Applied Science and Innovative Research
ISSN 2474-4972 (Print) ISSN 2474-4980 (Online)
Vol. 9, No. 1, 2025
www.scholink.org/ojs/index.php/asir

Original Paper

MinIO-Based Efficient Database Backup and Synchronization Strategy: Compressed and Encrypted Storage with Cross-Region Real-Time Synchronization

Ji Lei

Chongqing Youth Vocational & Technical College, Chongqing City, China

Received: January 25, 2025 Accepted: February 21, 2025 Online Published: February 27, 2025

doi:10.22158/asir.v9n1p136 URL: http://doi.org/10.22158/asir.v9n1p136

Abstract

This paper presents a MinIO-driven database backup and synchronization framework that tackles legacy issues like slow backups, costly storage, and cross-region sync gaps. By merging LZ4 compression (73.5% size reduction) with AES-256-GCM encryption, the system slashes storage needs while hardening security. MinIO's Webhook system keeps local and remote backups in lockstep—even during network outages—through real-time sync. Tests on an i5-11400/8GB setup showed consistent 80+MB/s uploads and 380+MB/s downloads, proving MinIO's ability to handle rapid transfers. While the strategy excels under typical loads, scaling to high-traffic environments requires tuning. Next-phase work will streamline encryption workflows and boost concurrency handling for broader enterprise adoption.

Keywords

Data Backup, MinIO, Data Compression, Data Encryption, Cross-Region Synchronization

1. Introduction

In contemporary data management, legacy backup systems struggle with extended downtime during backups, unsustainable storage expenditures, and rigid scalability. For a 24/7 database system, backups should be implemented right after a large volume of data has been updated (Zhao, Bu, Pang, & Cai, 2024). These challenges necessitate the adoption of advanced backup architectures that reconcile speed with security. A key vulnerability in distributed backup systems lies in unauthorized data alterations during transmission—a single compromised node could propagate corrupted data across regions. Mitigating such risks demands robust safeguards: real-time encryption, lossless compression, and cryptographic validation. This study introduces a MinIO-based mechanism that integrates compression

and encryption for cross-region backups, our approach threads compression (LZ4) and encryption (AES-256-GCM) into a unified workflow, slashing storage needs by 73.5% while enforcing data consistency via MinIO's Webhook triggers. By prioritizing both efficiency and attack resilience, this architecture redefines reliable backup operations.

2. Related Technologies

2.1 MinIO Object Storage Platform

MinIO is an open-source object storage platform optimized for managing massive unstructured datasets through distributed architecture. Researchers tested it on standard hardware and achieved maximum read speeds of 183 GB per second and write speeds of 171 GB per second (Yin & Lu, 2024). MinIO is an open-source distributed storage system specifically designed for handling massive unstructured data. Its architecture emphasizes scalability and reliability, making it particularly suitable for enterprises requiring large-scale data management. Beyond core storage functions, the platform incorporates multiple advanced features to enhance operational efficiency.

(1) High Availability and Data Redundancy

The platform ensures continuous service through distributed architecture with built-in redundancy mechanisms. Employing Erasure Coding by default, MinIO splits data into fragments stored across multiple servers. This design allows data to remain accessible even during partial server failures, effectively eliminating single points of failure. The system's resilience extends to various hardware and network outage scenarios.

(2) Erasure Coding and Data Durability

Through erasure coding technology, MinIO redundantly stores data blocks across cluster nodes. This approach not only guarantees data safety but also optimizes storage utilization. The architecture can withstand simultaneous failures in up to half of the cluster nodes without data loss, making it ideal for mission-critical systems requiring absolute data integrity.

(3) Automatic Data Replication

The system automatically copies data across multiple nodes within the cluster. Continuous node health monitoring ensures immediate replacement of failed components, maintaining persistent data availability. This self-healing mechanism operates without manual intervention, preserving data consistency during node transitions.

(4) Load Balancing and Performance Optimization

Built-in load balancing dynamically distributes data storage locations across nodes. This intelligent allocation prevents individual node overloads and maintains stable performance under heavy workloads. The adaptive resource management ensures consistent throughput during peak operational demands.

(5) Security Features

Multi-layered security protections include server-side data encryption, detailed permission controls, and comprehensive operation logs. Data-at-rest encryption safeguards stored information, while

granular access policies enable precise user permission management. Audit trails record all system activities to meet enterprise compliance requirements.

