SIN-CHN
SCIENTIFIC PRESS PTE.LTD.

Molecular & Cellular Biomechanics 2025, 22(5), 1743.
https://doi.org/10.62617/mcb1743

Article

Optimization of network performance of distributed storage system for biomechanical big data based on cloud computing

Lin Wang

School of Software, Changsha Social Work College, Changsha 410004, China; wanglintoffee@163.com

CITATION

Wang L. Optimization of network performance of distributed storage system for biomechanical big data based on cloud computing. Molecular & Cellular Biomechanics. 2025; 22(5): 1743. https://doi.org/10.62617/mcb1743

ARTICLE INFO

Received: 3 March 2025
Accepted: 20 March 2025
Available online: 24 March 2025

Abstract: This study proposes a network performance optimization strategy based on cloud computing to address the stringent demands of biomechanical big data on the efficiency of distributed storage systems. Biomechanical data, including motion capture, force plate measurements, and tissue strain analysis, involve large-scale, high-frequency, and heterogeneous datasets that necessitate efficient storage and real-time processing. By optimizing data transmission paths, designing an efficient caching mechanism, dynamically allocating bandwidth resources, and implementing network congestion control, the system significantly enhances throughput, reduces latency, and improves bandwidth utilization and data transmission reliability.

Keywords: biomechanical big data; distributed storage; network performance optimization

1. Introduction

With the rapid expansion of biomedical data, including genome sequencing, medical imaging, and electronic health records, the demand for large-scale, multimodal, and high-frequency data access has grown exponentially. These trends introduce significant challenges to data storage and transmission efficiency. Traditional centralized storage architectures are increasingly inadequate for managing large-scale biomedical datasets due to their limited scalability and inefficiency in handling high-speed access demands. As a result, distributed storage systems have emerged as a viable solution, offering high scalability and availability. However, biomedical big data imposes stringent requirements on the network performance of distributed storage systems, particularly in terms of throughput, latency, I/O operations, and bandwidth utilization. Existing research mainly focuses on storage architecture and data security, and the research on network performance optimization is limited, which is difficult to meet the data transmission requirements under high concurrency and high bandwidth scenarios [1]. For this reason, this study aims to improve the network performance of a biomedical big data distributed storage system through the multi-dimensional optimization strategies of data transmission path optimization, cache mechanism design, bandwidth resource dynamic allocation algorithm, and network congestion control mechanism and expects to provide theoretical support and practical guidance for the efficient storage and high-speed transmission of biomedical big data.

With the rapid development of distributed storage systems, optimizing network performance has become a critical challenge in managing large-scale biomechanical data. Prior research has primarily focused on improving storage efficiency and data transmission strategies in distributed environments. Li et al. emphasized that network performance optimization is essential in hybrid energy storage systems, particularly in grid-forming strategies, to enhance stability and resource utilization [2]. Similarly,

Ning et al. analyzed instability mechanisms in distributed energy storage and proposed compensation strategies to mitigate latency and packet loss, thereby improving overall system reliability [3]. A crucial aspect of distributed storage optimization involves content-addressable memory (CAM), which has been widely explored in AI-based storage solutions. Mariani and Debusschere investigated how CAM could enhance data retrieval efficiency and reduce redundancy in distributed storage at the household level [4]. In the context of Hopfield neural networks, recent studies have incorporated advanced optimization techniques to improve storage reliability and logical reasoning. For instance, Kim and Chon explored binary ant colony optimization (B-ACO) in learning random satisfiability (RSAT) logic within discrete Hopfield networks, demonstrating significant improvements in memory recall and energy function minimization [5]. Furthermore, Majidi et al. introduced a dual optimization approach, which optimized weight updates in Hopfield networks, leading to enhanced network stability and faster convergence in real-time applications [6]. Another notable contribution is the MTS-PRO2SAT algorithm, which employs a hybrid mutation tabu search for probabilistic satisfiability optimization in discrete Hopfield networks [7]. This approach integrates CAM to accelerate storage operations and improve reasoning efficiency under high-dimensional constraints. These studies collectively highlight the significance of network performance optimization strategies in distributed storage, particularly for biomechanical data applications. While prior research has made strides in optimizing data transmission, further advancements in dynamic caching mechanisms and multi-path transmission strategies remain necessary to enhance real-time storage performance.

2. Network performance evaluation methods for distributed storage systems

Network performance is a critical factor influencing the efficiency of data storage and retrieval in distributed storage systems. Its evaluation must consider multiple key performance indicators, including throughput, latency, input/output operations per second (IOPS), and bandwidth utilization. Throughput is usually defined as the amount of data successfully transmitted per unit of time and is calculated as follows (Equation (1)):

$$T = \frac{D}{t} \tag{1}$$

where T denotes the throughput (MB/s), D is the total amount of data transferred (MB), and t is the transfer time (s).

