European Journal of Engineering and Technologies

Vol. 1 No. 1 2025

Article Open Access

Application of Database Performance Optimization Technology in Large-Scale AI Infrastructure

Zhongqi Zhu [1,*]

[1] Tandon School of Engineering, New York University, 6 MetroTech Center, Brooklyn, NY, 11201, USA
[*] Correspondence: Zhongqi Zhu, Tandon School of Engineering, New York University, 6 MetroTech Center, Brooklyn, NY, 11201, USA

Received: 06 July 2025
Revised: 12 July 2025
Accepted: 30 July 2025
Published: 04 August 2025

Abstract: Large-scale AI infrastructure presents significant challenges to database systems, particularly in managing high concurrency, minimizing response latency, and ensuring high availability. This article focuses on addressing three critical performance bottlenecks: query efficiency, storage I/O throughput, and concurrency control mechanisms. To tackle these challenges, we propose a comprehensive suite of performance acceleration techniques, including structural reconstruction of database schemas, hierarchical layering of hot and cold data to optimize access patterns, and advanced transaction scheduling strategies to reduce conflicts and improve throughput. These optimization methods are rigorously validated through application in representative AI scenarios such as large-scale model training and real-time online inference services. Experimental results demonstrate that the integrated optimization framework significantly enhances database performance, providing more robust and scalable data support for complex AI workloads, ultimately enabling more efficient and reliable AI infrastructure operations.

Keywords: database performance optimization; AI infrastructure; query acceleration

1. Introduction

With the rapid advancement and widespread adoption of artificial intelligence (AI) across diverse industries, the demand for robust and high-performance large-scale AI infrastructure has grown exponentially. At the core of this infrastructure lies the database system, which serves as a fundamental backbone supporting critical AI operations such as model training, real-time inference, and comprehensive data management. The performance of these database systems directly influences the efficiency and scalability of AI workloads. However, traditional database architectures face significant challenges and inherent bottlenecks when confronted with the unique demands of AI applications, including low query latency requirements, intense storage input/output (I/O) pressure, and complex concurrency control under conditions of massive, heterogeneous, and frequently interacting datasets. To effectively meet these challenges, it is imperative to develop database optimization strategies that offer both strong generalizability and scalability tailored to the specific characteristics of AI systems. This study systematically examines database performance issues from three critical perspectives—query optimization, storage structure reconstruction, and transaction scheduling—and presents a set of practical, scalable solutions designed to enhance the operational efficiency, responsiveness, and reliability of databases within large-scale AI environments.

2. Characteristics of Database Applications in Large-Scale AI Infrastructure

As the central hub for data flow and task scheduling, the database system plays a critical role across multiple stages of AI workflows, including data collection, preprocessing, model development and training, online inference, and feedback integration. AI workloads typically involve extremely high-frequency data read and write operations, particularly during the modeling and training phases, where large volumes of feature data must be accessed efficiently for batch processing and large-scale computation. In real-time inference services, databases are required to deliver ultra-low latency and support high levels of concurrent access to meet stringent performance demands. Moreover, AI systems aggregate data from a variety of heterogeneous sources, including structured log files, unstructured image data, and time-series sensor streams, which significantly increases the complexity of data integration and necessitates highly flexible and scalable database architectures. Concurrently, the widespread adoption of containerization and microservices has driven database systems toward distributed, elastic, and cloud-native deployments. Consequently, beyond traditional data storage, modern databases must offer robust support for high concurrency, horizontal scalability, and advanced data analytics capabilities to effectively handle the diverse, dynamic, and complex workloads characteristic of large-scale AI infrastructures [1].

3. Analysis of Database Performance Bottlenecks

3.1. Query Performance Bottlenecks: Slow Queries, Index Failures, High Join Overhead

In large-scale AI infrastructure, database query frequency is high and query performance bottlenecks are obvious. One common problem is slow query speed in model training and online inference, which can result in significant response time delays due to complex SQL statements or large amounts of data. Another major issue is index failure, often caused by field type mismatches, nested functions, or uneven data distribution. These factors can force the database to perform full table scans, significantly increasing system load. Thirdly, in AI application environments, it is often necessary to perform multi table JOIN to integrate feature data. If the join order is poorly planned and indexing is inadequate, the consumption of computational resources and memory will increase significantly (Table 1).

Table 1. Analysis of Types and Causes of Query Performance Bottlenecks.

