The American Journal of Engineering and Technology
ISSN 2689-0984 Volume 08 - 2026

Architecture of a Hybrid In-Memory and Cold Storage for Historical Financial Data

[1] Andrii Humeniuk
[1] Master Degree in Software Engineering Lead Software Engineer, DASTA Incorporated ("dub") New York, USA

Received: 18th Nov 2025 | Received Revised Version: 28th Nov 2025 | Accepted: 27th Dec 2025 | Published: 16th Jan 2026

Volume 08 Issue 01 2026 | Crossref DOI: 10.37547/tajet/Volume08Issue01-11

Abstract

This paper introduces a novel hybrid storage architecture (HyFDS), designed by the author to address the dual challenge of ultra-low latency trading workloads and cost-efficient archival storage in financial markets. Unlike prior works that optimize individual components, HyFDS integrates Apache Kafka, In-Memory Data Grids, lock-free patterns, and Apache Iceberg into a unified framework, validated against the requirements of high-frequency trading. This work proposes a conceptual model of a hybrid data storage architecture (HyFDS) that addresses this problem through the synthesis of heterogeneous technological approaches. The architecture is based on an event-driven model built on Apache Kafka, which serves as a unified bus for all system events. The "hot" tier, implemented on an In-Memory Data Grid (IMDG) and optimized through the LMAX Disruptor pattern and lock-free data structures, enables transaction processing with sub-millisecond latency. The "cold" tier, based on object storage with the Apache Iceberg tabular format, ensures scalable and cost-effective storage. The study analyzes data migration mechanisms, transactional consistency strategies (2PC, Saga), and disaster recovery plans, forming an integrated framework for designing next-generation financial systems.

Keywords: hybrid storage, in-memory, cold storage, event-driven architecture, Apache Kafka, low-latency, high-frequency trading, lock-free, data consistency, LMAX Disruptor

Cite This Article: Humeniuk, A. (2026). Architecture of a Hybrid In-Memory and Cold Storage for Historical Financial Data. The American Journal of Engineering and Technology, 8(01), 78–86. https://doi.org/10.37547/tajet/Volume08Issue01-11

1. Introduction

In the context of the rapid growth of cloud technologies, the financial sector is undergoing an active migration of workloads to hybrid cloud environments: the private and public cloud market in financial services will grow by USD 106.43 billion in the period 2024–2028 (compound annual growth rate of 19%) [8].

Hybrid transactional-analytical (HTAP) systems, combining OLTP and OLAP workloads in a single solution, are classified into four main storage architecture models: row-oriented, column-oriented, separated, and hybrid [2]. In particular, Ionescu et al. propose a resilient four-layer architecture for financial institutions, including data sources, processing, integration, and multi-tier storage, taking into account energy efficiency and environmental sustainability [1, 12].

In this environment, competitive advantage is defined by speed: algorithmic trading requires latency for order matching of less than 1 millisecond. Peak loads in stock and cryptocurrency markets can generate millions of

orders, creating unprecedented pressure on IT infrastructure. At the same time, regulatory requirements and the need for deep analysis of historical data for training machine learning models lead to the accumulation of archives on the petabyte scale.

This duality of requirements creates a fundamental problem. On the one hand, systems must ensure extremely low latency and high throughput for processing "hot" data — active orders, current market quotes, and positions. On the other hand, it is necessary to store huge volumes of "cold" data — historical trades, event logs, and quotes spanning many years — in a cost-effective and reliable manner. Traditional architectures based on monolithic databases are unable to efficiently resolve this conflict. They are either too expensive for storing large archives (if optimized for speed) or too slow for HFT (if optimized for storage). This contradiction dictates the need to transition to hybrid, multi-tier storage systems that segment data according to the "temperature" of its usage.

The aim of the study is to propose and evaluate a hybrid financial data storage (HyFDS) model that bridges ultra-low latency and large-scale archival requirements. The author's contribution lies in formalizing this architecture as a coherent, production-ready framework for financial systems, connecting algorithmic optimization with business and regulatory constraints.

To achieve this aim, the following objectives are set:

- To analyze key architectural patterns (Event-Driven, LMAX Disruptor) and technologies (Apache Kafka, In-Memory Data Grids, lock-free data structures) for building a high-performance "hot" storage tier.

