scalability and correctness, the future of sharded ledgers will likely depend on hybrid models that

dynamically adjust the degree of coordination based on transaction type, system load, and risk profile [7].

Models of cross-shard transaction processing in distributed ledger environments

The ability of a distributed ledger system to reliably process transactions that span multiple shards is a crucial factor in its practical viability. As data and users are divided across independent partitions, transactions affecting multiple shards must be executed in a coordinated and consistent manner. Without proper synchronization, inconsistencies in ledger state may emerge, leading to conflicting records, invalid balances, or security vulnerabilities. Therefore, designing robust models for cross-shard transaction handling is essential for preserving system integrity in sharded architectures.

Multiple approaches to cross-shard transaction processing have been proposed, each offering a different balance between consistency, performance, and failure tolerance. Traditional methods such as two-phase commit ensure atomicity but are susceptible to delays and coordination deadlocks. More modern techniques, including optimistic concurrency control and asynchronous reconciliation, aim to reduce overhead but introduce risks of temporary divergence. In addition, cryptographic methods like Merkle proof-based validation provide lightweight, secure ways to verify state inclusion across shards. These strategies differ not only in their implementation complexity but also in their resilience to failures and impact on transaction finality time.

The following table 2 summarizes and compares five common models of cross-shard transaction processing, focusing on consistency level, latency implications, and fault sensitivity.

Table 2

Comparison of cross-shard transaction processing models

Transaction model	Consistency guarantee	Latency impact	Failure sensitivity
Two-phase commit (2PC)	Strong (atomic and durable)	High	High (vulnerable to node stalls)
Optimistic concurrency control	Eventual (with possible rollback)	Low to moderate	Medium (depends on conflict rate)
Asynchronous reconciliation	Weak (requires post-verification)	Low	Low (tolerates delays)
Versioned state tracking	Moderate (conflict-aware updates)	Moderate	Medium (requires revalidation)
Merkle proof-based validation	High (cryptographic verification)	Moderate	Low (state inclusion verifiable)

This comparative overview highlights that there is no one-size-fits-all solution to cross-shard transaction processing. Systems prioritizing strong consistency and transactional determinism may opt for protocols like 2PC, despite their higher latency. Applications tolerant of temporary divergence can benefit from optimistic or asynchronous approaches, improving responsiveness. Cryptographic verification offers an efficient compromise, enhancing trust in inter-shard data without complex coordination. Ultimately, the choice of model should be informed by the operational profile of the ledger system, the nature of its workload, and the criticality of real-time consistency [8].

Storage optimization techniques in partitioned ledger environments

Efficient data storage is a fundamental requirement for scalable sharded ledger systems. As the number of partitions grows and historical data accumulates, the choice of storage strategies directly influences system responsiveness, fault tolerance, and operational cost. Each shard must manage not only transactional state but also indices, metadata, and recovery checkpoints. Consequently, designers face trade-offs between redundancy, retrieval speed, and storage economy that must be resolved based on workload type and access frequency.

Several architectural models have emerged to address these concerns. Some partitions employ full data replication to ensure high availability and quick recovery, especially in safety-critical systems. Others apply erasure coding techniques to balance fault tolerance with reduced storage footprint by splitting data into fragments with mathematical parity. In certain implementations, only

indexed summaries or state deltas are retained at the shard level, with full data archived externally or offloaded to cold storage layers. These approaches differ in their retrieval complexity, failure recovery mechanisms, and consistency implications.

Figure 2 presents a schematic overview of common storage strategies employed across independent partitions in sharded ledger systems.

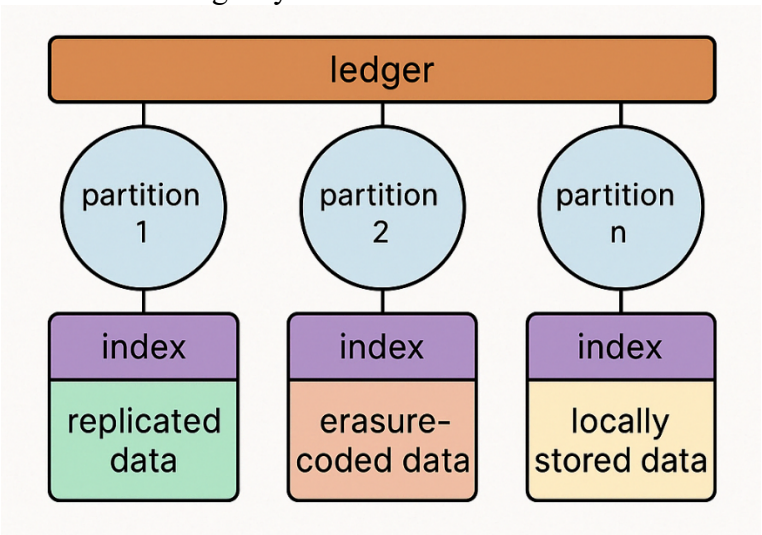

Figure 2. Storage strategies for sharded ledger systems

The figure illustrates how distinct shards may adopt different storage paradigms: full replication for resilience, erasure coding for storage efficiency, and local indexing for minimal operational load. The visual differentiation reinforces the idea that no single approach is universally optimal; rather, hybrid storage layering across partitions may offer the best trade-off between durability, space optimization, and retrieval latency in large-scale distributed ledger infrastructures.