Serial Number	question type	Main performance	Cause analysis	Typical impact
1	slow query	Long query response time and execution lag	Large amount of data, complex SQL logic, and index misses	System delay increases, blocking other operations
2	Index invalid	Query not indexed, resulting in full table scan	Type mismatch, nested functions, fuzzy matching, skewed data distribution	Low query efficiency, increased CPU and IO resource usage
3	JOIN has high overhead	Multiple table associations cause memory overflow or response timeout	Related fields not indexed, imbalanced data volume, connection order and algorithm not optimized	The execution efficiency of the query plan is low, and the system throughput capacity is reduced

3.2. Storage and IO Bottleneck: High Disk Load, Failure to Separate Hot and Cold Data

In large-scale AI systems, storage and I/O performance often become critical bottlenecks that directly impact the responsiveness, scalability, and stability of the entire platform. The first and most prominent challenge is high disk load. As AI platforms continuously ingest and process massive volumes of data—including training datasets, system logs, feature stores, and intermediate computation results—disk I/O frequently operates under high saturation. This persistent strain can lead to elevated read/write latency, increased data access contention, and even system stalls under peak workloads.

The second bottleneck involves the failure to separate hot and cold data effectively. AI systems typically maintain a mix of real-time operational data—such as user interaction logs or updated model parameters—and large repositories of historical or infrequently accessed data. Without a hierarchical or tiered storage strategy to distinguish between high-frequency (hot) and low-frequency (cold) data, frequently accessed datasets are forced to compete for limited I/O bandwidth alongside archival data [2]. This not only lowers system throughput but also diminishes the efficiency of caching mechanisms and increases memory pressure.

The third issue stems from suboptimal storage architecture, particularly when dealing with heterogeneous or unstructured data types. Unstructured data such as images, videos, and sensor signals are often stored alongside structured metadata in a monolithic design without proper categorization, indexing, or isolation. This lack of logical and physical separation not only degrades access patterns but also intensifies I/O contention due to fragmented storage layouts and inefficient retrieval paths.

Together, these factors severely constrain the scalability of AI infrastructure, making it difficult to meet the latency-sensitive demands of training, inference, and real-time analytics tasks. Addressing these bottlenecks requires a comprehensive redesign of the data storage strategy, incorporating intelligent data placement, multi-tier caching, and adaptive I/O scheduling to better align with the workload characteristics of AI systems (Table 2).

Table 2. Analysis of Storage and IO Bottleneck Issues.

Serial Number	question type	Main performance	Cause analysis	Typical impact
1	Disk load too high	IO waiting time is long, and disk utilization continues to approach 100%	Frequent batch read and write operations, centralized large file operations, and no IO scheduling optimization mechanism	Delay in query response and decrease in system throughput capacity
2	Cold and hot data not separated	High frequency access with low data access efficiency	All data is stored uniformly, lacking access frequency recognition and data layering mechanism	High frequency data access is blocked by low-frequency data, resulting in system instability
3	Slow processing of unstructured data	Slow response to reading logs, images, and model files	No dedicated storage strategy, incompatible with traditional row storage structure	Improved access latency and reduced data processing efficiency

3.3. Concurrent Access Bottlenecks: Lock Competition, Transaction Conflict, Connection Pool Exhaustion

In the high concurrency environment of large-scale AI tasks, database systems will encounter serious concurrency access bottlenecks. Firstly, lock competition is severe.

When multiple training and inference processes need to use the same data, they will face frequent row level and table level lock conflicts, which can cause blockages and lags, resulting in serious impacts. Secondly, transaction conflicts, especially in data write intensive scenarios, are evident. The backlog or conflicts of unsubmitted transactions can lead to frequent rollback, affecting database consistency and stability. Thirdly, there is a backlog of connection pools. Connection pool exhaustion is a common problem under high-concurrency conditions. In microservice architectures, excessive short-lived connection requests can quickly saturate the pool, impairing the system's ability to respond to new requests. These overlapping issues directly limit the AI infrastructure services provided by the database (Table 3).

Table 3. Analysis of Types and Causes of Concurrent Access Bottlenecks.

Serial Number	question type	Main performance	Cause analysis	Typical impact
1	lock contention	Query or write request blocking, slow transaction execution	Multiple concurrent transactions accessing the same resource without proper use of lock granularity or scheduling strategies	System latency increases and resource utilization efficiency decreases
2	Transaction Conflict	Submission failure, frequent rollback, data consistency risk	Frequent write operation conflicts and improper transaction isolation level settings	Increasing processing burden and affecting data reliability
3	Connection pool exhausted	New connection request rejected, system response failed	The concurrent access count exceeds the capacity of the connection pool, and the connection is not released in a timely manner	Request backlog, service unavailable

