Performance Optimization of Decentralized Databases: Challenges, Existing Solutions and Prospects

Jiamin Wang

School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing, China

Abstract. With the increasing demand for data sovereignty, privacy protection, and tamper-proof systems, decentralized databases have gained growing attention in recent years. This review systematically analyzes the tech stacks of representative decentralized database solutions, including BigchainDB, GUN, OrbitDB, Bluzelle, Fluree, TiesDB, Fabric, and CovenantSQL, and provides an in-depth analysis of the performance bottlenecks and existing optimizations on the consensus, network, storage, data engine, and application interface layer of the decentralized databases. This review specifies the efficacy and limitations of multiple optimizations on different layers and different points, including the network protocol, the consensus mechanisms, decentralized data storing technologies, a high-performance data engine, and an improved interface, according to the relevant studies and existing solutions. As a result, despite significant progress in performance optimization, the decentralized databases still face challenges in cross-chain tuning, standardization of benchmarking, balancing between security and performance, and the gap between theory and deployment. To promote the application of decentralized databases, lightweight consensus, smart protocol designs, and standardized performance measuring methods is in demand in the future.

1 Introduction

In an increasingly digitized society, databases are essential infrastructures for storing and managing information in modern digital applications. A centralized database, the most used database nowadays, indicates that the service provider stores the data. As information technology keeps evolving, along with blockchain development, a new type of database technology, decentralized databases, is emerging. Instead of relying on a single authority, in a decentralized database, the data is separated and maintained on multiple nodes, maintained by system participants. As a fundamental component of Web3 and distributed systems, this database has shown great potential in various domains, including cryptocurrency, decentralized finance, and decentralized identity management.

Despite many merits, decentralized databases still face performance problems, including lower throughput, higher latency, higher redundancy rate, lower query efficiency, and

miul@bupt.edu.cn

excessive overhead under specific consensus mechanisms, such as the "proof of work" mechanism [1]. Improving the general performance of database technology is crucial to making decentralized database technology a realistic solution for real-world scenarios, including finance, supply chain, and IoT, as well as reducing energy consumption and improving sustainability.

In recent years, researchers have actively explored diverse approaches to improve the performance of decentralized databases across multiple dimensions. Isaac Zhang et al. provided a solution to improve the querying and updating throughput and storage efficiency using the Merkle tree, significantly improving the scalability and maintainability [2]. To improve the overall performance of the decentralized autonomous database system, Prakash Aryan et al. used the Rust programming language to build an experimental framework. They achieved considerably higher consistency and throughput even with a certain proportion of ill-intentioned nodes, demonstrating the framework's advantages in security and performance [3]. Concerning reducing latency, Qiyu Zhuang et al. proposed a latency-aware geo-distributed transaction processing strategy that significantly improved distributed transaction throughput [4]. Similarly, a decentralized 2-phased commit protocol proposed by Zihao Zhang et al. significantly reduced the latency of transaction commitment [5].

In addition to the listed theories, multiple real-world solutions were given simultaneously. On the Internet of Things field, Thandile Nododile et al. introduced a hybrid blockchain-IPFS solution to improve the throughput and block-generating velocity [6]. On low-power devices, a blockchain-based academic credential verification system is developed by Gabriel Fernández-Blanco et al. to improve power efficiency and performance, including throughput and resource demands [7]. Their app was based on the Ethereum blockchain and IPFS distributed storage, adopted the PoA consensus mechanism, and achieved lower computational overhead. In 2024, a decentralized storage network, "FileDES," was proposed by Minghui Xu et al., which demonstrated an advantage in verification performance [8]. In healthcare, a blockchain-based record management solution was proposed by Geeta N. Brijwani et al. in 2025, which involves significant improvements in consensus efficiency and transaction latency [9].

This review aims to systematically analyze the key performance bottlenecks, examine existing optimization techniques, and identify potential directions for future improvements. The review first introduces fundamental concepts and performance metrics relevant to decentralized databases. Then, it categorizes representative systems and reviews current optimization strategies. Finally, the article highlights ongoing challenges and proposes directions for future development based on emerging trends in database technologies.

2 The Layered Structure and Performance Metrics of Decentralized Databases

2.1 A layered structure of the decentralized databases

Based on existing research on representative systems like BigchainDB and the common practice of information systems, a layered structure of the decentralized database is adopted to categorize, analyze, and compare the existing solutions [10]. The proposed layered structure is shown in Fig. 1 below.