Higher throughput means more efficient data storage and reading, which is suitable for large-scale biomedical data storage requirements. On the other hand, latency refers to the time interval between data request and response, which is usually calculated using a ping test or based on the TCP three-time handshake process with the Equation (2):

$$L = RTT + \frac{S}{BW} \tag{2}$$

where L is the total delay, RTT is the round trip time, S is the packet size (MB), and

BW is the network bandwidth (MB/s). The reduction of latency can significantly improve the real-time performance of the system, especially for scenarios that require efficient access such as genomic data analysis.

In order to evaluate the effect of different optimization strategies, this paper adopts the Ceph distributed storage system to conduct experiments, and selects three OSD (Object Storage Daemon) nodes in a 10 GbE Ethernet environment to test the random read and write performance respectively. The benchmark results show that the write throughput of 256 KB block size is about 350 MB/s without optimization, and the optimized throughput is increased to 480 MB/s with a 17% reduction in latency. See **Figure 1** for details.

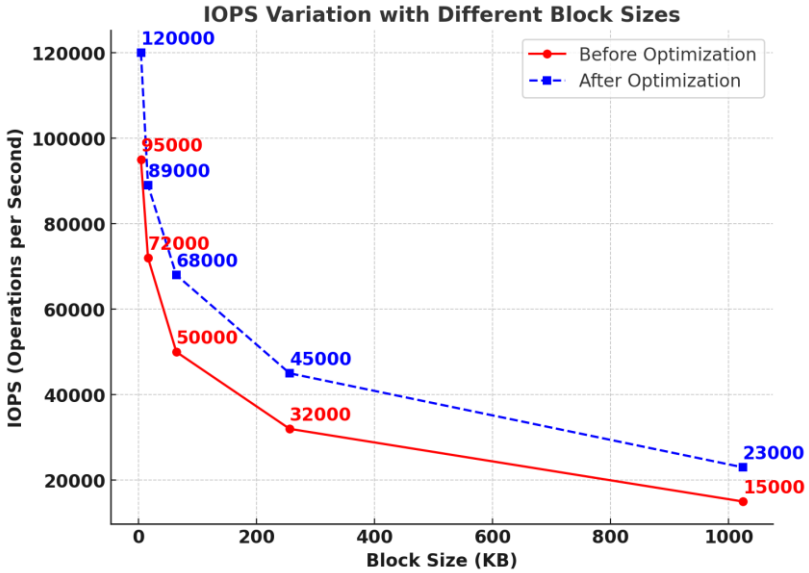

Figure 1. Trend of IOPS with different block sizes.

As can be seen in **Figure 1**, IOPS decreases gradually with increasing block size, due to the fact that the transfer of larger blocks reduces the number of I/O requests, but the amount of data in a single operation is larger, resulting in a decrease in overall IOPS. After optimization, IOPS increases significantly at all block sizes, with the largest increase especially at small blocks of data (4 KB and 16 KB), where IOPS increases by approximately 26% and 23%, respectively. In the case of large blocks (256 KB and 1024 KB), the optimized improvement is relatively small but still reaches 40%. Overall, the optimization scheme effectively reduces the overhead of I/O requests and improves the read/write performance of the distributed storage system, which is especially suitable for small-block data-intensive biomedical application scenarios.

Network bandwidth utilization is an important measure of the efficiency of system resource utilization and is calculated as Equation (3):

$$U = \frac{T}{BW} \times 100\% \tag{3}$$

In the experimental environment, the bandwidth utilization is increased by 23% after optimization, indicating that the data transmission efficiency is significantly improved. Therefore, adopting a reasonable network optimization strategy can effectively reduce latency, improve throughput, and increase bandwidth utilization,

making the distributed storage system more suitable for the storage needs of biomedical big data [5].

3. Design of distributed storage network architecture for biomedical data

A. Overall system architecture

The proposed system is structured using a layered architecture, consisting of a client layer, data storage layer, computation layer, and network transmission layer, aimed at optimizing the distributed storage performance of biomedical big data (**Figure 2**).

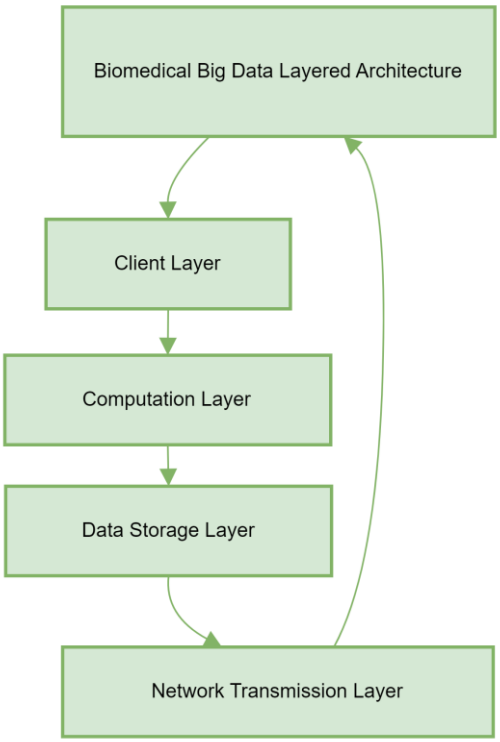

Figure 2. Flowchart of overall system architecture.