4. Database Performance Optimization Technology Strategy

4.1. Structured Optimization and Query Acceleration Techniques to Improve Query Efficiency

To address common query performance bottlenecks in AI platforms, database response efficiency can be enhanced through structural optimization and query acceleration techniques. Firstly, optimize the index by building joint indexes, overlay indexes, inverted indexes, etc., to reduce the data reading range. Secondly, by optimizing the execution of the query plan strategy, dynamically adjusting the order of connections, filtering positions, and execution methods, we can reduce the resource consumption of JOIN operations for connections [3]. Thirdly, with the help of materialized views and result caching mechanisms, results can be calculated and cached in advance in AI tasks with stable query scenarios, avoiding further processing. For high concurrency queries, partition table design and parallel query strategy can also be used to evenly distribute the query load of large tables to various nodes during a large number of accesses. The combination of these technologies is beneficial in avoiding the system pressure caused by slow and complex queries, improving data extraction speed and system throughput.

4.2. Optimizing Storage Structure and Hierarchical Management Strategy for Cold and Hot Data

To alleviate the storage and IO bottlenecks in AI systems, it is necessary to construct a reasonable storage structure and implement a hierarchical management strategy for hot and cold data. The access frequency f (i) can be used to classify data objects, where:

$$f(i)=\frac{n_i}{T} \quad (1)$$

Among them, n_i represents the number of times data object i is accessed within time window T. If f (i)>θ, the object is classified as hot data; otherwise, it is considered cold data, where θ represents the access frequency threshold.

In practical systems, hot data should be prioritized for storage in SSD or memory cache layers to improve read and write speeds; Cold data is stored in HDD or archiving systems to reduce costs. Further, a layered storage model can be adopted:

$$S=\{L_1，L_2，\ldots，L_k\} \quad (2)$$

Among them, S is the storage system, and Lk represents the data storage area of the k-th layer (such as cache, main memory, disk). By employing dynamic migration mechanisms and access pattern analysis, data can be intelligently moved between layers, reducing disk pressure and enhancing overall I/O performance and resource utilization.

4.3. Strengthen the Transaction Optimization Mechanism for Concurrency Control and Resource Scheduling

To address transaction conflicts and resource contention in high-concurrency AI environments, it is essential to enhance database concurrency control and resource allocation mechanisms. Firstly, through the Multi Version Concurrent Control (MVCC) mechanism, version snapshots are used to achieve read-write separation, alleviate read-write lock conflicts, and improve transaction concurrency performance [4]. Secondly, for a large number of write operations, a lightweight lock mode is adopted, combined with row level locks and intent locks to control granularity and reduce performance consumption caused by lock competition. Thirdly, asynchronous submission and batch transaction processing can help merge small transactions, reduce the number of I/O operations, and improve write throughput. For resource scheduling, connection pool elastic adjustment or priority scheduling can be used to prevent situations where the connection pool is exhausted or resources are insufficient. The combination of the above methods can effectively alleviate data write conflicts, lock waiting, and system crashes on big data platforms, ensuring stable database operations and rapid response on big data platforms.

5. Application Practice of Optimization Technology in Typical Scenarios of Large-Scale AI Infrastructure

5.1. Data Preprocessing Acceleration in AI Model Training Platform

Data preprocessing is a crucial step in the training process of AI models, often consuming a significant amount of time and resources, and having a decisive impact on the entire training process. For large-scale datasets, traditional database queries cannot meet the requirements of high concurrency and low latency [5]. Partitioned table design enables task-level parallel data access, significantly increasing data throughput; The use of materialized views and caching techniques can reduce redundant calculation processes and lower the time required for queries; Using a columnar storage format improves field-level access efficiency and is particularly suitable for feature filtering operations; In addition, heterogeneous data preloading techniques enable the efficient preparation of multi-source data prior to merging, enhancing the stability and responsiveness of the training platform (Table 4).

Table 4. Analysis of Data Preprocessing Acceleration Strategies for AI Model Training Platform.

Serial Number	optimization strategy	Application methods and characteristics	Performance improvement performance
1	Partition table structure design	Divide by time, task, or label dimension to achieve parallel queries	Read efficiency increased by 20%~50%

2	Materialized views and intermediate result caching	Cache duplicate query results to avoid redundant calculations	Significant reduction in query latency and load reduction
3	Column based storage structure optimization	Only load required fields to reduce irrelevant data access	IO overhead decreases, field reading speed increases by more than 2 times
4	Heterogeneous Data Mapping and Preloading Strategy	Preprocess multi-source data into a unified structure and load hotspot fields in advance	Reduced data fusion time and more stable platform response