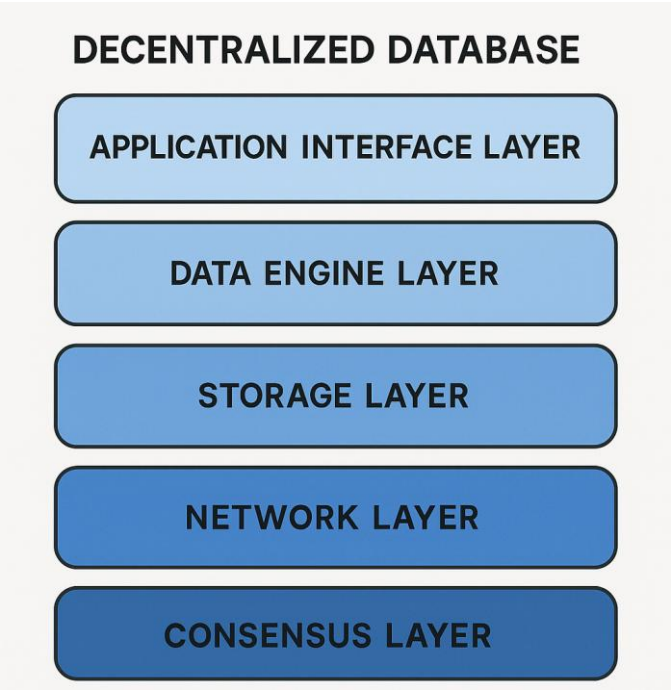

Fig. 1. Layered Structure of the Decentralized Databases (Picture credit: Original)

As shown in the graph, the consensus layer, the network layer, the storage layer, the data engine layer, and the application interface layer roughly combine to form the decentralized database system. Notably, the structure of layers is only generally described and may vary depending on the implementation of the specific database system. The listed 5 layers constitute a complete structure overview of the decentralized database, which solves the data consistency, security, and availability problems, significantly reducing the complexity of developing such a system.

2.1.1 The Consensus Layer

In a decentralized database system, the data stored partially on different nodes must be consistent to ensure a shared view of the data status. The consensus layer adopts various algorithms to solve the "Byzantine Generals Problem," a problem that compromises data consistency across distributed systems, to prevent common security threats, including data poisoning, double spending, etc. [11]. Common consensus mechanisms include "Proof of Work," "Proof of Stake," "Practical Byzantine Fault Tolerance," "Delegated Byzantine Fault Tolerance," "Proof of Previous Transactions," etc, and their derived algorithms. These mechanisms have different features, and their performances vary on multiple aspects, including energy cost, processing cost, and scalability [12].

2.1.2 The Network Layer

The network layer of the decentralized database system is responsible for establishing and maintaining communication between nodes, enabling data to spread across them. The decentralized database generally adopts an identical network structure, eliminating central servers. Other network layer functions include fault tolerance to ensure the reliability of

data transmission under a network with nodes that keep joining and exiting. Major network protocols used include the P2P protocol and Gossip protocols, which have enhanced fault-tolerant features [13].

2.1.3 The Storage Layer

The storage layer handles the storage and data objectification with decentralization capabilities. At this layer, data is sliced and separated across multiple nodes so that a single failure will not cause data loss. At this layer, the distributed file system can be implemented. One popular distributed file system is IPFS, which uses DHT, a distributed hash table [14].

2.1.4 The Data Engine Level

Like conventional databases, the data engine provides querying and optimization functions, including creating indexes, query optimization, and data aggregation analysis. The data engine is required to be capable of representing complex data structures, including key values, documents, and graphs, and transaction features are desirable. To provide better performance, the data engine for the decentralized database system needs to optimize its operation over multiple nodes and cooperate with the consensus layer to ensure data integrity and consistency. Therefore, at this level, the solution provider must choose their data model and develop a suitable engine for the system.

2.1.5 The Application Interface (API) Level

The application interface level provides encapsulated interfaces to operate the database. This layer allows the user and developer to use a simple API without concern about the implementation details. The key functions of the API include identity authorization, data querying and modification, and data encryption. At this level, programming languages ensure the convenience of interacting with the system.

2.2 Performance Metrics

Evaluating the performance of the decentralized database system is crucial to ensuring its reliability and efficiency in a decentralized environment. In this section, this review researches and determines key metrics for evaluating the performance of the decentralized database system with the help of existing evaluation research [14]. The factors include throughput, latency, availability, consistency, and scalability. The definition of the factors is given below.