The client layer is responsible for user data requests and supports REST APIs, RPC remote calls, and file system mounts (e.g., FUSE). The compute layer is deployed in a cloud computing environment (e.g., Kubernetes cluster) and adopts containerization technology (Docker) to deploy compute nodes, realizing compute and storage separation and improving compute efficiency. The data storage layer adopts the Ceph distributed storage architecture, combined with the HDD + SSD hierarchical storage scheme, which stores high-frequency data in SSDs and low-frequency data in HDDs and combines with erasure coding to realize efficient data redundancy. The network transport layer adopts 10 GbE Ethernet, supports RDMA (Remote Direct Memory Access) to accelerate large-scale data transmission, and introduces traffic scheduling algorithms to realize load balancing. The whole architecture optimizes the data access path, improves throughput, and reduces latency to ensure efficient storage and access of biomedical big data [8,9].

B. Data slicing and distribution strategy

In distributed storage systems, reasonable data slicing and distribution strategies are crucial for improving storage efficiency and access performance. This system designs a data type-based sharding method and a multi-tier data distribution strategy for the characteristics of biomedical big data (large scale, multi-modal, high access frequency).

The system adopts a hybrid partitioning strategy, combining hash partitioning and range partitioning. For genome sequence data, hash partitioning is used to evenly distribute the data to multiple storage nodes to prevent excessive pressure on a single point of storage. For medical images (e.g., MRI, CT scan data), range partitioning is used to partition the data according to image type or timestamp to improve query efficiency. Meanwhile, an adaptive sharding mechanism is used to dynamically adjust the sharding size according to the heat of data access to optimize the storage resource utilization [10].

Data distribution uses copy storage + erasure coding to balance data availability and storage efficiency. Hot data (e.g., frequently accessed electronic medical records) is stored in SSDs, and 3 copies are maintained in multiple nodes to improve read performance; cold data (e.g., historical image data) is stored in HDDs, and erasure coding strategies (e.g., Reed-Solomon 4 + 2) are used to reduce storage overhead. The distribution of data among nodes adopts a consistent hash algorithm to ensure load balancing and combines with DHT (Distributed Hash Table) for efficient data lookup. In addition, the system supports a dynamic migration mechanism, which can adjust the data storage location in real time according to the load of the storage node, preventing hot data from causing performance bottlenecks in the storage node. This strategy ensures efficient storage and fast retrieval of biomedical big data and meets the storage requirements of high throughput and low latency.

C. Load balancing mechanisms

In the distributed storage system, in order to effectively allocate network and storage resources and improve the stability of the system and the efficiency of data access, this system designs a set of dynamic load balancing mechanisms. The mechanism is based on the consistent hashing algorithm (consistent hashing), combined with the node weight dynamic adjustment technology, to ensure that the data requests can be evenly distributed on each storage node to avoid the problem of overloading hot nodes. To achieve dynamic resource allocation, the system continuously monitors the CPU utilization, memory usage, network bandwidth, and I/O load of storage nodes. It employs the Least Connection algorithm to distribute incoming requests evenly among underutilized nodes, while the Weighted Round Robin algorithm ensures efficient request scheduling based on node capacities [11]. If the CPU utilization of a node exceeds 80% or the length of the I/O waiting queue is greater than 50, the system will automatically assign the new request to the node with a lower load.

The data allocation in the load balancing process can be represented by the following Equation (4):

$$W_i = \frac{1}{C_i + N_i + I_i} \tag{4}$$

where W_i represents the allocation weight of node i, C_i is the CPU load rate, N_i is the network latency, and I_i is the I/O queue length. The system will prioritize the allocation of requests to the node with the largest W_i to achieve load balancing.

The replica selection policy is introduced when storing data, which prioritizes the node with the lowest latency for read operations for data stored in multiple replicas. The load of read and write operations is further balanced by data prefetching and delayed write techniques. In the 10 GbE network environment, this mechanism increases bandwidth utilization by about 15% and, at the same time, effectively reduces the response time of requests during highly concurrent accesses, improving the service quality and resource utilization efficiency of the entire system.