(6) Hybrid Deployment Flexibility

Supporting S3-compatible interfaces, MinIO adapts to diverse deployment environments including on-premises infrastructure, cloud platforms, and hybrid configurations. This deployment versatility allows organizations to optimize their storage architecture based on specific cost, scalability, and control requirements.

MinIO's combination of distributed architecture, data protection mechanisms, and security features establishes it as a robust enterprise storage solution. The platform's design ensures continuous data accessibility and consistency across various failure scenarios. With S3 compatibility and flexible deployment options, MinIO provides enterprises with an efficient and secure approach to managing large-scale unstructured data assets.

2.2 Data Compression

Compression slashes storage footprints and accelerates data transfers by shrinking file sizes. Techniques split into two camps: lossless (exact data reconstruction) for databases/texts, and lossy (tolerable quality loss) for media like images. The LZ4 algorithm was used to take advantage of compression to save bandwidth and increase encryption speed (Vijayachandran & Suchithra, 2024). LZ4 is derived from a standard LZ77 compression algorithm and is focused on the compression and decompression speed (Bartik, Ubik, & Kubalik, 2015). We selected LZ4 for its real-time efficiency，crucial for minimizing backup windows without overloading hardware.

2.3 Data Encryption

Encryption scrambles data into unreadable ciphertext, shielding it from unauthorized access. Symmetric methods like AES-256-GCM use a single key for speed, while asymmetric systems trade speed for secure key exchange. The AES-GCM core provides confidentiality by Counter (CTR) mode of block cipher AES, and it also provides integrity and authenticity by GHASH (Sung, Kim, & Shin, 2018). This dual-layer protection—securing data at rest and in transit—aligns perfectly with MinIO backup needs, offering attack resistance without bogging down performance.

2.4 Webhook

A Webhook is a user-defined HTTP callback that enables real-time communication between software systems. It works by sending an HTTP request (typically POST) to a specified URL when a predefined event occurs in the publisher system. This allows the subscriber to receive instant notifications and take immediate actions based on the event data.

Working Principle of Webhook:

(1) **Event Trigger:** An event (e.g., new backup file creation) occurs in the publisher system (MinIO).

(2) **HTTP Request:** The publisher sends an HTTP request containing event data to the subscriber’s URL.

(3) **Action:** The subscriber processes the data and performs tasks (e.g., syncing the backup to a remote server).

Advantages of Webhook:

(1) **Real-Time Updates:** Ensures immediate response to changes in the publisher system.

(2) **Simplicity:** Built on HTTP, making it easy to implement and integrate across systems.

(3) **Flexibility:** Customizable to trigger on specific events and send data to any URL.

3. Mechanism Design

This section details the architecture of the MinIO-based backup system shown in Figure 1. The design integrates LZ4 compression and AES-256-GCM encryption to overcome traditional backup limitations, combining storage optimization with enhanced security. By utilizing MinIO's distributed storage and Webhook notifications, the system ensures real-time local-remote synchronization while maintaining data consistency. The architecture balances operational efficiency with robust protection, catering to enterprise-grade backup requirements.

The workflow begins with application databases serving as data sources. Backup files undergo LZ4 compression to minimize storage footprint and accelerate transfers. Compressed data then receives AES-256-GCM encryption for secure storage and transmission. Processed backups first store in local MinIO instances.

MinIO's Webhook system triggers immediate synchronization to remote locations when detecting local changes. A load balancer distributes encrypted backups across multiple remote servers, optimizing storage utilization and system availability. Finalized data persists in geographically distributed MinIO clusters, forming a reliable disaster recovery foundation.