The throughout is the data quantity processed per unit of time. A higher throughput indicates higher scalability and the competence to serve many users, which is critical for a decentralized database. The latency refers the time consumption from data submission to confirmation. Low latency is crucial for real-time applications like Defi. The availability indicates the proportion of time that is available in a period. A higher availability makes the system more reliable and can improve the user experience. The consistency evaluates the status where all the nodes share the same data status is a basic requirement for decentralized applications. The scalability represents the ability to increase the scale of the system without severe performance loss. Good scalability enables the system to adapt to the growing number of users and data.

This review provides a qualitative analysis of the selected systems to understand the current progress. If available, precise data will be noted.

2.3 Performance Comparison Between Decentralized Databases and Traditional Databases

Given their special mechanism and characteristics, decentralized databases face unique performance challenges. This section discusses performance setbacks and trade-offs on the chosen performance metrics, including throughput, latency, availability, consistency, and scalability.

The Table 1 below provides an overview of the performance comparison between the traditional and decentralized databases on the chosen metrics.

Table 1 Comparison of performance metrics between traditionalized databases and decentralized databases

Metrics	Traditional Databases	Decentralized Databases
Throughput	Higher, supportive of concurrent data processing	Relatively lower, constricted by consensus mechanisms and network latency
Latency	Usually, Lower	Higher
Availability	High, but it has a risk of single-point failure	Higher with fault tolerance
Consistency	Strong consistency with real-time synchronization	Eventual consistency with the possibility of inconsistency
Scalability	Limited and complex to expand	Easier to expand

2.3.1 Throughput and Latency

A traditional database system adopts a centralized architecture that relies on a single node to handle transactions. It is more cost-efficient to achieve high throughput and low latency. For instance, traditional relational databases provide rapid response by adopting efficient query optimization and indices.

Decentralized databases, on the other hand, face heavy latency and throughput limitations due to the decentralized consensus mechanisms they rely on. Consensus must be established between multiple nodes before any operation on the database is enacted [15]. For example, in some blockchain systems, the confirmation time of transactions can reach up to several minutes, significantly higher than traditional databases' millisecond response time.

2.3.2 Availability and Consistency

Traditional databases rely on a centralized serving structure, which introduces the risk of a single failure. A centralized database uses strategies to improve availability, including master-slave replication and failover. Even though higher availability is achieved through decentralization and consensus mechanisms, which avoid single point failure, the decentralized databases make a trade-off, which sacrifices consistency. For instance, allowing the eventual consistency model allows for inconsistency across the system in a short time after modification. This consistency model may lead to read latency and dirty data, affecting user experience [16].

2.3.3 Scalability

Traditional database systems face challenges in scalability, even with load balancing and segmentation, which increase complexity.

A decentralized database is scalable by nature. Increasing the number of nodes can easily improve the system's capacity. However, this can decrease system performance because communication time and time to reach consensus will increase correspondingly. Therefore, the balance between capacity and other performance metrics should be considered thoroughly based on the demands before constructing the system.

3 Popular Decentralized Databases: Comparison of Tech Stack

3.1 Comparison table of representative decentralized databases

The following Table 2 compares the technologies used in each layer by several popular decentralized databases, including BigchainDB, GUN, OrbitDB, Bluzelle, Fluree, TiesDB, Fabric, and CovenantSQL.

Table 2 Comparison of tech stack of representative decentralized databases

Database System	Consensus Layer	Network Layer	Storage Layer	Data Engine Layer	Application Interface Layer
BigchainDB [17]	Tendermint (BFT PoS)	Tendermint P2P Network	MongoDB Distributed NoSQL Database	Blockchain-based Transactional JSON Asset Model	HTTP API, SDK (Python/JS)
GUN [18]	CRDT-based Conflict Resolution, GUN HAM Algorithm	Decentralized P2P, Browser and Runtime Nodes	Local Storage: Browser IndexedDB, Persistent Node Storage	Graph Database, Node-Edge Model	JavaScript Realtime API (Browser/ Node.js)
OrbitDB [18]	Conflict-free Replicated Data Type, CRDT	IPFS P2P Network with PubSub	IPFS Content-addressed Storage, Decentralized File System	Multiple CRDT Storages: Key-Value, Log, Document	JavaScript API With IFPS Node
Bluzelle [19]	Tendermint BFT Proof of Stake	Cosmos/Tendermint P2P Network	Swarm Distributed Storage, DHT Network	NoSQL Key-Value Store, Dynamic Sharding	REST/JS (Cosmos SDK)