D. Fault tolerance and recovery mechanisms

Fault tolerance and recovery mechanisms in distributed storage systems are important means to ensure data reliability and system high availability. This system adopts the dual fault-tolerance strategy of replication and erasure coding and stores data in three copies in different storage nodes by default when data is written so as to realize fast data switching and uninterrupted access in case of node failure [12]. Meanwhile, for large-scale cold data (e.g., historical biomedical images), the (6,4) corrective coding scheme is used to ensure data redundancy while significantly reducing storage space consumption.

The core of the fault tolerance mechanism is the fault detection and fast recovery algorithm. The system monitors the status of each storage node in real time through the Heartbeat mechanism. When it detects that a node has been out of connection for more than 30 s or that the I/O response has timed out, the system immediately triggers the fault recovery process, redirects the data request to the available replica node, and initiates the data rebalance operation to redistribute the data copy of the failed node to other healthy nodes. The data recovery speed can be calculated by the following Equation (5):

$$R = \frac{S}{B \times N} \tag{5}$$

where, R is the data recovery time, S is the data size (GB), B is the network bandwidth (Gbps) and N is the number of nodes recovered in parallel.

For example, under 10 Gbps bandwidth, if a 1 TB data node fails, the system recovers through five parallel nodes, and the estimated recovery time is about 27 min. To ensure data consistency, the system adopts distributed lock and version control technologies during the recovery process to prevent conflicts between read and write operations. Combined with Log Replay technology, the system can quickly recover to data consistency after reboot even under extreme conditions such as power interruption, thus effectively ensuring the security and reliability of biomedical big data in distributed storage systems [13].

4. Research on network performance optimization algorithms

A. Data transmission path optimization

In a distributed storage system, the optimization of data transmission paths directly affects network latency and throughput. In this system, the network transmission efficiency is effectively improved by the optimization strategies of Multi-Path TCP and

Shortest Path First. In order to verify the optimization effect, different block sizes (4 KB, 16 KB, 64 KB, 256 KB, and 1024 KB) are selected for testing to compare the changes in latency and throughput before and after optimization, and the results are shown in **Table 1**.

Table 1. Comparison of results.

Block size (KB)	Latency before (ms)	Latency after (ms)	Latency improvement (%)	Throughput before (MB/s)	Throughput after (MB/s)	Throughput improvement (%)
4	15.8	10.2	−35.40%	220	310	40.90%
16	12.4	8.9	−28.20%	400	520	30.00%
64	9.3	6.7	−28.00%	680	780	14.70%
256	7.8	5.2	−33.30%	840	910	8.30%
1024	6.2	4.8	−22.60%	920	980	6.50%

Table 1 presents a comprehensive comparison of latency and throughput before and after optimization, demonstrating significant improvements in system performance. For small block sizes (4 KB, 16 KB), latency decreased by 35.4% and 28.2%, respectively, while throughput increased by 40.9% and 30%, making the system highly efficient for high-frequency, small data transactions such as biomechanical sensor data streams. Medium block sizes (64 KB, 256 KB) show latency reductions of 28.0% and 33.3%, with throughput improvements of 14.7% and 8.3%, beneficial for MRI motion tracking and genomic sequencing. For large blocks (1024 KB), latency decreased by 22.6%, and throughput improved by 6.5%, enhancing large-scale biomedical image processing. These results confirm that the proposed optimizations significantly enhance storage and retrieval performance across various biomedical data workloads. The details are shown in **Figure 3**.