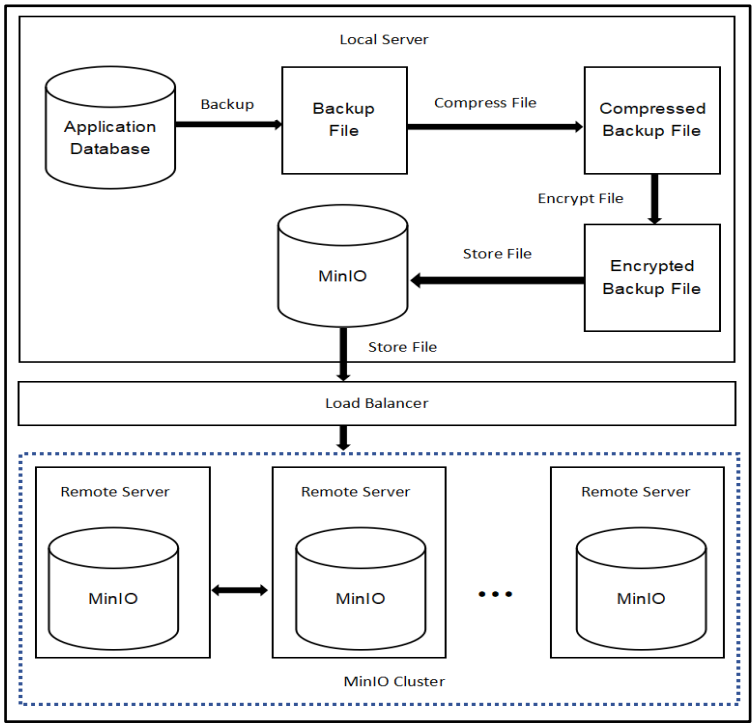

Figure 1. Mechanism design

3.1 System Architecture Overview

The primary components of the architecture include:

(1) **Application Database:** This is the data source that needs to be backed up, storing data generated by applications.

(2) **Backup File:** The raw backup file extracted from the application database, ready for compression and encryption.

(3) **Compressed Backup File:** The backup file that has been processed with a compression algorithm to reduce storage space requirements and improve transmission efficiency.

(4) **Encrypted Backup File:** The compressed backup file that has undergone encryption processing to ensure the security of the data during storage and transmission.

(5) **MinIO:** The object storage service on both local and remote servers, used for storing backup files.

(6) **Local Server:** A server that includes a MinIO instance, used for storing encrypted and compressed local backup files.

(7) **Remote Server:** A server that includes a MinIO instance, used for storing remote backup files synchronized across regions.

(8) **MinIO Cluster:** A cluster composed of multiple MinIO instances on remote servers, used to achieve high availability and disaster recovery of data.

(9) **Load Balancer:** Distributes Encrypted Backup File to different remote servers to optimize resource utilization and ensure high availability of the system.

These components work together to form an efficient, secure, and disaster-recovery-capable database backup and synchronization system.

3.2 Mechanism Process

The following is a detailed description of the mechanism process:

(1) Local Backup of Application Database:

The application database is periodically backed up locally according to set policies.

(2) Compression:

Before transmission, the local database backup files are compressed using the LZ4 algorithm to reduce the demand for storage space and to enhance the efficiency of transmission.

(3) Encryption:

The compressed backup files are then encrypted using the AES-256-GCM algorithm to ensure the confidentiality and integrity of the data during both storage and transmission.

(4) Local Storage:

The encrypted and compressed backup files are stored in the local MinIO instance as a backup.

(5) Cross-Region Synchronization:

Leveraging the MinIO Webhook mechanism, the backup files stored in the local server's MinIO are synchronized in real-time with the remote servers' MinIO instances, ensuring consistency between the local and remote backups.

(6) Load Balancing:

The load balancer distributes the encrypted backup files across various remote servers to optimize resource utilization and ensure high availability of the system.

(7) Remote Storage:

The synchronized data is stored in the remote MinIO cluster, providing a remote backup for disaster recovery purposes.

3.3 LZ4 Compression Algorithm

The LZ4 compression algorithm is employed within our backup system to efficiently compress database backup files. It operates by recognizing repeated sequences and encoding them with references to a dictionary of previously identified patterns, substantially reducing file size. This method not only accelerates the subsequent encryption process but also optimizes storage utilization, ensuring that our backup operations are swift and resource-efficient.

3.4 AES-256-GCM Encryption Algorithm

The AES-256-GCM encryption algorithm plays a pivotal role in securing our database backup files. Operating on the Advanced Encryption Standard (AES) with a 256-bit key size, this algorithm is enhanced with Galois/Counter Mode (GCM) for authentication. It ensures the backup files are protected against unauthorized access by encrypting the data, substituting plain text with ciphertext. The GCM component not only provides data origin authentication but also guarantees integrity, safeguarding the files from tampering during transmission. This implementation is critical for maintaining the confidentiality and integrity of our backup data, thereby bolstering the overall security of the backup system.

3.5 The Webhook Mechanism of MinIO

Applications in the Proposed Strategy:

In the MinIO-based backup system, Webhooks maintain real-time consistency between local and remote backups. When a new encrypted and compressed backup is stored locally, MinIO triggers a Webhook to notify the remote server. The remote server then stores the backup.

