Scholar Publishing Group
International Journal of Big Data Intelligent Technology
https://doi.org/10.38007/IJBDIT.2025.060215
ISSN 2790-0932 Vol. 6, Issue 2: 139-146

The Impact of Distributed Data Query Optimization on Large-Scale Data Processing

Jin Li

Morgan Stanley, 65 Irby Ave NW, Atlanta, GA, 30305, US

Keywords: Distributed Data Query, Data Optimization, Large Scale Data Processing, Performance Improvement, System Bottleneck

Abstract: With the rapid advancement of big data technology, distributed architecture has become the mainstream in the industry when processing massive amounts of information. However, when dealing with such large datasets, the query efficiency and performance of the system become key factors that constrain its response speed and accuracy. This study analyzed the key performance bottlenecks of distributed data queries, such as storage response latency, hardware processing capacity limits, and data consistency assurance. Based on this, a series of targeted improvement measures were proposed. Specifically, in terms of distributed storage network latency, hardware resource upgrade requirements, and data consistency maintenance, research has proposed solutions such as optimizing data distribution, regularly upgrading hardware facilities, and adopting distributed locking strategies. After implementing these optimization measures, query response can be accelerated, data accuracy can be ensured, and hardware costs and maintenance expenses can be reduced. The research results show that these optimization methods can enhance the overall performance of processing large-scale data systems.

Introduction

In the era of information explosion, the rapid increase in data volume makes it difficult for traditional independent data processing facilities to meet the demand for high-speed and accurate data queries. Distributed architecture has become the preferred solution for dealing with massive data processing due to its scalability and powerful data processing efficiency. With the continuous deepening application of distributed data systems, how to improve query efficiency and overcome obstacles in data processing has become an urgent challenge. The obstacles to query efficiency mainly stem from the slow response of storage systems, bottlenecks in hardware processing capabilities, and difficulties in ensuring data consistency in distributed environments, which directly affect the timeliness of queries and the accuracy of results. This article will explore these issues and propose a series of optimization strategies to improve the overall performance of large-scale data processing systems.

1. Query bottleneck in large-scale data processing

1.1 Delay of Distributed Storage Systems

In distributed architectures that handle massive amounts of information, the response time of storage nodes becomes a core obstacle that constrains performance. This type of storage system typically distributes information across numerous servers in order to enhance the scalability and stability of data queries.However, data calls between different nodes can cause network latency issues, especially when dealing with large amounts of data requests, where this latency is more prominent. If a query operation requires the mobilization of information from multiple servers, each remote data acquisition is accompanied by network transmission delay, which prolongs the feedback time of the entire query. In addition, distributed storage systems also need to address the challenges of duplicate data storage and replica synchronization. Although redundant backup of data enhances the system's error resistance, the process of data replication and synchronization also brings additional latency. The response time of storage nodes can reduce the efficiency of data reading and may also lead to fluctuations in query results, especially when the system is under high load, this problem becomes more prominent.

1.2 System throughput is limited by hardware during queries

In distributed data queries, system throughput is one of the key indicators to measure its performance. The strength of throughput directly determines the agility of system response and the ability to handle multiple tasks. However, in many distributed information processing frameworks, hardware infrastructure often becomes a limiting factor that hinders further improvement of system throughput capacity.Especially based on traditional hardware structures, it is difficult to meet the efficient and concurrent query requirements of massive data in terms of storage and computing node performance. The limitations of hardware facilities such as hard disk input/output efficiency, central processing unit computing power, and network transmission bandwidth are the main reasons for the limited query processing capabilities. Even though distributed storage has been implemented for data, data interaction between nodes is still constrained by hardware conditions, making it impossible to fully unleash system performance. When the system load reaches its peak, insufficient hardware resources may lead to server overload and slow down the system's response speed. In some scenarios where a large number of concurrent query requests need to be processed, hardware resource limitations may lead to a backlog of query tasks and delayed responses, affecting the overall operational efficiency of the system.