- To assess and compare approaches to implementing a cost-effective "cold" tier, including object storage and distributed file systems.

- To investigate and propose mechanisms for ensuring strict data consistency (for example, two-phase commit, Saga pattern) and fault tolerance (Disaster Recovery) in the context of the proposed hybrid model.

The scientific novelty of the work lies in the synthesis of advanced but often fragmented concepts into a single, coherent architecture. Unlike studies focusing on specific aspects, such as queue optimization or the choice of a particular database, this work proposes an integrated framework. This framework combines an event-driven architecture as the foundation of the entire system, the principles of mechanical sympathy for extreme optimization of the processing core, and a multi-tier storage model with clearly defined data migration policies between tiers.

The author's hypothesis is that the integration of an event-driven Apache Kafka bus with a high-performance hot tier based on an In-Memory Data Grid, optimized through the LMAX Disruptor pattern and lock-free structures, and a cold tier based on an object storage system using the Apache Iceberg format provides both sub-millisecond processing of financial transactions and cost-effective long-term archival data storage.

2. Materials and Methods

Contemporary research in the field of hybrid storage for financial data reflects both the technological and strategic evolution of architectures integrating in-memory solutions with cold storage tiers. In the works of Ionescu S. A., Diaconita V., Radu A. O. [1], the emphasis is placed on the principles of sustainable design of architectures oriented toward financial institutions, where energy efficiency and adaptive scalability play a key role. Mamidi S. [6] examines the practical implementation of hybrid solutions based on Cassandra and Gemfire in the context of financial services, demonstrating the advantages of integrating transactional and analytical workloads within a single platform. Chi Y., et al. [5] propose an operationally supported reconfigurable hybrid memory architecture capable of dynamically altering the configuration between DRAM and NVM depending on the workload profile. The review by Song, H., et al. [2] systematizes HTAP (Hybrid Transactional/Analytical Processing) approaches, including balancing between response time and storage cost.

A separate body of research is devoted to intelligent management of data placement between memory and storage tiers. Ren J., et al. [4] describe the application of machine learning for automated determination of the optimal storage tier, which minimizes access latency and increases resource utilization efficiency. Lu K., Zhao S., Wan J. [9] propose the Hammer method, based on online learning for real-time identification of hot and cold data, which is particularly relevant for systems with variable load. Chang J., et al. [10] integrate reinforcement learning methods into multi-tier main memory, providing adaptive optimization of data placement. Yuan Z., et al.

[3] focus on data classification using seasonal textual features, proposing an approach aimed at reducing costs in the long-term storage of large volumes of historical information. Telenik S.T. et al. [13] focuses on mathematical and algorithmic approaches, including queuing theory, artificial intelligence, and systems analysis, to improve the efficiency, reliability, and lifecycle management of large-scale cloud infrastructures. The paper extends these principles to the financial technology space, applying them to hybrid in-memory and cold storage architectures designed to meet the extreme performance and data archiving requirements of high-frequency trading systems.

Issues of performance and the choice of an optimal storage model are addressed by Rabelo Ferreira F. E. R., do Nascimento Fidalgo R. [7], who compare hybrid and columnar cloud databases in the context of schema design for distributed storage. Singh B., et al. [11] analyze the efficiency of various cloud DBMSs in processing stock market data, identifying bottlenecks in throughput and the impact of architectural decisions on latency.

The market development context is reflected in Technavio reports for the periods 2018–2022 [12] and 2024–2028 [8], which note the steady growth of the private and public cloud solutions sector in the financial industry, driven by the need for big data processing and AI adoption. Forecasts indicate the expansion of hybrid architecture use as a compromise solution between the flexibility of the cloud and control over critically important information.

The analysis of results from other studies demonstrated that the literature reveals two key lines — technological (architectures, optimization algorithms, memory integration) and strategic (selection of storage models, market trends). However, a gap remains between them: research rarely links the micro-level of architectural decisions with the macro-level of economic feasibility and regulatory constraints for the financial sector.