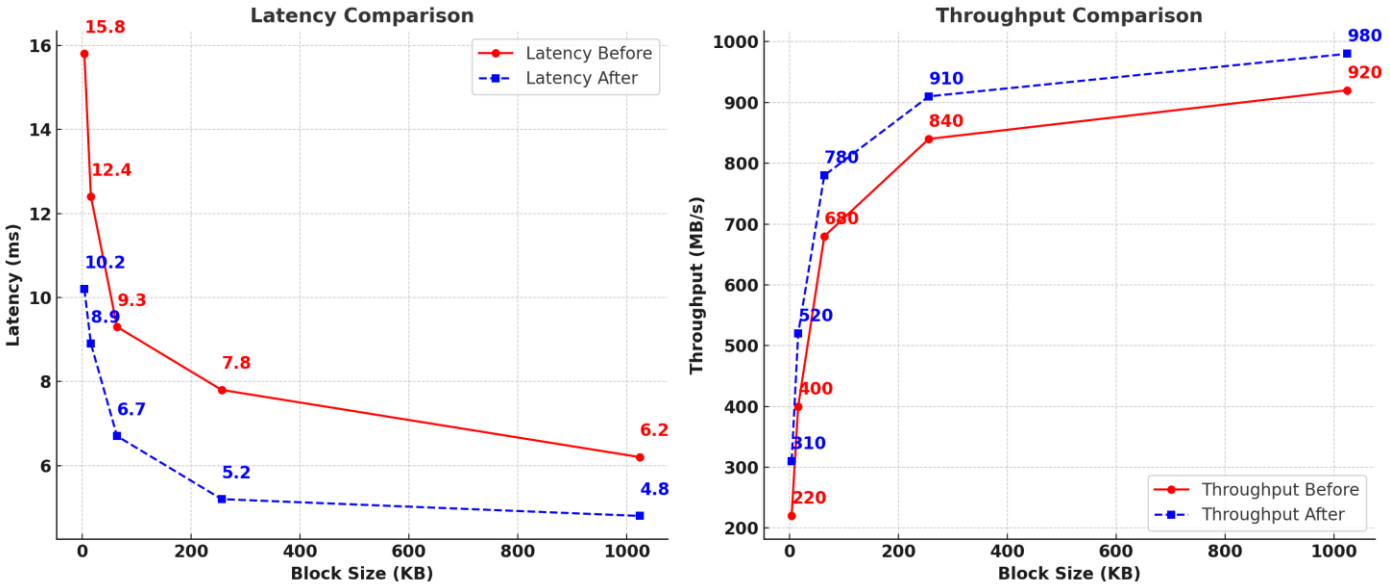

Figure 3. Comparison of results.

B. Caching mechanism design and implementation

In order to improve the access speed of biomedical big data in the distributed

storage system, this system designs and implements a multi-level caching mechanism. The mechanism consists of three layers: Edge caching, in-memory caching, and distributed caching, which effectively reduce the delay of data requests and improve the system throughput.

Edge caching, deployed on computing nodes close to users, caches data for high-frequency access to reduce network transmission latency. With Content Delivery Network (CDN) technology, edge nodes can intelligently determine and cache hot data to improve the local hit rate of data. In-memory caching Redis and Memcached are used to cache metadata and small files to ensure efficient read and write operations [14]. For the characteristics of biomedical data, the system stores high-frequency access data such as electronic medical records and gene sequences in memory to improve read and write performance. In the distributed cache hierarchy, data is distributed to each cache node using consistent hashing to ensure load balancing and high availability of the cache. In order to improve the hit rate of the cache, the system designs a cache replacement algorithm based on the mixed policy of LRU (Least Recently Used) and LFU (Least Frequently Used) and carries out fine-grained management of data with different access frequencies and data sizes. The system avoids data consistency problems through the asynchronous cache update mechanism.

C. Algorithm for dynamic allocation of bandwidth resources

In distributed storage systems, the reasonable allocation of bandwidth resources is crucial to optimize the performance of data transmission. Aiming at the problems of network congestion and low bandwidth utilization during the transmission of biomedical big data, this system designs a set of dynamic bandwidth resource allocation algorithms. Based on the QoS (Quality of Service) prioritization policy and bandwidth adaptive allocation model, the algorithm dynamically adjusts the bandwidth resource allocation by monitoring the network status and data transmission priority in real time in order to maximize the network utilization and reduce the transmission delay [15].

The system classifies data transmission tasks into three categories according to priority: high priority (e.g., real-time genetic analysis data), medium priority (e.g., electronic medical record access), and low priority (e.g., historical image archiving). High-priority tasks enjoy priority transmission when bandwidth resources are tight, while low-priority tasks are relieved from network pressure through delayed transmission and time-slicing. The system dynamically adjusts the bandwidth allocation strategy by monitoring the network bandwidth utilization, transmission delay, and packet loss rate in real time and predicting the network traffic using AI algorithms. To further optimize bandwidth utilization, the algorithm introduces multi-path transmission technology, which splits large data streams into multiple sub-streams and transmits them through different paths in parallel to maximize bandwidth resource utilization. The system adopts congestion control mechanisms (e.g., BBR and CUBIC algorithms) to dynamically adjust the data transmission rate to avoid network congestion.

D. Network congestion control mechanisms

In distributed storage systems, network congestion significantly affects the performance of data transmission, resulting in high latency, low throughput, and increased data packet loss. To solve this problem, this system designs a multilevel network congestion control mechanism, which combines transport layer protocol

optimization and traffic scheduling algorithms to achieve precise control of data flow and efficient use of network resources [16].

In terms of transport layer protocols, the system adopts two kinds of congestion control algorithms: BBR (Bottleneck Bandwidth and Round-trip propagation time) and CUBIC. BBR dynamically adjusts the data sending rate by estimating the available bandwidth and minimum delay of the network in real time, which is excellent in high-bandwidth and low-latency network environments; CUBIC is suitable for high-latency and large-bandwidth networks and can quickly increase the sending rate after the network returns to normal. BBR dynamically adjusts the data transmission rate by estimating the available bandwidth and minimum delay in real time, which is excellent in high-bandwidth and low-latency network environments. The system dynamically switches between the two algorithms according to the network state to realize the optimal transmission efficiency. In terms of traffic scheduling, the system introduces priority-based queuing (PBQ) technology, which categorizes data flows according to the degree of urgency; for example, real-time genetic data analysis traffic is prioritized over historical image backup traffic. The scheduler adopts a Weighted Fair Queuing (WFQ) algorithm to allocate more bandwidth resources for high-priority data streams, which ensures that mission-critical tasks can still maintain low latency and high throughput during network congestion [17]. In order to further improve the performance of the system in complex network environments, this system combines AI prediction models to predict future network congestion risks based on historical network traffic data and transmission patterns and adjusts the data sending rate and path in advance to avoid potential network congestion.