1.3 Inconsistencies in Data in Distributed Systems

In a distributed architecture, maintaining data consistency has become an important bottleneck due to information being segmented and stored on numerous server nodes. Data consistency refers to the ability of each server node's data to remain synchronized throughout the entire system, ensuring that the data obtained by users is accurate and updated. However, in a distributed environment, ensuring data consistency becomes extremely difficult due to factors such as network transmission delays, server failures, or asynchronous data replication. In high concurrency data reading scenarios, different server nodes may access different versions of data, leading to inconsistencies between query results. This type of data inconsistency phenomenon is particularly sensitive in business scenarios that require high precision, such as financial data analysis and real-time monitoring systems. Although many distributed systems adopt the "ultimate consistency" principle in CAP theory to enhance system availability and fault tolerance, "ultimate consistency" is not equivalent to real-time consistency and may still result in discrepancies between query results and actual data. In order to overcome the problem of inconsistent data, the system needs to adopt complex coordination mechanisms, such as distributed locking and transaction control. The introduction of these mechanisms can alleviate the problem, but it also increases query latency and system complexity, which has an impact on performance.

2. Optimization strategy for distributed data query

2.1 Optimize data distribution to reduce network latency impact

The core purpose of optimizing data distribution is to reduce the frequency of data interaction between nodes through scientific segmentation and reasonable arrangement of data. The commonly adopted methods include data segmentation and data location optimization processing. Data segmentation generally divides a dataset into numerous parts based on specific attributes (such as user ID, timestamp, etc.), and then stores these parts independently on different servers. By segmenting and storing data on geographically adjacent servers, it is possible to reduce remote data calls for query requests and prevent unnecessary data traffic consumption. Determining appropriate segmentation keywords and segmentation schemes is a crucial step in practice.

Taking the transaction query function in the banking industry as an example, when a customer initiates a query, their historical transaction data may be scattered across numerous servers. If all transaction records of the same customer are stored in a single server, only that server can be involved in the query, thereby avoiding cross server data exchange and reducing query response time. A distributed system consists of N nodes, each storing S data items, and each query request requires access to data from D nodes. If the optimized distribution of data can reduce cross node access, the total latency T can be expressed as:

$$T = (N - k) \times L \times D \tag{1}$$

In formula (1), k represents the number of cross node accesses reduced by optimizing data distribution, L is the network transmission delay, and D is the amount of data queried each time. By optimizing data distribution and reducing unnecessary cross node access (i.e. reducing k), the system's query latency can be reduced and query efficiency can be improved.

2.2 Regularly updating server hardware to maintain high throughput

The update of computing resources has a direct impact on system throughput. With the rapid development of processor technology, most current servers are equipped with efficient multi-core processors, which can process a large amount of computing work in a shorter time and improve the parallel processing level of the system.The size of memory capacity is equally critical, as it can reduce latency caused by frequent data exchange and alleviate disk I/O pressure, ensuring the speed of data processing and querying. In addition, with the innovation of storage technology, the application of solid-state drives (SSDs) has gradually replaced traditional mechanical hard drives (HDDs), accelerating the speed of data reading. When dealing with a large number of read requests, the performance advantages of SSDs are more prominent. The increase in network bandwidth is also key to maintaining high system throughput, especially when processing massive amounts of data, which requires strong local computing power and data transmission between different nodes. The increase in network bandwidth enables faster data transfer between nodes, reduces query waiting time, and improves overall system throughput efficiency.

To verify the positive impact of regular hardware updates on system processing capabilities, we can measure their effectiveness through data analysis. The following data table 1 is derived from performance testing of a large-scale e-commerce website, with the aim of comparing the effect of hardware upgrades on improving system processing efficiency before and after. This data reflects the specific changes in system processing capabilities after performing regular hardware update operations.