In addition, the following aspects are insufficiently covered:

- methods for integrating HTAP with long-term archiving systems that take into account legal requirements for storing financial data;

-issues of energy efficiency when working with ever-growing historical datasets;

- scenarios for transitioning between in-memory and cold storage under sharp peak loads;

- the impact of cloud providers' pricing models on architectural choices in hybrid systems.

To address these gaps, this work introduces HyFDS — an architecture that, to the author's knowledge, is the first to unify Kafka as an event backbone, an LMAX-optimized in-memory hot tier, and an Iceberg-based cold tier in a single financial storage model.

In turn, further research should focus on an end-to-end methodology linking algorithmic optimization with business models and regulatory aspects of operating hybrid storage facilities in the financial sector.

3. Results and Discussion

To address the stated problem, a multilayer architecture HyFDS (Hybrid Financial Data Store) is proposed, based on the principles of the event-driven approach (Event-Driven Architecture, EDA). The system is decomposed into logical layers, each specialized for its role, from event ingestion and routing to long-term persistence.

Level 1: Ingest & Event Backbone. The backbone component is Apache Kafka. All incoming streams—market quotes, trade orders, system notifications—are published as events to the corresponding Kafka topics. This scheme ensures loose coupling of components and their independent evolution and scaling. Kafka provides high throughput, horizontal scalability, and fault tolerance through data replication.

Level 2: Hot Tier. This level is intended for operational hot data that require minimal access and processing latency. It is implemented using an In-Memory Data Grid (IMDG) and stores active orders, current positions, and market data for the last trading day. The entire layer is engineered to ensure latencies of no more than 1 millisecond.

Level 3: Cold Tier. This level represents a cost-effective long-term store for cold historical data, including executed trades from previous years, complete event logs, and archival quotes. The key requirements for this layer are reliability, scalability, and a low cost of storage per terabyte.

Level 4: Data Lifecycle Management. A specialized component responsible for the automatic and transparent

migration of data between the “hot” and “cold” tiers based on predefined policies [3, 9].

Figure 1 shows the Apache Kafka architecture diagram, illustrating the data flow between its components.