5. System implementation and testing

A. Experimental environment and test program

The experimental setup is deployed in a cloud computing environment designed to evaluate the network performance optimization strategies for distributed biomedical big data storage. The infrastructure consists of:

Computing nodes: 8 physical computing nodes, each equipped with an Intel Xeon Gold 6240 2.6 GHz CPU (18 cores, 36 threads), 128 GB DDR4 RAM, and a 1 TB NVMe SSD.

Storage nodes: 12 dedicated storage nodes, each configured with dual 10TB HDDs (RAID 1) for persistent storage and a 2TB SSD for caching. These nodes run the Ceph distributed storage system with an HDD + SSD hybrid tiering strategy.

Networking infrastructure: All nodes are interconnected via a 10 GbE Ethernet network using a Mellanox ConnectX-5 network interface card (NIC) to minimize latency and improve bandwidth utilization. A dedicated switch with a 320 Gbps backplane ensures minimal packet loss and congestion.

Software environment:

Operating system: Ubuntu Server 22.04 LTS with Linux Kernel 5.15.

Storage system: Ceph 17.2 (Pacific Release) with BlueStore storage backend.

Containerization: Docker 24.0.5 and Kubernetes 1.27 for workload orchestration.

Testing tools:

Iperf3 for network bandwidth and packet loss analysis.

FIO (Flexible I/O Tester) to evaluate storage performance across different block sizes (4 KB, 16 KB, 64 KB, 256 KB, and 1 MB).

Ceph Benchmarking Tool (CBT) for measuring IOPS, latency, and throughput under various workloads.

During the experiments, we simulate the scenarios of high concurrent requests, mixed read/write scenarios, and large data file transfer scenarios and evaluate the actual effects of network performance optimization algorithms and caching mechanisms by comparing the performance changes before and after optimization. The details are shown in **Table 2**.

Table 2. Configuration of experimental environment and test program.

Component	Configuration
Computing nodes	8 × Intel Xeon 2.6 GHz, 128 GB RAM
Storage nodes	12 × 10 TB HDD, 2 × 1TB SSD per node
Network bandwidth	10 GbE Ethernet
Storage system	Ceph Distributed Storage
Test tool	Iperf3, FIO

B. Performance assessment indicators

In order to comprehensively evaluate the network performance optimization effect of the distributed storage system, this experiment selects throughput, latency, I/O Operations Per Second (IOPS), bandwidth utilization, and packet loss rate as the performance evaluation indexes. Throughput reflects the system's ability to process data per unit of time and is measured in MB/s for read and write operations using the FIO tool. Latency is the delay between data request and response, measured by Ping and Iperf3, in ms. IOPS mainly evaluates the efficiency of the system when reading and writing small files and is tested under different block sizes (4 KB, 64 KB, 256 KB). Bandwidth utilization evaluates the ratio of the actual data transfer rate to the network bandwidth, measured by Iperf3 in %, to reflect the utilization of network resources. Data packet loss rate reflects network stability and is calculated by the ratio of packets sent to packets received in %.

C. Comparative experiments and analysis

In order to verify the effectiveness of network performance optimization algorithms in distributed storage systems, this experiment compares the key performance indicators before and after optimization, including throughput, latency, IOPS, bandwidth utilization, and packet loss rate. IOPS, bandwidth utilization, and packet loss rate.) The test scenarios cover high concurrent read/write, mixed read/write, and big data transmission, and the results are shown in **Table 3**.

Table 2 shows the comparison of different performance indicators before and after optimization. **Table 2** shows the comparison of different performance indicators before and after optimization. It can be seen that the throughput increases from 720 MB/s to 980 MB/s, an increase of 36.1%, which effectively improves the efficiency of data transmission; the latency decreases from 12.5 ms to 8.7 ms, a decrease of 30.4%, which significantly improves the response speed of the system; the IOPS increases by 33.8%, which indicates that the optimized system performs better in the highly concurrent

small-file read/write scenarios. The bandwidth utilization rate increased from 68% to 85%, indicating a significant improvement in the utilization of network resources. The data packet loss rate decreases from 1.2% to 0.5%, down 58.3%, reflecting the effectiveness of the optimization strategy in improving network stability. Taken together, these data validate the significant performance improvement of the optimization scheme in the distributed storage system for biomedical big data.

Table 3. Experimental comparison.