Table 1. Changes in system throughput after regular hardware updates

Cycle time	Processor type	Memory capacity (GB)	Storage type	Network bandwidth (Gbps)	Throughput (queries per second)
Initial state	Dual core CPU	16	HDD	1	500
First year update	Four core CPU	32	SSD	2	800
Second year update	Eight core CPU	64	SSD	4	1200
Third year update	Sixteen core CPU	128	NVMe SSD	10	1800

Observing the data shown in Table 1, as hardware iteration upgrades are carried out year by year, there is a growing trend in the number of processor cores, memory space, storage medium types, and network transmission bandwidth, which promotes the annual increase in system throughput. Regular hardware upgrades enhance the processing performance and data transmission efficiency of the system, ensuring that even under the enormous pressure of large-scale data queries, the system can maintain high data processing throughput and fast response time.

2.3 Introduction of Distributed Lock Mechanism

Ensuring data synchronization and concurrency management is the core when conducting data queries in distributed systems. Especially in situations with high concurrency, many query operations may read and write the same data resource concurrently, resulting in data inconsistency issues. Adopting a distributed locking mechanism is a key way to improve query efficiency and maintain data consistency in order to prevent data errors and ensure the accuracy of query results. Distributed lock is a synchronization control mechanism that spans multiple nodes, ensuring that only one node can access specific data resources at any given time, avoiding data conflicts. In traditional distributed architectures, ordinary locking mechanisms are not applicable due to the relative independence of each node. Distributed locks manage and allocate locks through coordinated services such as Zookeeper, etcd, Redis, etc., ensuring effective coordination of resources between nodes and implementation of access control.

On an e-commerce platform, when many consumers try to place orders for a popular product at the same time, there is a high possibility of oversold inventory. After adopting the distributed lock mechanism, any request must first obtain lock qualification to ensure that only one server node can make changes to inventory information at any given moment, and the remaining nodes need to queue and wait. This mechanism has N nodes, and the application and release actions of locks are uniformly regulated by a coordination service (such as Zookeeper). Whenever a node needs to access a common resource, it will actively attempt to apply for a lock. If the lock is already occupied by another node, the current node must wait. The time for each node to obtain a lock is Tlock, and in high concurrency, lock competition may result in an average waiting time Twait. The total delay Ttotal can be expressed as:

$$T_{total} = T_{lock} + T_{wait} = T_{lock} + \frac{N-1}{N} \times T_{lock} \tag{2}$$

In formula (2), $\frac{N-1}{N}$ the probability of each node waiting for a lock is represented, and Tlock is the time required to obtain the lock. After adopting distributed locking technology, although the processing time of individual nodes slightly increases, it ensures data consistency, effectively prevents conflicts in concurrent operations, and enhances the overall robustness of the system. In the process of distributed data processing, the use of distributed locks can restrict concurrent access, maintain data consistency, perform efficient concurrency control for systems in high load environments, improve query efficiency, and ensure stable system operation. This mechanism ensures the consistency of inventory data and avoids issues such as duplicate charges or inaccurate inventory caused by concurrent operations.

3. Optimize the impact on large-scale data processing

3.1 Significant reduction in query response time

The core of improving the operational efficiency of distributed data query systems lies in reducing query response time, which is one of the most important performance metrics in massive data processing frameworks. The reduction of query response time requires comprehensive consideration from multiple dimensions, such as rationalization of data layout, enhancement of hardware facilities, reduction of network transmission latency, and optimization of concurrent processing. After adopting these improvement measures, the system can maintain efficient response and reduce query waiting time when dealing with numerous query demands.

In the conventional distributed system data query process, due to issues such as imbalanced data allocation, limited device resources, and network bandwidth bottlenecks, data requests need to cross multiple nodes, causing delays in data transmission and processing and resulting in extended response times. By adjusting the data layout strategy, reducing the frequency of data calls between nodes, enhancing device performance, and reducing network transmission latency, the system response speed can be accelerated. When users on a certain e-commerce platform query product information, the system needs to quickly provide accurate results. If data distribution is improper or device resources are scarce, user requests may encounter significant delays that affect the user experience. Through reasonable planning of data slicing, regular hardware upgrades, and optimization of query execution schemes, the query response time has been successfully reduced from 2 seconds to 1 second, and the query processing capability has been improved by 50%. The following is a quantitative analysis of the optimization effectiveness, showing the comparison data of query response time before and after optimization.