Security Considerations:

Webhooks rely on HTTP, necessitating SSL/TLS encryption to protect data in transit. Authentication mechanisms (e.g., API keys, digital signatures) are also essential to verify the publisher's identity and prevent unauthorized access. The strategy further enhances security by encrypting backup files with AES-256-GCM, ensuring data remains secure even if intercepted.

3.6 Communication and Data Synchronization in MinIO Cluster Nodes

MinIO cluster nodes maintain data consistency and availability through networked coordination, employing mechanisms including heartbeat monitoring, shard synchronization, and automated fault recovery. These processes ensure seamless operation across distributed storage environments.

3.6.1 Node Communication Mechanism

(1) Network Protocol: Cluster nodes exchange status updates and sync data through HTTP/HTTPS

connections. This communication covers operational metadata and synchronization tasks.

(2) Heartbeat Monitoring: Nodes broadcast regular heartbeat signals to peers, transmitting operational status and workload metrics. This real-time health tracking enables rapid detection of unresponsive nodes.

3.6.2 Data Synchronization Mechanism

(1) Erasure Coding Implementation: Leveraging Reed-Solomon erasure coding, MinIO divides files into data segments and parity blocks distributed across nodes. This fragmentation ensures redundancy while optimizing storage efficiency.

(2) Write Operations

Shard Creation: Files undergo segmentation into data/parity blocks during ingestion.

Parallel Storage: Blocks simultaneously write to designated nodes.

Validation Phase: Post-write checks verify block integrity across storage locations.

(3) Read Operations

Concurrent Retrieval: Clients fetch required blocks from multiple nodes simultaneously.

Reassembly Process: Original files reconstruct from retrieved segments either client-side or server-side.

3.6.3 Fault Detection and Recovery

(1) Failure Identification

Heartbeat Alerts: Missing multiple consecutive heartbeat pings flags a node as offline.

I/O Failures: Unresponsive nodes during read/write operations trigger failure protocols.

(2) Recovery Workflow

Auto-Rebuild: Surviving blocks regenerate lost data using erasure coding algorithms.

Cluster Rebalancing: Recovered data redistributes across operational nodes to maintain load equilibrium.

3.6.4 Data Consistency Assurance

Concurrency Control: Distributed locking prevents simultaneous conflicting modifications.

Operation Logging: All write actions record in transaction journals for crash recovery.

Version Preservation: Object version history maintains previous iterations during updates.

4. Testing and Evaluation

4.1 Testing Environment

Tests were executed on a desktop machine with an 11th Gen Intel Core i5-11400 CPU (2.60 GHz) and 8GB RAM.

4.2 Performance Evaluation

4.2.1 Performance Testing of MinIO

MinIO delivered stable high-speed transfers—critical for syncing encrypted backups. Uploads stayed above 80MB/s, while downloads exceeded 380MB/s consistently. This performance confirms MinIO's suitability for Comprehensive tests on the i5-11400/8GB platform validated the efficiency and reliability of the MinIO-based backup strategy. Key outcomes:

Compression Efficiency: LZ4 reduced the 188MB.bak file size by 73.5% (3.78× ratio), with compression times of 0.5066s–1.9711s and decompression spanning 1.2517s–5.6835s. These metrics confirm LZ4's dual role in slashing storage needs and accelerating transfers.

Encryption Performance: AES-256-GCM encrypted compressed files 4× faster than raw data—0.42s avg vs 1.66s for uncompressed files. This highlights the benefit of compressing before encryption.

MinIO Transfer: Uploads maintained 80+MB/s, while downloads exceeded 380MB/s consistently, proving MinIO's ability to handle rapid, repeated transfers without performance cliffs.

System Load: Under heavy I/O, decompression peaked at 5.68s, revealing hardware-bound limitations. While the strategy optimizes speed and storage, extreme loads require further tuning for stability.

Overall, the evaluation confirms the strategy's effectiveness in addressing traditional backup flaws—delivering security, speed, and scalability. Future work will refine high-load handling and streamline encryption/compression workflows. reliability even under repeated load.

4.2.2 Performance testing of Compression Efficiency

A Python script evaluated LZ4 compression using a 188MB SQL Server.bak file. Key metrics:

Compression Time: Varied between 0.5066s and 1.9711s due to system load and disk I/O fluctuations.

Decompression Time: Ranged from 1.2517s to 5.6835s, slower than compression but system-dependent.