				and Cache	
Fluree [18]	Hybrid Consensus: PoS+BFT, Proprietary Protocol	Enterprise Blockchain Network, Private/Permissioned Chain	Ledger Log and Graph Storage	Semantic Graph Database: RDF, Temporal Property Graph	GraphQL, RESTful API
TiesDB [18]	Proof-of-Stake	Decentralized Self-organized P2P Network	Distributed Relational Storage, SQL-like Database	SQL-like Relational Model, Complex Query Supported	SQL-like Interface
Fabric [18]	Pluggable Consensus: PBFT, Raft, etc.	Enterprise Permissioned Blockchain, Gossip Channel	LevelDB/ CouchDB for State, Blockchain Ledger	Chaincode, Smart Contract, and KV Storage	Go/Java/Node SDK, REST API
CovenantSQL [20]	DPoS Xenomintor BFT-Raft Kayak	Decentralized Custom P2P Network with DHT	SQLite Single-instance Engine, Database Engine Node	SQL-92 Relational Model, Transaction Handling, Efficient Indexing	SQL Interface

3.2 Layered solution comparison

3.2.1 The Consensus Layer

Among the database systems selected, BigchainDB and Bluzelle are based on Tendermint BFT PoS. Fabric supports both PBFT and Raft. Fluree designed a hybrid consensus mechanism based on PoS and BFT. GUN and OrbitDB, however, adopt a CRDT eventual consistency mechanism instead of traditional blockchain consensus. TiesDB and CovenantSQL adopt an improved Pos/BFT consensus to ensure both performance and security.

3.2.2 The Network Layer

The selected database systems are all based on P2P network architecture but have varied forms. OrbitDB relies on the PubSub mechanism of IPFS. GUN nodes can interconnect between browsers and servers to form a decentralized, identical network. Nodes of Fabric communicate through the Gossip channel based on a permission chain architecture. CovenantSQL constructed a global subchain network itself. Overall, the P2P network is crucial for decentralized data storage and sharing. The selected projects have different characteristics in network topology, roles of nodes, and communication protocols.

3.2.3 The Storage Layer

Regarding storage implementation, BigchainDB stores blockchain ledger data on a traditional database [17], MongoDB [18]. Fabric uses LevelDB and CouchDB to store on-chain status and keeps track of blogs simultaneously. GUN and OrbitDB store data locally on the nodes like browser storage or IPFS, replicating the data under content addressing. Bluzelle uses Cosmos SDK to construct a decentralized storage network and slices the data into "swarm" [19, 20]. TiesDB and CovenantSQL, as relational systems, store data in SQL-like ways on each node. To sum up, some systems use traditional databases and distributed ledgers simultaneously, while others construct their storage with the help of decentralized file systems or customized storage networks.

3.2.4 The Data Engine Layer

BigchainDB uses a “transaction and block" model, which keeps track of assets and metadata. The MongoDB engine provides its query capability. GUN uses a graph-based data model. The data is stored in the form of key-value and vertex-edge. OrbitDB provides various data structures, including event log, key-value, and document sets, and uses Merkle-CRDT to realize conflict-free merge. Fluree is built on a temporal semantic graph and supports complex queries on the blockchain. TiesDB provides SQL-like relational query capability and maintains a relational model in a decentralized background. CovenantSQL implemented the SQL-92 standard entirely. Its engine is based on SQLite and can handle complex transactions and indices. In general, the data engines of decentralized databases contradict substantially. GUN, Fluree, and OrbitDB prefer graphs and documents. TiesDB and CovenantSQL maintain the relational model. BigchainDB stands in the middle and focuses on asset transactions.

3.2.5 The Application Interface Layer

Most systems provide a familiar programming interface for developers. BigchainDB and GUN provide SDKs for multiple programming languages and support JSON/REST operations. OrbitDB and GUN allow browser access through a JavaScript API. Fabric provides multiple language SDKs and supports various language interfaces. Fluree supports GraphQL and RESTful Queries. TiesDB and CovenantSQL are compatible with the standard SQL interface, improving transferability. The different designs of APIs reflect system concerns and objects. Some are mainly for Web applications, while others are for enterprises.

3.3 Layered solution analysis

The selected solutions have different preferences for technology combinations. BigchainDB and Fabric focus on the security and reliability of blockchain transactions. Bigchain specifically improved throughput using Tendermint. Fabric allows customization to satisfy the needs of the consortium blockchains. GUN and OrbitDB focus on decentralized web applications and adopt decentralized topography and CRDT to ensure consistency. Bluzelle aims to provide a scalable storage layer to dApps under the Cosmos/Tendermint tech stack. Fluree combines semantic knowledge graphs and ledgers to support complex data governance and auditing for enterprises. TiesDB and CvenantSQL share similar purposes of providing SQL-like services.