Observing the data in Table 2, it can be observed that the optimized distributed data query system has shortened the response time of queries and improved the overall performance of the system. Especially in environments with large amounts of data and frequent concurrent requests, optimization results are particularly outstanding. With these improvements, the system can efficiently handle numerous concurrent queries, enhancing the overall user experience.

Table 2. Query response time before and after optimization

Optimization measures	Query response time (seconds)	Query throughput (number of queries per second)	Number of data access nodes	System load
Initial state	2.0	200	5	high
Data distribution optimization	1.5	300	4	in
Hardware upgrade (CPU, memory)	1.2	350	3	low
Fully optimized (distributed locking, network enhancement)	1.0	400	2	low

3.2 Enhancement of Data Accuracy Assurance

In a wide range of distributed data query scenarios, data accuracy is the key to ensuring system stability and reliability. With the increase in data size and query frequency, there is a challenge in maintaining data consistency and accuracy under highly concurrent conditions. Especially when multiple nodes are processing data simultaneously, data inconsistency and conflicts are highly likely to have adverse effects on the system.

Distributed architectures that address this issue often rely on data consistency protocols and transaction processing strategies to solve it. For example, strong consistency protocols such as the two-stage commit protocol and Paxos protocol ensure synchronization among nodes when updating data, maintaining consistency between data replicas. The distributed transaction control method ensures that data changes are either completely completed or completely revoked to prevent data inconsistency. Taking inventory management on online shopping platforms as an example, in the face of multiple users simultaneously purchasing a product, distributed lock technology plays a key role in avoiding simultaneous modification of the same data by different nodes. The use of a lock mechanism system can ensure that only one node can make changes to inventory data at any given time, preventing issues such as overselling. With these optimization strategies, the system can avoid conflicts and inconsistencies in data updates, ensure the accuracy of data queries, and maintain system stability and efficiency in high load environments.

3.3 Reduction in hardware resource investment

In widely distributed data processing architectures, the investment scale of hardware facilities directly affects the economic investment and operational efficiency of the system. In the face of processing large amounts of data, the conventional approach is to expand processing units, memory space, and storage resources. However, by implementing optimization methods for data retrieval, it is possible to effectively reduce resource requirements and improve system efficiency without increasing hardware costs.

By making reasonable adjustments to data layout, reducing information exchange between nodes, and enhancing query processing speed, the system has reduced its dependence on devices. Appropriate data segmentation and localization processing can ensure that most query tasks are completed within the corresponding nodes, reduce data exchange between nodes, and alleviate the pressure on network bandwidth and storage resources. In addition, distributed caching strategies can temporarily store frequently accessed data in memory, reducing the occupation of hard disk storage and lowering the demand for storage devices.

4. Conclusion

After deeply analyzing the performance bottlenecks in the distributed retrieval process, this study proposes a series of targeted improvement measures and further analyzes the actual effectiveness of these measures in processing massive amounts of data. Research has found that by improving data layout, periodically upgrading hardware facilities, and adopting distributed locking technology, retrieval latency can be shortened, data consistency can be improved, and hardware investment and operation costs can be reduced. These improvement measures have enhanced the retrieval capability of the system, providing a more reliable and durable technical guarantee for big data management. In the future, with the continuous advancement of technology, the optimization of distributed data retrieval will evolve towards higher efficiency and intelligence, helping large-scale data processing systems to be deeply applied in many fields. The results of this study have important practical significance for optimizing the performance of large-scale data processing systems and reducing their operational burden.