Security implications of sharding in distributed ledgers

While sharding significantly enhances the scalability of distributed ledger systems, it also introduces unique security considerations that must be carefully addressed during protocol design and deployment. The very act of partitioning data and computation across independent subnets alters the traditional trust and threat models inherent to monolithic architectures. Each shard becomes a potential point of vulnerability, where local compromise may impact the integrity or availability of a segment of the global ledger.

One of the most critical security concerns is the risk of shard takeover attacks, in which an adversary gains control of a majority of the validating nodes within a single shard. Unlike global consensus mechanisms that rely on distributed quorum, individual shards often operate under reduced participant diversity, making them more susceptible to targeted collusion or sybil attacks. To mitigate this, some systems implement periodic re-shuffling of shard membership, use randomized assignment of nodes, or require cross-shard notarization before a transaction is finalized. However, these techniques introduce operational overhead and must be balanced against performance and complexity.

Another dimension of risk involves cross-shard transaction manipulation. As transactions span multiple partitions, the opportunity arises for adversaries to intercept, delay, or reorder messages to create inconsistent state transitions or double-spending scenarios. Ensuring secure coordination across shards requires cryptographic proofs, verifiable delay functions, and secure messaging protocols that are resistant to tampering or timing attacks [9]. Moreover, safeguarding transaction integrity depends on strict atomicity and rollback mechanisms, which must operate effectively even under partial network failure or adversarial interference.

Finally, data privacy and leakage become nuanced issues in sharded environments. While segmentation may isolate data access to specific shards, it can also reveal patterns in data distribution, transaction frequency, or network topology that adversaries could exploit for inference attacks. Zero-knowledge proofs and homomorphic encryption are being explored as privacy-preserving enhancements, though their integration with sharded architectures remains an open research

challenge. Additionally, careful attention must be paid to metadata exposure in inter-shard routing and consensus logs, where auxiliary information may inadvertently disclose sensitive context.

In summary, the decentralization benefits provided by sharding must be weighed against the complexity of maintaining consistent security guarantees across a fragmented system. A secure sharded ledger must not only defend each partition independently but also ensure that the collective behavior of the system preserves confidentiality, integrity, and availability at scale. The design of such architectures demands a multidisciplinary approach, integrating distributed systems theory, cryptography, and real-world operational insight.

Conclusion

Data sharding has emerged as a foundational technique for enhancing the scalability and responsiveness of distributed ledger systems. By segmenting state and workload across independent partitions, sharding enables parallelism, reduces transaction congestion, and aligns computational responsibilities with network topology. However, this architectural evolution introduces new challenges that span coordination, consistency, storage, and security domains.

This study has provided a comparative analysis of diverse sharding strategies, including static and dynamic partitioning, cross-shard transaction models, storage optimization schemes, and security mechanisms. Through structured examination of design trade-offs and performance characteristics, the paper highlights that the choice of sharding approach must be carefully aligned with system objectives, including fault tolerance, real-time responsiveness, and consistency requirements. No single model offers universal superiority; instead, adaptive and hybrid architectures often present the most viable path forward.

As distributed ledger technologies continue to evolve toward broader adoption in finance, supply chain, identity, and beyond, future research must focus on the refinement of sharding protocols that combine efficiency with verifiable trust guarantees. The advancement of formal verification, cryptographic coordination, and context-aware orchestration will play a critical role in shaping the next generation of scalable, secure, and interoperable ledger systems.

References

1. Quan B.L.Y., Wahab N.H.A., Al-Dhaqm A., Alshammari A., Aqarni A., Abd Razak S., Wei K.T. Recent Advances in Sharding Techniques for Scalable Blockchain Networks: A Review // IEEE Access. 2024.
2. Blazhkovskii A. Collecting metrics for continuous platform monitoring // Universum: technical sciences : electronic scientific journal. 2025. No. 3(132). P. 10-15.
3. Dhulavvagol P.M., Totad S.G. Performance enhancement of distributed system using HDFS federation and sharding // Procedia Computer Science. 2023. Vol. 218. P. 2830-2841.
4. Wu J., Yuan L., Xie T., Dai H. A sharding blockchain protocol for enhanced scalability and performance optimization through account transaction reconfiguration // Journal of King Saud University-Computer and Information Sciences. 2024. Vol. 36. No. 8. P. 102184.
5. Terletska K. Low-level memory management in scalable distributed architectures: Approaches to improving reliability and performance of digital services // International Journal of Research Publication and Reviews. 2025. Vol. 6(4). P. 5078-5081.
6. Aslam A., Postolache O., Oliveira S., Pereira J.D. Securing IoT Sensors Using Sharding-Based Blockchain Network Technology Integration: A Systematic Review // Sensors (Basel, Switzerland). 2025. Vol. 25. No. 3. P. 807.
7. Heo J.W., Ramachandran G.S., Dorri A., Jurdak R. Blockchain data storage optimisations: a comprehensive survey // ACM Computing Surveys. 2024. Vol. 56. No. 7. P. 1-27.
8. Yang H., Zhang X., Wu Z., Wang L., Chen X., Liu L. Co-sharding: a sharding scheme for large-scale internet of things application // Distributed Ledger Technologies: Research and Practice. 2024. Vol. 3. No. 1. P. 1-16.
9. Xu G., Zhou Z., Song X., Huang Y. Research on transaction allocation strategy in blockchain state sharding // Future Generation Computer Systems. 2025. P. 107756.