5.2. High Concurrency Query Optimization in Online Inference Services

In AI online inference services, model calls and result delivery must be completed within milliseconds. This requires robust high-concurrency query capabilities from the database system [6]. Firstly, the application of read-write separation architecture directs real-time inference requirements to read-only replicas, relieving the pressure on the main library; Secondly, multi-level caching mechanisms—such as using Redis to cache model features, user profiles, and precomputed results—can significantly reduce database access frequency; Thirdly, the application of dynamic expansion and reuse technology in connection pools can avoid connection exhaustion and improve request processing capabilities. Additionally, delay-tolerant algorithms and prefetching strategies allow high-frequency data to be loaded in advance, further reducing response latency. For hot data, partitioning and local indexing can also be used to reduce query pressure. By integrating these methods, inference services can ensure stability and real-time performance in high concurrency situations, improving overall user experience and system availability (Figure 1).

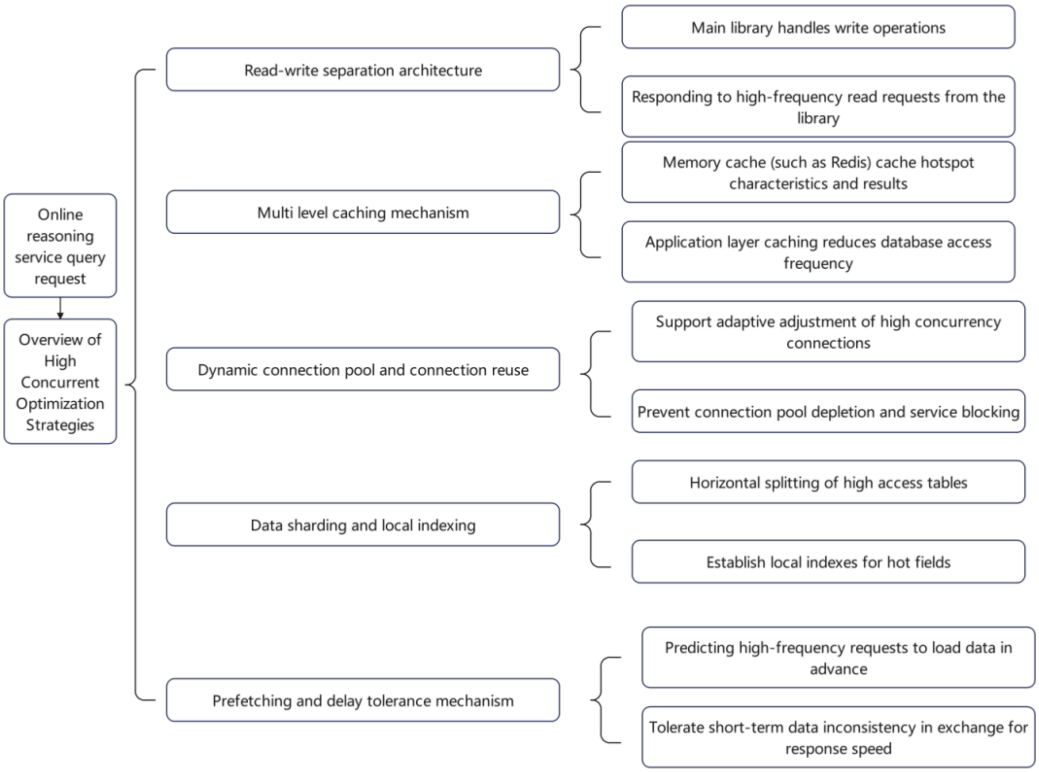
Online reasoning service query request
Overview of High Concurrent Optimization Strategies
Read-write separation architecture
Main library handles write operations
Responding to high-frequency read requests from the library
Multi level caching mechanism
Memory cache (such as Redis) cache hotspot characteristics and results
Application layer caching reduces database access frequency
Dynamic connection pool and connection reuse
Support adaptive adjustment of high concurrency connections
Prevent connection pool depletion and service blocking
Data sharding and local indexing
Horizontal splitting of high access tables
Establish local indexes for hot fields
Prefetching and delay tolerance mechanism
Predicting high-frequency requests to load data in advance
Tolerate short-term data inconsistency in exchange for response speed

Figure 1. Framework diagram of high concurrency query optimization strategy in online inference services.