All the solutions also have drawbacks. BigchainDB has limited privacy control. Gun and OrbitDB failed to realize strong consistency. Fabric and Fluree are more complex when deployed. Users are advised to choose different solutions based on their needs.

4 The Bottlenecks and Optimizations of the Five-Layered Decentralized Databases

This section focuses on common performance bottlenecks and existing optimization strategies and practices for the five layers in decentralized databases.

4.1 The consensus layer: scalability and efficient consensus protocols

4.1.1 The Bottlenecks and The Corresponding Optimization strategies

The decentralized databases require reaching consensus on transaction sequence and status. CFT algorithms like Raft have a bottleneck from the single main node limiting overall capacity and throughput [21]. PBFT consensus mechanism has a time complexity of O(n^2) due to the multistage broadcast, causing severe performance decrease with many nodes [22]. To improve the performance, the HotStuff Protocol significantly improves the efficiency of the BFT consensus by realizing linear complexity, responsive consensus, and low latency under optimal network conditions [23]. Another BFT consensus, FastBFT, relies on hardware assistance to improve scalability, which successfully reduced message complexity to the level of O(n) [24]. At the same time, DAG and parallel consensus achieved progress on the public blockchain. Regarding lightweight PoS consensus, the Gasper consensus for Ethereum 2.0 adopts Casper FFG and LMD-Ghost chain selection rules, improving the transaction throughput while maintaining security [25].

4.2 The network layer: latency and routing efficiency in dynamic p2p networks

4.2.1 The Bottlenecks and The Corresponding Optimization strategies

The decentralized databases usually adopt a P2P network architecture, in which nodes communicate through protocols like libp2p. On one hand, due to the wide distribution and dynamic topography, content requests are likely to go through multiple routers, which increases latency. IPFS, for instance, broadcasts a request message using the Bitswap protocol, which introduces certain latency when the cache is missed [26]. Another example is Kademlia DHT, a common routing algorithm [27] which can face performance challenges in a highly dynamic P2P network due to the frequent ups and downs of nodes. One improvement is the OP algorithm, which is capable of accelerating content-providing announcements in IPFS. According to Trautwein et al., the OP algorithm significantly reduces storage latency to seconds [28]. On the Kademlia routing, the IPFS community launched the refactoring of the public DHT, called Amino, to streamline code to improve concurrency performance and to optimize publication flows to make the content-providing broadcasts more efficient.

4.3 The storage layer: latency, redundancy, and data availability

4.3.1 The Bottlenecks and The Corresponding Optimization strategies

Compared to traditional centralized storage, decentralized storage faces challenges in terms of latency and data access stability. Firstly, data access latency is usually higher due to the distributed data across multiple nodes. IPFS requires a thorough lookup in the DHT when querying content not stored locally, which brings in significant latency [16]. In addition, decentralized storage usually requires storing more copies or slices on multiple nodes to improve data availability, which brings in high data redundancy. Filecoin requires data replication by storing miner collateral and proof of time, which introduces significant overhead at the same time. Arweave's initial writing overhead is also substantial, though it features durability. Correspondingly, focusing on a single node bandwidth bottleneck, future decentralized storage tends to slice the data and store it across multiple nodes for parallel transmission to improve overall bandwidth. For instance, the Storj network encodes the file and slices the file into constructable pieces [29]. Secondly, on data compression and de-duplication, IPFS implemented natural data deduplication through Merkle DAG with content addressing [14]. Moreover, combining decentralized storage with existing infrastructure can improve performance. For example, Filecoin and IPFS are usually used in conjunction [30]. On the protocol side, IPFS is developing a decentralized indexing service to accelerate content locating without reading through the DHT [31].