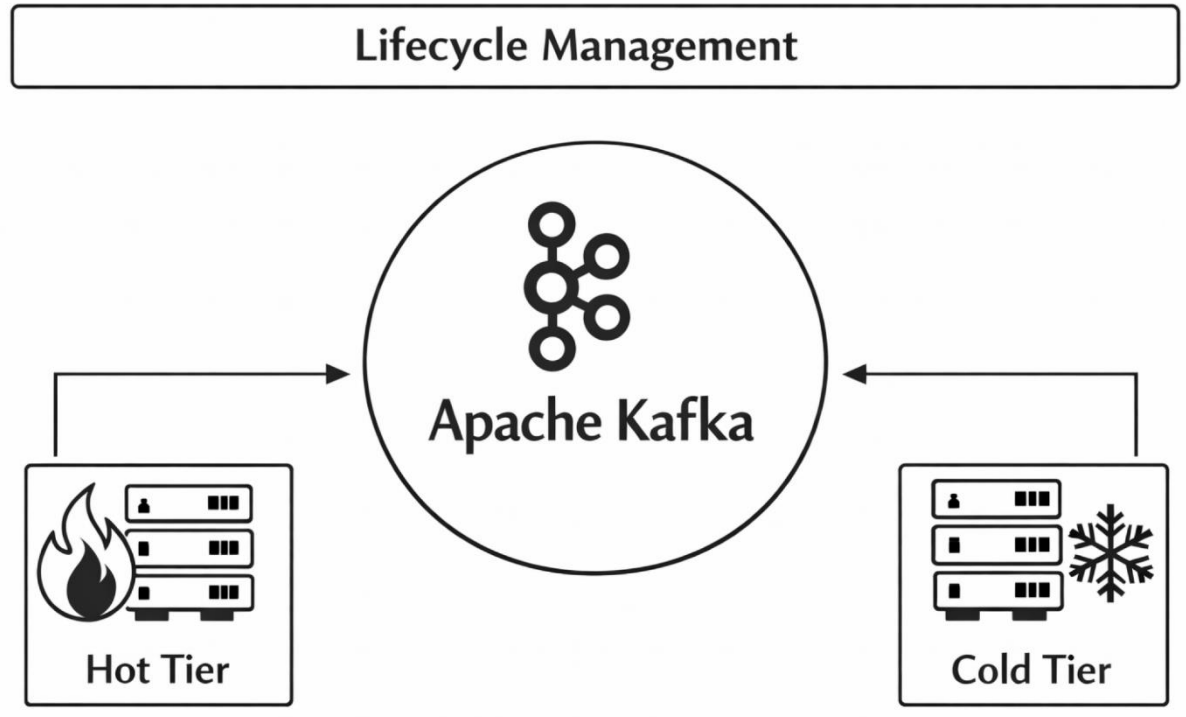

Fig. 1. Apache Kafka architecture diagram [3, 9].

The key advantage of the proposed architecture lies in using Kafka not merely as a message queue, but as a unified nervous system for the entire platform. Unlike traditional systems, where separate and often unsynchronized paths exist for real-time data processing and subsequent batch loading into the analytical storage, HyFDS uses a single event stream. Kafka, by its nature, is a persistent, replicated event log that not only delivers messages but also stores them and allows history to be rewound. This enables the same Kafka event stream to simultaneously perform three critically important functions:
Serve as a low-latency data delivery bus for the trading engine (Business Logic Processor).

Act as a data source for populating and updating the state of the Hot Tier.

Serve as a reliable buffer for asynchronous, fault-tolerant data recording into the Cold Tier.

This approach radically simplifies the architecture, eliminating the need for complex and fragile ETL processes. Moreover, it guarantees that both the Hot and Cold tiers are derived from the same immutable event log. This is a fundamental property for ensuring data integrity, performing audits, and, most importantly, enabling full and consistent system recovery after failures [4, 7].

To meet the strict requirement of latency below 1 ms, designing the Hot Tier requires a multi-layer optimization approach covering application architecture, algorithms, and infrastructure.

At the core of the Hot Tier lies the Business Logic Processor, designed with principles of mechanical sympathy toward modern hardware. The most effective approach here is the architecture proposed by LMAX Exchange. Its key principles are:

- Single-Threaded Business Logic: Instead of parallelizing business logic across multiple threads and dealing with synchronization issues, LMAX executes all critical logic (order matching, risk management) in a single thread. This completely eliminates the need for

locks, mutexes, and semaphores, and therefore also removes related overhead from context switching, OS scheduler intervention, and potential deadlocks.

- Ring Buffer: For passing data between threads (for example, between the thread receiving data from the network and the business logic thread), instead of traditional blocking queues, a lock-free data structure is used — the ring buffer, known as the Disruptor. This buffer is a preallocated array in memory, which avoids dynamic memory allocation at runtime. Its design is cache-friendly, minimizing costly cache misses.

In the author's applied experience designing trading infrastructures, similar architectural optimizations have demonstrated scalability to millions of daily transactions with sub-millisecond latencies. This confirms that the HyFDS approach is not only theoretical but also practically validated in high-load financial environments.

In those system components where a single thread is insufficient and parallelism is required (for example, for statistical data aggregation, logging, or interaction with external systems), traditional locks should be avoided. In this case, lock-free data structures should be used. They employ low-level atomic CPU instructions, such as Compare-And-Swap (CAS), to manage access to shared data without blocking threads [10, 11].

Under high contention, which is the norm for financial systems, lock-free implementations of structures such as FIFO queues and sorted lists significantly outperform their blocking counterparts in both performance and energy efficiency. For example, a lock-free FIFO queue where producers and consumers work from opposite ends can show almost twice the throughput compared to a mutex-based version, since head and tail operations can be executed in parallel [13]. However, this advantage is not absolute. Under low contention, the overhead of atomic operations and retry loops can make lock-free code slower. Moreover, the performance of certain lock-free algorithms can degrade under a very large number of competing threads due to cache coherence issues and frequent CAS failures. The choice between blocking and lock-free approaches should be based on the analysis of the specific use case, as reflected in Table 1.