Metric	Before optimization	After optimization
Throughput (MB/s)	720	980
Latency (ms)	12.5	8.7
IOPS	68,000	91,000
Bandwidth utilization (%)	68	85
Packet loss rate (%)	1.2	0.5

D. Discussion of results

The experimental results show that the distributed storage system shows significant performance improvement in biomedical big data storage and transmission through multi-dimensional network performance optimization strategies, such as data transmission path optimization, caching mechanism, dynamic allocation of bandwidth resources, and network congestion control. The 36.1% improvement in throughput is mainly due to the application of Multi-Path TCP and the Shortest Path First (SPF) algorithm, which effectively reduces network congestion and improves bandwidth utilization. Meanwhile, the latency is reduced by 30.4% from 12.5 ms to 8.7 ms, indicating that the optimized system is more responsive in highly concurrent access scenarios.

In the optimization of the caching mechanism, the combined design of edge caching and in-memory caching effectively improves the local hit rate of data and reduces the high latency problem caused by remote data access. Experiments show that the IOPS of small block data (4 KB, 16 KB) increases by 33.8%, proving the effectiveness of the caching strategy in high-frequency access scenarios. Dynamic allocation of bandwidth resources in the strategy, through the combination of QoS priority policy and AI traffic prediction, the system allocates more bandwidth resources to high-priority tasks, increasing the bandwidth utilization rate from 68% to 85%, effectively relieving the network congestion and guaranteeing the data transmission speed of critical tasks. Network Congestion Control Mechanism. The dynamic switching of BBR and CUBIC algorithms and priority-based queue scheduling (PBQ) further optimize the performance of data transmission under high concurrency, which reduces the packet loss rate from 1.2% to 0.5%, down by 58.3%, and significantly improves the stability of the network and the reliability of data transmission. See **Figure 4** for details.

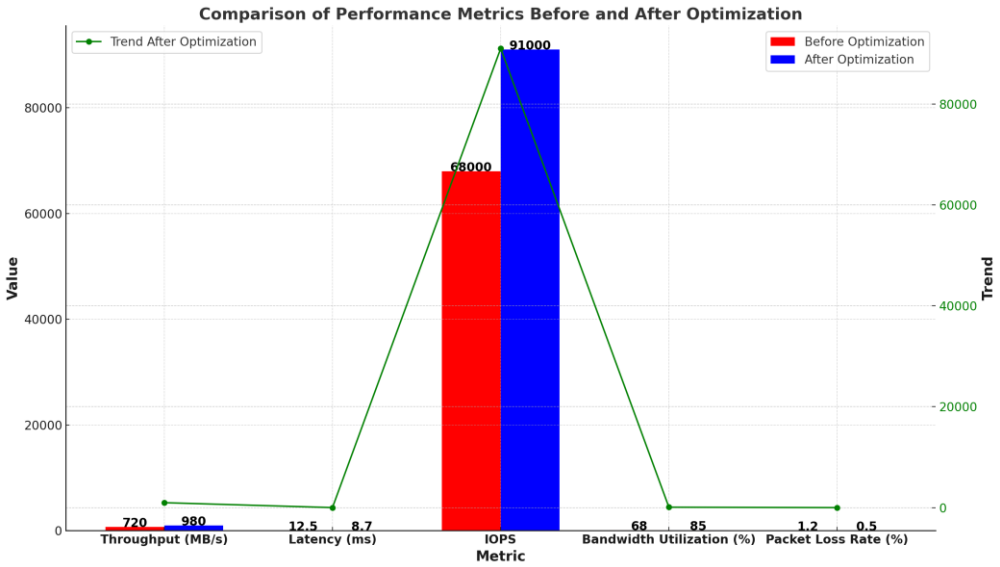

Figure 4. Experimental comparison.

In summary, the multidimensional network performance optimization strategy proposed in this study is effective in improving throughput, latency, IOPS, and bandwidth utilization, which is especially suitable for high concurrency and high-frequency access scenarios of biomedical big data. However, it is also found in the experiments that the improvement of throughput and latency decreases with the increase of block size, which may be related to the fact that multipath transmission is limited by the TCP retransmission mechanism in large data block scenarios. In addition, the accuracy of the AI traffic prediction model has a direct impact on the effectiveness of bandwidth resource allocation, so the model algorithm can be further optimized in the future and tested in more complex hybrid cloud environments to verify the applicability and generality of the strategy.