References:

[1] *Deepthi B. Gnana, et al. "An efficient architecture for processing real-time traffic data streams using apache flink."Multimedia Tools and Applications 83.13(2023):37369-37385.*

[2] *Trinh Thanh, et al. "A novel ensemble-based paradigm to process large-scale data. "Multimedia Tools and Applications 83.9(2023):26663-26685.*

[3] *Lin Zihang, et al. "SciSciNet: A large-scale open data lake for the science of science research." Scientific data 10.1(2023):315-315.*

[4] *Kontou Eftychia E, et al. "UmetaFlow: an untargeted metabolomics workflow for high-throughput data processing and analysis." Journal of cheminformatics 15.1(2023):52-52.*

[5] *Su H, Luo W, Mehdad Y, et al. Llm-friendly knowledge representation for customer support[C]//Proceedings of the 31st International Conference on Computational Linguistics: Industry Track. 2025: 496-504.*

[6] *Su Jinshu, et al. "Technology trends in large-scale high-efficiency network computing." Frontiers of Information Technology & Electronic Engineering 23.12(2022):1733-1746.*

[7] *Zou, Y. (2025). Design and Implementation of a Cloud Computing Security Assessment Model Based on Hierarchical Analysis and Fuzzy Comprehensive Evaluation. arXiv preprint arXiv:2511.05049.*

[8] *Liu, B. (2025). Design and Implementation of Data Acquisition and Analysis System for Programming Debugging Process Based On VS Code Plug-In. arXiv preprint arXiv:2511.05825.*

[9] *Zhu, P. (2025). The Role and Mechanism of Deep Statistical Machine Learning In Biological Target Screening and Immune Microenvironment Regulation of Asthma. arXiv preprint arXiv:2511.05904.*

[10] *Chang, Chen-Wei. "Compiling Declarative Privacy Policies into Runtime Enforcement for Cloud and Web Infrastructure." (2025).*

[11] *F. Liu, "Transformer XL Long Range Dependency Modeling and Dynamic Growth Prediction Algorithm for E-Commerce User Behavior Sequence," 2025 2nd International Conference o`n Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, 2025, pp. 1-6, doi: 10.1109/IACIS65746.2025.11211467.*

[12] *F. Liu, "Architecture and Algorithm Optimization of Realtime User Behavior Analysis System for Ecommerce Based on Distributed Stream Computing," 2025 International Conference on Intelligent Communication Networks and Computational Techniques (ICICNCT), Bidar, India, 2025, pp. 1-8, doi: 10.1109/ICICNCT66124.2025.11232744.*

[13] Q. Hu, "Research on Dynamic Identification and Prediction Model of Tax Fraud Based on Deep Learning," 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, 2025, pp. 1-6, doi: 10.1109/IACIS65746.2025.11211426.

[14] D. Shen, "Complex Pattern Recognition and Clinical Application of Artificial Intelligence in Medical Imaging Diagnosis," 2025 International Conference on Intelligent Communication Networks and Computational Techniques (ICICNCT), Bidar, India, 2025, pp. 1-8, doi: 10.1109/ICICNCT66124.2025.11232821.

[15] X. Liu, "Research on User Preference Modeling and Dynamic Evolution Based on Multimodal Sequence Data," 2025 2nd International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), Hassan, India, 2025, pp. 1-7, doi: 10.1109/IACIS65746.2025.11211273.

[16] Ding, J. (2025). Intelligent Sensor and System Integration Optimization of Auto Drive System. International Journal of Engineering Advances, 2(3), 124-130.

[17] Mingjie Chen. (2025). Exploration of the Application of the LINDDUN Model in Privacy Protection for Electric Vehicle Users. Engineering Advances, 5(4), 160-165.

[18] Liu, X. (2025). Research on Real-Time User Feedback Acceleration Mechanism Based on Genai Chatbot. International Journal of Engineering Advances, 2(3), 109-116.

[19] Zhang, M. (2025). Research on Collaborative Development Mode of C# and Python in Medical Device Software Development. Journal of Computer, Signal, and System Research, 2(7), 25-32.

[20] Wang, Y. (2025). Intervention Research and Optimization Strategies for Neuromuscular Function Degeneration in the Context of Aging. Journal of Computer, Signal, and System Research, 2(7), 14-24.