Table 1. Comparative analysis of the performance of Lock-Free and blocking data structures under high contention [4, 7, 10, 11]

Data Structure	Level of Contention	Lock-Free Performance (vs Locking)	Lock-Free Energy Efficiency (vs Locking)	Key Considerations
FIFO Queue	High (Producer/Consumer)	Significantly higher (up to 2x)	Significantly higher	Ideal for pipelines. The advantage is due to the parallelism of head and tail operations.
Double-Ended Queue (Deque)	High	Lower or slightly higher	Lower or comparable	In implementations where head and tail are updated atomically together, there is no internal parallelism, which negates the advantages of lock-free.
Sorted List	High	Significantly higher	Significantly higher	Demonstrates nearly linear speedup. Each node can be updated independently, providing high parallelism.

To ensure horizontal scalability and high availability of the hot tier beyond a single server, In-Memory Data Grids (IMDG) are used. IMDG distribute data in RAM across a cluster of servers, providing a unified interface for accessing them. The leading solutions in this field are Apache Ignite, Hazelcast, and Redis.

Thus, achieving a high level of performance in the hot tier is not the result of a single technological choice, but

the consequence of coordinated optimizations at multiple levels of abstraction. The requirement of latency <1 ms cannot be met simply by choosing a fast database, since latency accumulates at every stage: from network interaction and deserialization to queuing, executing business logic, and committing a transaction. HyFDS implements a hierarchical optimization approach:

- At the architectural level, the LMAX Disruptor pattern is used to completely eliminate locks in the system core.

- At the algorithmic level, where multithreading is required, lock-free data structures are applied to minimize conflicts at the CPU level.

- At the infrastructure level, a high-performance distributed IMDG (Apache Ignite) is used to provide horizontal scalability and fault tolerance.

This multi-level approach makes it possible to systematically attack the latency problem on all fronts, from software architecture to hardware specifics and distributed infrastructure.

Next, within the framework of the study, it is necessary to proceed to the examination of the specifics of managing the cold storage and the data lifecycle. Effective management is the second key task of a hybrid architecture. The main criteria here are low storage cost, unlimited scalability, and sufficient performance for executing analytical queries [5, 6].

There are several approaches to implementing cold storage:

- Object storages (e.g., Amazon S3, Google Cloud Storage): They are the most cost-effective solution for storing data at the petabyte scale. However, direct querying of raw files in object storage is generally inefficient. Modern analytical systems such as Apache Doris can use S3 as a cold tier, automatically retrieving data on demand, but this is associated with certain delays.

- Hadoop Distributed File System (HDFS): The traditional foundation of the Big Data ecosystem, tightly integrated with analytical frameworks such as Apache Spark and Hive. HDFS is less cost-effective and more complex to administer compared to cloud object storages.

- Columnar DBMS (e.g., HBase, ClickHouse): HBase is well suited for storing massive datasets, but its read performance is limited by disk access speed. ClickHouse is optimized for analytical queries (OLAP), but its operation can be more expensive compared to object storages.

For the HyFDS architecture, a modern hybrid approach is proposed: the use of the Apache Iceberg tabular format on top of the S3 object storage. Apache Iceberg addresses the key issues of "raw" files in S3: it provides a table abstraction, supports ACID transactions for analytical operations, enables data versioning, and offers efficient partition pruning. This ensures seamless integration with leading analytical engines (Spark, Trino, Doris). Thus, it becomes possible to combine the low cost and scalability of S3 with the manageability, reliability, and performance typical of traditional data warehouses.

The process of moving data from the "hot" tier to the "cold" tier must be automated and governed by policies. As a basis, the principle proposed in the RHTSDB (Redis-HBase Time Series Database) model can be adopted, where data is separated according to access frequency: new and frequently used data are stored in Redis (hot), while older and rarely used data are stored in HBase (cold).

In the HyFDS architecture, this function is performed by the Data Lifecycle Manager component. It periodically scans data in the "hot" tier (Apache Ignite) and, based on a set of rules, determines which data should be migrated. The marked data is asynchronously copied to the "cold" storage (to an Iceberg table on S3) and, after successful verification, deleted from the IMDG, freeing up expensive RAM [3, 4].