4.4 The data engine layer: concurrency control and transaction performance trade-offs

4.4.1 The Bottlenecks and The Corresponding Optimization strategies

The implementation of the data engine is concerned with concurrency control, transaction isolation, and index structure. Concurrency via locks can introduce overhead, including deadlock and contention. MVCC involves maintaining multiple data versions, increasing storage overhead. Moreover, many decentralized databases choose weak consistency models like eventual consistency in exchange for performance, which can lead to reading errors. What's more, on data models, document model databases like CouchDB can have large volumes of records. Key-value databases like LevelDB, due to their bottleneck of limited querying and transaction capabilities, this type of database usually adopt an LSM tree as an index structure, which can result in compaction overhead [32]. On lock races, modern databases tend to use finer-grained locking or lock-free concurrency control. Meanwhile, the version recycling mechanism is improved to minimize the impact of old versions on performance. For high writing conflict scenarios, researchers proposed dynamic graph scheduling or hardware timestamp-based concurrency control to reduce waiting overhead [33, 34]. Regarding the level of isolation, considering performance, databases can adjust the provided isolation level, like Read Committed and Repeatable Read, which have lower overhead. Some blockchain databases completely give up traditional isolation and turn to sequential transactions to reach consistency.

4.5 The application interface layer: response time and availability

4.5.1 The Bottlenecks and The Corresponding Optimization strategies

One main bottleneck of the performance for the API layer of the decentralized databases is the high response time of requests due to decentralization, resulting in a high response time of the API request [35]. The other obstacle to the API layer of the decentralized databases is the request fail rate and the retry rate. This layer faces challenges of maintaining availability with the possibility of offline nodes and a fragmented network environment. To avoid the request flooding the P2P network, researchers suggest that the network should redirect the request to the most probable nodes with the requested data to cut unnecessary communication [35].

5 Challenges and Open Problems

Despite various decentralized database solutions and optimization strategies, performance optimization still faces multiple open challenges, including performance bottlenecks exposed by experiments and problems lying at the design of architectures. In this section, this review discusses several representative challenges and their actual impact in real practice.

5.1 The challenge of balancing performance and security

The decentralized databases usually need to reach a balance between high performance and high security. This problem corresponds to the challenge of the famous "impossible triangle" (decentralization, security, scalability) in the blockchain field. To improve performance usually means to compromise the extent of decentralization or security. As a comparison between the centralized ledger, QLDB from Amazon, and the decentralized database, BigchainDB, indicates, despite the flexibility and openness, BigchainDB has lower performance and higher complexity [36]. Therefore, the decentralized databases are naturally disadvantaged in terms of performance, where tradeoffs between security and throughput are vital.

5.2 Coordination difficulties in cross-layer optimization

The decentralized databases are typically composed of multiple layers. Though various optimization strategies are available for each layer, the coordinated optimization across multiple layers is still considerably complicated due to the conflicts of optimization between different layers. For example, at the consensus layer, increasing the number of transactions or reducing the interval of epochs can increase system throughput. But this strategy can put huge pressure on the storage layer due to the storage layer's bottleneck. This type of inconsistency requires careful selection of parameters in multiple layers.

5.3 The lack of adequate performance measurement standards and benchmarking methods

As of now, the field of decentralized databases does not have adequate performance measuring standards like traditional databases do. Projects use self-defined figures to discuss their own performance, which poses a difficulty in fair comparison. For example, TPS, transaction per second, is used broadly. However, the definition of transaction varies

in different projects. The direct comparison is pointless since the size and complexity of the transactions can vary substantially. This phenomenon urges the design of standardized measurement methods.

The lack of representative and compatible benchmark solutions is also significant. As traditional databases have well-accepted benchmarks like TPC-C and YCSB, the decentralized databases lack such programs [37, 38]. Despite preliminary attempts to measure the performance of private blockchains, like BLOCKBENCH, strong limitations still apply [39]. Developing compatible benchmark methods is crucial for the healthy and steady development of the decentralized database industry.

5.4 She gap from theory to actual deployment

Many decentralized database projects have excellent performance in theory and simulations. But their actual performance can be significantly worse due to the complexity of a real network. For example, many scientific studies run the experiments in a local area network or on cloud simulation machines, where nodes' latency and performance consistency are ideal. But in a practical environment, the performance of such systems can be significantly exacerbated due to jitters and inconsistent latencies. This gap between experiments and real practices poses challenges to the developers, requiring them to reconsider the hypotheses made in the experiments and to test repeatedly under various settings.

6 Conclusion

This review provides a systematic analysis of the performance of the decentralized databases. As the crucial basis of Web3 and decentralized systems, decentralized databases show great potential in the fields of finance, IoT, and supply chain. However, the existing problems in throughput, latency, redundancy, and efficiency pose significant limitations on its application. This review provides an in-depth discussion of the main bottlenecks and optimization strategies of the 5 layers of the decentralized databases and analyzes the efficacy and limitations of the solutions in recent years.

Despite multiple valid optimization strategies, the gap between theory and practice remains considerable. In addition, the difficulty of cross-layer optimization and the lack of a standardized performance measuring method are critical.