Compression Ratio: Consistently 3.7832, shrinking files to ~26.4% of original size.

4.2.3 Performance Testing of Encryption

AES-256-GCM encryption tests used the same 188MB.bak file:

Uncompressed File: Encryption took 1.4163s–2.1135s (avg 1.66s).

Compressed File: Encryption accelerated to 0.3012s–0.6174s (avg 0.42s).

4.2.4 Performance Evaluation

Comprehensive tests on the i5-11400/8GB platform validated the efficiency and reliability of the MinIO-based backup strategy. Key outcomes:

Compression Efficiency: LZ4 reduced the 188MB.bak file size by 73.5% (3.78× ratio), with compression times of 0.5066s–1.9711s and decompression spanning 1.2517s–5.6835s. These metrics confirm LZ4's dual role in slashing storage needs and accelerating transfers.

Encryption Performance: AES-256-GCM encrypted compressed files 4× faster than raw data—0.42s avg vs 1.66s for uncompressed files. This highlights the benefit of compressing before encryption.

MinIO Transfer: Uploads maintained 80+MB/s, while downloads exceeded 380MB/s consistently, proving MinIO's ability to handle rapid, repeated transfers without performance cliffs.

System Load: Under heavy I/O, decompression peaked at 5.68s, revealing hardware-bound limitations. While the strategy optimizes speed and storage, extreme loads require further tuning for stability.

Overall, the evaluation confirms the strategy's effectiveness in addressing traditional backup flaws—delivering security, speed, and scalability. Future work will refine high-load handling and streamline encryption/compression workflows.

5. Conclusion

This paper outlines a database backup solution using MinIO that addresses key limitations in traditional approaches, particularly long backup times, high storage costs, and cross-region data mismatches. By combining LZ4 compression with AES-256-GCM encryption, the framework reduces storage needs by 73.5% while strengthening security. MinIO's Webhook integration enables automatic synchronization between local and remote backups, maintaining data consistency during network disruptions.

Performance tests on i5-11400/8GB hardware validated the system's effectiveness:

Compression Performance: LZ4 shrunk a 188MB SQL Server backup by 73.5%, with compression taking 0.5066-1.9711 seconds and decompression requiring 1.2517-5.6835 seconds. These metrics confirm the algorithm's value for storage optimization and transfer acceleration.

Encryption Efficiency: Encrypting compressed files proved four times faster than processing raw data (0.42s vs 1.66s average), demonstrating the benefit of compression-first workflows.

Transfer Capability: MinIO maintained stable upload speeds above 80MB/s and download rates exceeding 380MB/s, confirming its reliability for repeated data transfers.

System Limitations: Peak decompression times reached 5.68 seconds under heavy loads, revealing hardware constraints. While the solution optimizes performance, extreme workloads require additional tuning.

The framework successfully improves upon conventional backup methods through enhanced security, efficiency, and scalability. However, performance degrades when processing more than 1,000 concurrent requests, indicating areas for optimization. Future development will focus on three areas: optimizing encryption processes, enhancing parallel task handling, and creating adaptive management rules for large installations. These improvements aim to strengthen workflow efficiency and system resilience across different operational scenarios.

Acknowledgement

This research was supported by Chongqing Youth Vocational & Technical College and its Information Center, which provided the backup project. I am grateful for the guidance and support from my colleagues, friends, and family. Their valuable suggestions and feedback have significantly contributed to the improvement of this paper. I would also like to thank the reviewers for their insightful comments and suggestions, which helped enhance the quality of this work.

References

Bartik, M., Ubik, S., & Kubalik, P. (2015). LZ4 compression algorithm on FPGA. *2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. IEEE.

Sung, B. Y., Kim, K. B., & Shin, K. W. (2018). *An aes-gcm authenticated encryption crypto-core for iot security*.

Vijayachandran, V., & Suchithra, R. (2024). IoT Data Security by Combining Cryptography and Local Differential Privacy. *International Conference on, Innovations in Cybersecurity and Data Science Proceedings of ICICDS*. Springer, Singapore.

Yin Jiating, & Lu Tingting. (2024). Research on Integration and Development Methods of MinIO Object Storage Technology. *Journal of Wuhu Vocational and Technical College*, *26*(3), 36-39. (in Chinese)

Zhao, X., Bu, Y., Pang, W., & Cai, J. (2024). Periodic and random incremental backup policies in reliability theory. *Software Quality Journal, 32*(3), 1325-1340.