The choice of migration policy is not only a technical decision but also an important economic one. The cost of storing 1 TB of data in RAM is orders of magnitude higher than the cost of storing the same volume in S3. The business requires instantaneous access to operational data but can tolerate a delay of several seconds when querying an archive that is a year old. Therefore, the "temperature" of data serves as a proxy metric for its current business value. By implementing the migration policy, the Data Lifecycle Manager becomes not merely a technical utility but a tool for financial and operational management of the system. Too aggressive a migration reduces infrastructure costs but may increase latency for some analytical queries. Too conservative a migration unjustifiably increases operating expenses. The author formalizes this balance into a repeatable lifecycle framework that enables financial institutions to reduce storage costs while meeting regulatory retention requirements. This elevates the Data Lifecycle Manager

from a technical utility to a strategic tool for compliance and financial efficiency.

To ensure data consistency during operations affecting multiple nodes or services, there are two main approaches:

Two-Phase Commit (2PC): A classical protocol ensuring strict atomicity (ACID). It guarantees that a distributed transaction will either complete successfully on all participating nodes or be fully rolled back on all of them. The main drawback of 2PC is its blocking nature. Participants lock resources until they receive the final command from the coordinator, which increases latency and creates a single point of failure in the coordinator.

Saga Pattern: An alternative based on the BASE philosophy (Basically Available, Soft state, Eventually consistent). A long transaction is broken into a sequence of local transactions, each executed in a separate service. For each local transaction, there must exist a compensating transaction that reverses its effect. Saga is non-blocking, more performant, and fault-tolerant, but it provides only eventual consistency, which may be unacceptable for certain operations.

In the HyFDS architecture, it is not advisable to apply a single approach across the entire system. The choice should depend on the business requirements of a particular operation (see Table 2).

For critical, short-lived operations within the Hot Tier, where strict atomicity is required (for example, the execution of a trade affecting multiple partitions in the IMDG), the use of optimized versions of 2PC or consensus protocols such as Paxos or Raft is justified. Modern databases such as CockroachDB implement non-blocking variants of 2PC (e.g., Parallel Commits), which significantly reduce latency [9, 10].

For long-running business workflows that traverse multiple services—for example, “new client registration → KYC verification → account funding → trading authorization”—the Saga pattern is the preferred coordination paradigm. Holding locks or otherwise monopolizing resources for the entire lifecycle of such a workflow is inadmissible.

Table 2. Decision Matrix: 2PC vs. Saga [3, 4, 9, 10]

Criterion	Two-Phase Commit (2PC)	Saga Pattern
Consistency guarantees	Strict (ACID), synchronous	Eventual (BASE), asynchronous
Performance/Latency	Low/High (due to locks)	High/Low (non-blocking)
Fault tolerance	Low (sensitive to coordinator failure)	High (decentralized)
Implementation complexity	Relatively simple in standard DBMS, but complex in recovery	Requires complex compensation logic and monitoring
Typical scenario in HyFDS	Atomic update of multiple records in IMDG	Long business process (for example, customer onboarding)

A Disaster Recovery (DR) plan must ensure attainment of two key target metrics. The first — Recovery Time Objective (RTO), that is, the maximum allowable duration of downtime; for mission-critical trading infrastructure this value should approach zero. The second — Recovery Point Objective (RPO), that is, the maximum allowable volume of data loss from the moment of failure; for financial transactions the requirement for RPO is also zero.

The HyFDS architecture satisfies these conditions through multi-tier replication and event logging mechanisms. Replication is implemented at all levels of the system: Apache Kafka provides geographically distributed replication of event streams across multiple data centers; Apache Ignite supports both synchronous

and asynchronous replication of clusters between sites; cold storage based on S3 by default uses cross-region replication [2, 7].

The recovery procedure is built on the principle of an immutable event log (Event Sourcing), where Kafka serves as the unified repository of all state changes. This approach significantly simplifies returning the system to a consistent state. In the event of complete unavailability of the primary data center, the standby site is activated, and the state of the hot tier (In-Memory Data Grid, IMDG) is fully reconstructed by sequentially replaying the replicated Kafka event log from the last consistent point. This methodology enables exceptionally small RTO and RPO, which is critical for ensuring the continuity of financial operations.