In future research, the lightweight consensus and intelligent protocol design, and the effort toward standardization are crucial for the development of the industry.

References

1. Nakamoto, S.: 'Bitcoin: A Peer-to-Peer Electronic Cash System'. SSRN, 2008
2. Zhang, I., Zarick, R., Wong, D., et al.: 'QMDB: Quick Merkle Database'. arXiv preprint arXiv:2501.05262, 2025
3. Aryan, P., Khatri, R., Vijayakumar, B.: 'An Experimental Framework for Implementing Decentralized Autonomous Database Systems in Rust'. Proc. 2024 Int. Conf. Computational Intelligence and Network Systems (CINS), 2024, pp. 1–8
4. Zhuang, Q., Shi, X., Liu, S., et al.: 'GeoTP: Latency-aware Geo-Distributed Transaction Processing in Database Middlewares (Extended Version)'. arXiv preprint arXiv:2412.01213, 2024

5. Zhang, Z., Hu, H., Zhou, X., et al.: 'Fast Commitment for Geo-Distributed Transactions via Decentralized Co-coordinators', Proc. VLDB Endow., 2024, 17, (10), pp. 2555–2567
6. Nododile, T., Nyirenda, C.: 'A Hybrid Blockchain-IPFS Solution for Secure and Scalable Data Collection and Storage for Smart Water Meters'. arXiv preprint arXiv:2502.03427, 2025
7. Fernández-Blanco, G., Froiz-Míguez, I., Fraga-Lamas, P., et al.: 'Design, Implementation and Practical Energy-Efficiency Evaluation of a Blockchain Based Academic Credential Verification System for Low-Power Nodes'. arXiv preprint arXiv:2410.20605, 2024
8. Xu, M., Zhang, J., Guo, H., et al.: 'FileDES: A Secure, Scalable and Succinct Decentralized Encrypted Storage Network'. Cryptol. ePrint Arch., Rep. 2024/182, 2024
9. Brijwani, G.N., Ajmire, P.E., Junaid, M.A.M., et al.: 'Revolutionizing Healthcare Record Management: Secure Documentation Storage and Access through Advanced Blockchain Solutions'. arXiv preprint arXiv:2503.00742, 2025
10. Wang, Y., Hsieh, C.-H., Li, C.: 'Research and Analysis on the Distributed Database of Blockchain and Non-Blockchain'. Proc. 2020 IEEE 5th Int. Conf. Cloud Comput. Big Data Anal. (ICCCBDA), Chengdu, China, 2020, pp. 307–313
11. Fedotova, N., Veltri, L.: 'Byzantine Generals Problem in the Light of P2P Computing'. Proc. 2006 Third Annual Int. Conf. Mobile and Ubiquitous Systems: Networking & Services, San Jose, CA, USA, 2006, pp. 1–5
12. Sharma, S., Sharma, O., Arora, J.: 'Consensus Mechanisms Analysis: A Remedy for the Byzantine Generals Problem'. Proc. 2023 3rd Int. Conf. Technological Advancements in Computational Sciences (ICTACS), Tashkent, Uzbekistan, 2023, pp. 674–678
13. Jelasity, M.: 'Gossip', in Serugendo, G.D.M., Gleizes, M.-P., Karageorgos, A. (Eds.): 'Self-organising Software' (Springer Berlin Heidelberg, 2011), pp. 139–162
14. Trautwein, D., Raman, A., Tyson, G., et al.: 'Design and Evaluation of IPFS: A Storage Layer for the Decentralized Web'. Proc. ACM SIGCOMM 2022 Conf., 2022, pp. 739–752
15. Islam, S., Apu, K.U.: 'Decentralized vs. Centralized Database Solutions in Blockchain: Advantages, Challenges, and Use Cases'. Global Mainstream J. Innov. Eng. Emerg. Technol., 2024, 3, (04), pp. 58–68
16. Bailis, P., Ghodsi, A.: 'Eventual Consistency Today: Limitations, Extensions, and Beyond'. Queue, 2013, 11, (3), pp. 20–32
17. BigchainDB: 'BigchainDB 2.0 The Blockchain Database'. Whitepaper, 2019
18. Chauhan, A.: 'A review on various aspects of MongoDB databases', Int. J. Eng. Res. Technol. (IJERT), 2019, 8 (5), pp. 90–92
19. Bluzelle: 'Bluzelle Technical Paper'. https://bluzelle.com/
20. Petriv, P., Opirskyy, I., Mazur, N.: 'Modern technologies of decentralized databases, authentication, and authorization methods', Cybersecurity Providing in Information and Telecommunication Systems II, 2024, 3826, pp. 60–71
21. Tang, W., Sheng, P., Roy, P., et al.: 'Raft-Forensics: High Performance CFT Consensus with Accountability for Byzantine Faults'. arXiv preprint arXiv:2305.09123, 2023
22. Alqahtani, S., Demirbas, M.: 'Bottlenecks in Blockchain Consensus Protocols'. arXiv preprint arXiv:2103.04234, 2021