5.3. Batch Write Optimization for Log and Behavioral Data Collection

In AI infrastructure, the frequency of collecting user behavior data and system logs is extremely high, and the write pressure far exceeds the read demand. Without proper management, this may lead to write congestion, disk I/O overload, and potential data loss. Therefore, a batch write strategy is needed to merge small amounts of data into large transaction submissions, reducing write frequency and IO costs. Asynchronous write strategy can also decouple the collection end from the database pressure. Incoming data is first directed to a cache layer before being committed to the database, improving system responsiveness and reducing write-blocking issues. At the same time, using pre write logging (WAL) and streaming write channel technology can ensure the atomicity and traceability of data writing. In response to the rapid growth of logs, a combination of time partitioning and archiving strategies can be used to automatically archive historical logs to cold storage and free up the primary tablespace [7]. This approach alleviates real-time write load, enhances data durability, and ensures reliable and efficient behavioral data management for AI platforms (Table 5).

Table 5. Analysis of Batch Writing Optimization Strategies for Log and Behavioral Data.

Serial Number	optimization strategy	Technical Key Points Explanation	Performance improvement and advantages
1	Batch write mechanism	Merge multiple records into one transaction submission to reduce the number of write operations	Reduce IO frequency and significantly improve write throughput
2	Asynchronous write strategy	Decouple the data writing process from the main process and use a buffer to cache data	Improve response speed and avoid blocking the main business process
3	Pre writing logs and streaming storage	Ensure the integrity and consistency of the data writing process	Improved data reliability for easy recovery from anomalies
4	Partition and archiving strategy	Partition storage by time or type, automatically archive historical data	Reduce the pressure on the main gauge and improve query and write efficiency

5.4. Model Version Management and Metadata Query Efficiency Improvement

As AI systems continue to evolve, the number of models per version increases significantly, making it more challenging to manage associated metadata such as training parameters, data sources, and evaluation metrics. Traditional databases often suffer from latency fluctuations and index failures when processing small-sized, high-frequency queries. To address this, graph-based data modeling is employed to manage model dependencies and simplify multi-level reference query paths; Additionally, constructing a dedicated metadata index and tagging system enables efficient filtering by timeline, model type, and task scope. By integrating caching mechanisms and partitioned storage strategies, query loads are reduced and model retrieval and scheduling efficiency is enhanced. The optimized metadata query mechanism significantly improves the traceability, auditability, and maintainability of the AI platform.

6. Conclusion

With the continuous expansion of AI infrastructure and the increasing complexity of tasks, database systems play a crucial role in supporting model training, inference services, and data flow. However, increasing database bottlenecks—stemming from high concurrency, storage overload, and transactional conflicts—necessitate systematic optimization strategies. This article focuses on the core issues of query acceleration, hierarchical data storage, and transaction scheduling in databases. A series of technical strategies were proposed and validated through real-world AI application scenarios. The results demonstrate that targeted optimization measures significantly improve database performance and contribute to greater system stability and scalability. Future database optimization technologies will become more intelligent and adaptive, offering stronger support for building efficient and resilient AI infrastructures.

References

1. S. J. Kamatkar et al., "Database performance tuning and query optimization," in *Int. Conf. Data Mining Big Data*, Cham: Springer Int. Publishing, 2018, pp. 1, doi: 10.1007/978-3-319-93803-5_1.
2. S. Huang et al., "Survey on performance optimization for database systems," *Sci. China Inf. Sci.*, vol. 66, no. 2, p. 121102, 2023, doi: 10.1007/s11432-021-3578-6.
3. S. Yang, “The Impact of Continuous Integration and Continuous Delivery on Software Development Efficiency”, *J. Comput. Signal Syst. Res.*, vol. 2, no. 3, pp. 59–68, Apr. 2025, doi: 10.71222/pzvfqm21.
4. H. Ledford, "Social scientists battle bots to glean insights online," *Nature*, vol. 578, p. 6, 2020. doi: 10.1038/d41586-020-00141-1.
5. A. Kaun and M. Männiste, "Public sector chatbots: AI frictions and data infrastructures at the interface of the digital welfare state," *New Media Soc.*, vol. 27, no. 4, pp. 1962–1985, 2025, doi: 10.1177/14614448251314394.
6. F. Gao, “The Role of Data Analytics in Enhancing Digital Platform User Engagement and Retention”, *J. Media Journal. Commun. Stud.*, vol. 1, no. 1, pp. 10–17, Apr. 2025, doi: 10.71222/z27xzp64.
7. J. Wang et al., "An optimized RDMA QP communication mechanism for hyperscale AI infrastructure," *Cluster Comput.*, vol. 28, no. 1, pp. 66, 2025, doi: 10.1007/s10586-024-04796-7.

Disclaimer/Publisher’s Note: The views, opinions, and data expressed in all publications are solely those of the individual author(s) and contributor(s) and do not necessarily reflect the views of PAP and/or the editor(s). PAP and/or the editor(s) disclaim any responsibility for any injury to individuals or damage to property arising from the ideas, methods, instructions, or products mentioned in the content.