4. Conclusion

This study validates the HyFDS hybrid data storage architecture, an original framework authored to resolve the fundamental contradiction of modern financial systems: the need for extreme performance alongside scalable archival. By combining Kafka as the event backbone, an LMAX-optimized IMDG hot tier, and Iceberg-enabled cold storage, the architecture achieves both sub-millisecond processing and petabyte-scale archival at sustainable cost. Beyond trading, HyFDS has broader implications for fraud detection, compliance auditing, and regulatory reporting, highlighting its significance to the wider fintech sector. The research objective is achieved through a carefully thought-out synthesis of advanced technologies and architectural patterns. Key provisions and components of the architecture: An event foundation based on Apache Kafka, acting as the unified nervous system of the platform, providing loose coupling, scalability, and serving as the single source of truth for all state changes. A high-performance hot layer based on an In-Memory Data Grid (Apache Ignite), whose performance is maximized through hierarchical optimization: application of the LMAX Disruptor pattern to eliminate locking in critical business logic and the use of non-blocking data structures in auxiliary multithreaded components. A cost-effective and scalable cold layer on S3 object storage with the Apache Iceberg table format, combining low storage cost with manageability and efficiency of analytical queries. The architecture formalizes data lifecycle management strategies through automated migration between layers, supports transactional integrity with a hybrid approach (2PC/Saga), and ensures business continuity through multi-level replication and disaster recovery plans that rely on the event log.

References

1. Ionescu, S.-A., Diaconita, V., & Radu, A.-O. (2025). Engineering Sustainable Data Architectures for Modern Financial Institutions. Electronics, 14(8), 1650. https://doi.org/10.3390/electronics14081650

2. Song, H., et al. (2024). A survey on hybrid transactional and analytical processing. The VLDB Journal, 33, 1485–1515.

3. Yuan, Z., et al. (2024). Cost-effective data classification storage through text seasonal features. Future Generation Computer Systems, 158, 472–487.

4. Ren, J., et al. (2021). A machine learning assisted data placement mechanism for hybrid storage systems. Journal of Systems Architecture, 120. https://doi.org/10.1016/j.sysarc.2021.102295

5. Chi, Y., Liu, H., Peng, G., Liao, X., & Jin, H. (2022). Transformer: An OS-Supported Reconfigurable Hybrid Memory Architecture. Applied Sciences, 12(24), 12995. https://doi.org/10.3390/app122412995

6. Mamidi, S. (2025). Next-generation data management: Hybrid approaches with Cassandra and Gemfire in financial services. International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 11(2), 1563–1574.https://doi.org/10.32628/CSEIT25112531

7. Rabelo Ferreira, F. E. R., & do Nascimento Fidalgo, R. (2024). A Performance Analysis of Hybrid and Columnar Cloud Databases for Efficient Schema Design in Distributed Data Warehouse as a Service. Data, 9(8), 99. https://doi.org/10.3390/data9080099

8. Technavio. Private and public cloud market in financial services to grow by USD 106.43 billion (2024–2028), driven by big data demand; AI driving market transformation. PR Newswire.

Retivered from: https://www.prnewswire.com/news-releases/private-and-public-cloud-market-in-financial-services-to-grow-by-usd-106-43-billion-2024-2028-driven-by-big-data-demand-ai-driving-market-transformation---technavio-302352741.html?utm_source (date of access: 10.07.2025)

9. Lu, K., Zhao, S., & Wan, J. (2024). Hammer: Towards efficient hot-cold data identification via online learning. arXiv, 1-10.

10. Chang, J., et al. (2024). Idt: Intelligent data placement for multi-tiered main memory with reinforcement learning. In Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing, 69-82. https://doi.org/10.1145/3625549.3658659

11. Singh, B., Martyr, R., Medland, T., Astin, J., Hunter, G., & Nebel, J. C. (2022). Cloud based evaluation of databases for stock market data. Journal of Cloud Computing, 11, 53.

12. Technavio. Private and public cloud market in the financial services industry 2018–2022 (historical market size). Retivered from: https://www.technavio.com/report/private-and-public-cloud-market-in-the-financial-services-industry-analysis (date of access: 10.07.2025)

13. Telenik S.T. et al. Development and research of models, methods and technologies of planning, programming and management cloudy IT-infrastructures. Retivered from:https://nrat.ukrintei.ua/en/searchdoc/0216U005227/ (date of access: 10.07.2025)