23. Yin, M., Malkhi, D., Reiter, M.K., et al.: 'HotStuff: BFT Consensus in the Lens of Blockchain'. arXiv preprint arXiv:1803.05069, 2019
24. Liu, J., Li, W., Karame, G.O., Asokan, N.: 'Scalable Byzantine Consensus via Hardware-Assisted Secret Sharing'. IEEE Trans. Comput., 2019, 68, (1), pp. 139–151
25. Laneve, C., Solmonte, S., Veschetti, A.: 'A Stochastic Analysis of the Gasper Protocol'. Proc. 2024 IEEE Int. Conf. Pervasive Comput. Commun. Workshops, Biarritz, France, 2024, pp. 518–523
26. De la Rocha, A., Dias, D., Psaras, Y.: 'Accelerating Content Routing with Bitswap: A Multi-path File Transfer Protocol in IPFS and Filecoin', 2021
27. Maymounkov, P., Mazières, D.: 'Kademlia: A Peer-to-Peer Information System Based on the XOR Metric'. Proc. First Int. Workshop Peer-to-Peer Systems (IPTPS '01), 2002, pp. 53–65
28. Bistas, F., Xylomenos, G.: 'Evaluating IPFS Optimistic Provide in the Wild'. Proc. 2024 IEEE Conf. Comput. Commun. Workshops (CSCN), 2024, pp. 160–165
29. Gomes, B., Eisa, S., Matos, D.R., Pardal, M.L.: 'Decentralized Storage and Self-Sovereign Identity for Document-Based Claims'. arXiv preprint arXiv:2411.16987, 2024
30. Doan, T., Bajpai, V., Psaras, Y., Ott, J.: 'Towards Decentralised Cloud Storage with IPFS: Opportunities, Challenges, and Future Directions'. arXiv preprint arXiv:2202.06315, 2022
31. Buri, I., van Hoboken, J.: 'The Digital Services Act (DSA) Proposal: A Critical Overview'. Inst. Inf. Law (IViR), Univ. Amsterdam, Oct. 2021
32. Mishra, S.: 'A Survey of LSM-Tree Based Indexes, Data Systems and KV-Stores'. Proc. 2024 IEEE Int. Students' Conf. Electr. Electron. Comput. Sci. (SCEECS), 2024, pp. 1–6
33. Rusinkiewicz, M.E., Leiss, E.L.: 'Evaluation of Timestamp-Based Concurrency Control Mechanisms Incorporating Livelock Avoidance'. Inf. Sci., 1991, 55, (1–3), pp. 1–26
34. Su, J., Hao, C., Sun, S., et al.: 'Revisiting the Design of In-Memory Dynamic Graph Storage'. Proc. ACM Manag. Data, 2025, 3, (1), Art. no. 70
35. Wei, Y., Trautwein, D., Psaras, Y., et al.: 'The Eternal Tussle: Exploring the Role of Centralization in IPFS'. Proc. 21st USENIX Symp. Netw. Syst. Des. Implement. (NSDI), Santa Clara, CA, USA, 2024, Art. no. 25
36. Lupaiescu, S., Cioata, P., Turcu, C.E., et al.: 'Centralized vs. Decentralized: Performance Comparison between BigchainDB and Amazon QLDB'. Appl. Sci., 2023, 13, (1), Art. no. 499
37. Leutenegger, S.T., Dias, D.: 'A Modeling Study of the TPC-C Benchmark'. SIGMOD Rec., 1993, 22, (2), pp. 22–31
38. Cooper, B.F., Silberstein, A., Tam, E., et al.: 'Benchmarking Cloud Serving Systems with YCSB'. Proc. 1st ACM Symp. Cloud Comput., Indianapolis, IN, USA, 2010, pp. 143–154
39. Dinh, T.T.A., Wang, J., Chen, G., et al.: 'BLOCKBENCH: A Framework for Analyzing Private Blockchains'. arXiv preprint arXiv:1703.04057, 2017