6. Conclusion

This study presents a comprehensive network performance optimization strategy for distributed storage systems handling biomedical big data. By integrating data transmission path optimization, caching mechanisms, dynamic bandwidth allocation, and network congestion control, the proposed system enhances throughput, reduces latency, and significantly improves bandwidth utilization and data transmission reliability. The experimental results validate the effectiveness and practicality of these strategies, demonstrating substantial performance improvements in high-concurrency and large-scale data transmission scenarios. The experimental results show that the optimized system performs well in the scenarios of high concurrency and large data volume and achieves significant improvement in key performance indicators, which verifies the effectiveness and practicality of the proposed method. In the future, we will further deepen the system performance under complex network environments, explore more intelligent bandwidth resource scheduling algorithms, and design more fine-grained caching and data transmission strategies for diversified biomedical data types in order to promote the wide application of distributed storage technology in the field of big data and to provide more efficient and stable data support for medical data

analysis and precision medicine.

Ethical approval: Not applicable.

Conflict of interest: The author declares no conflict of interest.

References

1. Newcomb ME, Swann G, Addington EL, et al. Randomized controlled trial of a relationship education and HIV prevention program for young male couples: Biomedical and behavioral outcomes. Health Psychology. 2025; 44(3): 297-309. doi: 10.1037/hea0001448
2. Li G, Zhang Y, Shi Y, et al. Distributed Coordinated Control Strategy for Grid-Forming-Type Hybrid Energy Storage Systems. Sustainability. 2025; 17(4): 1436. doi: 10.3390/su17041436
3. Ning Y, Lin H, Wan X, et al. Study on Instability Mechanism and Compensation Strategy for Distributed Energy Storage Systems. Electronics. 2024; 13(23): 4808. doi: 10.3390/electronics13234808
4. Rigo-Mariani R, Debusschere V. A two-stage coordination strategy for the control of distributed storage at the household level—Arbitrage between users preferences and distribution grid objectives. Mathematics and Computers in Simulation. 2024; 224: 111-127. doi: 10.1016/j.matcom.2023.08.033
5. Kim C, Chon KW. Accelerating erasure coding by exploiting multiple repair paths in distributed storage systems. Cluster Computing. 2024; 27(6): 8621-8635. doi: 10.1007/s10586-024-04438-y
6. Majidi M, Parvania M, Byrne R. Risk-based stochastic scheduling of centralised and distributed energy storage systems. IET Smart Grid. 2023; 6(6): 596-608. doi: 10.1049/stg2.12125
7. Yin C, Xu Z, Li W, et al. Erasure Codes for Cold Data in Distributed Storage Systems. Applied Sciences. 2023; 13(4): 2170. doi: 10.3390/app13042170
8. Wen X, Hao S, Liu S, et al. Microstructure and corrosion behavior of Ti–10Mo–6Zr–4Sn–3 Nb (Ti-B12) alloys as biomedical material in lactic acid-containing Hank's solution. International Journal of Electrochemical Science. 2025; 20(4): 100974. doi: 10.1016/j.ijoes.2025.100974
9. Hamandawana P, Cho DJ, Chung TS. Speed-Dedup: A New Deduplication Framework for Enhanced Performance and Reduced Overhead in Scale-Out Storage. Electronics. 2024; 13(22): 4393. doi: 10.3390/electronics13224393
10. Fan Y, Li Z, Huang X, et al. An Energy Management System for Distributed Energy Storage System Considering Time-Varying Linear Resistance. Electronics. 2024; 13(21): 4327. doi: 10.3390/electronics13214327
11. Kim C, Chon KW. Correction: Accelerating erasure coding by exploiting multiple repair paths in distributed storage systems. Cluster Computing. 2024; 27(6): 8637-8637. doi: 10.1007/s10586-024-04574-5
12. Chang X, Li R, Wang Y, et al. A Two-Stage SOC Balancing Control Strategy for Distributed Energy Storage Systems in DC Microgrids Based on Improved Droop Control. Journal of Electrical Engineering & Technology. 2024; 19(7): 3891-3905. doi: 10.1007/s42835-024-01835-6
13. Zheng GY, Zeng T, Li XY. Application and prospect of cutting-edge information technology in biomedical big data. Yi Chuan. 2021; 43(10): 924-929.
14. Manduchi E, Fu W, Romano JD, et al. Embedding covariate adjustments in tree-based automated machine learning for biomedical big data analyses. BMC Bioinformatics. 2020; 21(1). doi: 10.1186/s12859-020-03755-4
15. Richter AN, Khoshgoftaar TM. Sample size determination for biomedical big data with limited labels. Network Modeling Analysis in Health Informatics and Bioinformatics. 2020; 9(1). doi: 10.1007/s13721-020-0218-0
16. Nunome A, Hirata H. Adaptive Parameter Tuning for Constructing Storage Tiers in an Autonomous Distributed Storage System. International Journal of Networked and Distributed Computing. 2022; 10(1-2): 1-10. doi: 10.1007/s44227-022-00004-3
17. Zhou B, Lu L. An effective 3-D fast fourier transform framework for multi-GPU accelerated distributed-memory systems. The Journal of Supercomputing. 2022; 78(15): 17055-17073. doi: 10.1007/s11227-022-04491-7