SSRG International Journal of Computer Science and Engineering
ISSN: 2348–8387 / https://doi.org/10.14445/23488387/IJCSE-V12I1P103

Volume 12 Issue 1, 11-15, January 2025
© 2025 Seventh Sense Research Group®

Original Article

Modernizing Banking Systems: A Strategic Shift from Mainframes to Agile Infrastructure

Tarun Mathur

Senior Architect, New Jersey, United States.

Corresponding Author : tarunmathur@live.in

Received: 17 November 2024 Revised: 23 December 2024 Accepted: 12 January 2025 Published: 30 January 2025

Abstract - *Since the 1960s and 1970s, mainframe systems have been at the heart of core banking operations, automating processes and carrying out high-volume transactional work. Using legacy systems has reliably served customers' needs but also presents major concerns: operational costs, a lack of scalability, and fewer resources specializing in COBOL. The need for a smooth, secure, and fast customer experience through digital banking encounters keeps the struggling aged infrastructure out of pace, which demands modernization as an absolute necessity. The present paper looks at the methodology for gradual, step-by-step shifting dependency on mainframes without breaking operational continuity. Based on workload segregation, the paper identifies operations under three categories of critical, semi-critical, and noncritical for prioritization of modernization efforts. Critical workloads, such as real-time customer transactions, use caching solutions and secondary databases to optimize performance with consistency in data. Semi-critical workloads, such as batch processing, are migrated to secondary systems with periodic synchronization, while non-critical workloads, such as the retrieval of archival data, are migrated to lower-cost platforms. The proposed framework provides secondary databases for reading purposes, synchronized in real-time through tools like Apache Kafka, RabbitMQ or NATS, and caching solutions to reduce the further load on the mainframes. In due course of time, all write operations can be routed slowly to the secondary database, thus gradually retiring the mainframe systems. This is a step-by-step methodology of minimizing disruption, preserving data integrity, reducing risk, and providing a scalable, cost-efficient infrastructure. The framework not only solves issues like Million Instructions Per Second (MIPS) costs and integration of newer technologies but also positions financial institutions for evolving customer demand, innovation, and operational excellence, instilling a sense of optimism and hope for the future of banking systems.*

Keywords - *Mainframe modernization, COBOL dependency, Banking systems, Resiliency, Caching solutions, MIPS cost reduction.*

1. Introduction

The banking industry has relied heavily on mainframe systems for core operations since the 1960s and 1970s, automating routine tasks to save manual labor costs. Revolutionizing the sector, these COBOL applications with DB2 databases replaced paper-based records with digital storage, bringing faster data access and better data accuracy—supporting high-volume transaction processing [1]. Mainframes were instrumental in introducing revolutionary services like Electronic Funds Transfer (EFT), increasing the speed and security of transactions, and laying the foundation for future developments like Automated Teller Machines (ATM) and online banking. However, in this increasingly digitized financial landscape, banks are under heavy pressure to revamp their technology infrastructure as clients demand frictionless, efficient, and safe banking via mobile devices and online platforms. Since most of the existing mainframe systems are aging and not designed to handle the demands of a modern digital banking ecosystem, transformation is urgently needed without disrupting critical operations [2]. This continued reliance on legacy systems is a barrier to innovation and growth, which is further compounded by the gradually dwindling reservoir of COBOL and FORTRAN expertise as seasoned developers are retiring. Younger generations are not getting trained in these technologies. Integration with cloud-based technologies and API-driven innovations—needs for scalability and flexibility—is also quite cumbersome for mainframes. Most institutions cannot make any changes to the mainframe software without adequate documentation in place. High operating costs, fueled by MIPS pricing models and ever-increasing transaction volumes from mobile and online banking, compound these issues [3, 4]. While financial institutions acknowledge the challenges of aging mainframe systems, the need for pragmatic and incremental modernization solutions is more urgent than ever. Most of the literature focuses on comprehensive system overhauls, which can be prohibitively costly and risky for organizations reliant on legacy infrastructure. For instance, an expensive 'big-bang' mainframe modernization program could disrupt operations in a way that would make incremental approaches more attractive for cost reduction and mitigation of operational risks. The "big-bang approach," where the entire mainframe system is replaced at once, is generally referred to as high-risk, which can cause disruption, budget overrun, and unforeseen technical problems. With incremental modernization, this approach provides a step-by-step, iterative process where specific components of the mainframe ecosystem are slowly updated and improved.

While the advantages of an incremental approach, for example, have been deliberated upon at several platforms, much research and case studies have not been conducted on such approaches. This indicates that further probes in phase modernization methodologies become indispensable. In such ways, it is essential that the banking industry continuously upgrade the systems by maintaining stability and reducing risks related to operations. This paper introduces a novel approach to incrementally modernize the aging mainframe systems of the banking sector, considering all associated challenges with such legacy systems. Unlike conventional methods involving complete overhauls, this study emphasizes practical, low-risk solutions for modernizing specific components. It provides key innovations: secondary database integration, offloading data processing, innovative caching mechanisms that manage high-frequency requests much more effectively, and the development of hybrid models for enabling mainframes towards modern cloud-based and API-driven platforms. Finally, it should also offer various cost optimization approaches, such as alleviating the costs linked with MIPS pricing and disaster recovery. The paper presents a unique, phased modernization plan, balancing innovation with the realities of maintaining critical banking operations through the lens of operational continuity and minimal disruption.

2. Methodology

A three-phase approach to deal with the complexities in mainframe systems and for progressive reduction of the dependence on them is being discussed here. The solution includes secondary databases for reading operations to be performed, plus caching for critical applications, with both providing improvement in performance and scalability while decreasing related mainframe operational costs—those that pertain mainly to MIPS charges.

2.1. Introducing a Secondary Database for Read Operations

The first step is to create a second database to handle read operations like customer logins, ATM withdrawals, and service inquiries. Write operations will continue in the primary database due to compatibility issues related to the existence of legacy systems and backend processes, most of which are neither documented nor well understood by the developers and architects involved. A secondary database will be maintained in sync with the primary one by a daily batch process, ensuring it has the most recent data. Besides, there will also be real-time updates whenever data gets written to the primary database. This will be achieved using distributed messaging systems such as Apache Kafka, RabbitMQ and NATS as a transport layer to ensure efficient and reliable data replication [5].

2.2. Implementing a Caching Solution for Critical Use Cases

The system is implemented first to seek data from the cache for fast retrieval. Implementing a caching system can improve performance and reduce mainframe dependency for mission-critical applications. If the required data is not found, corrupted, or outdated in the cache, the system will send a query to a secondary database. The system rarely resorts to the primary database in case there are technical problems with the secondary database. The layered approach, therefore, brings about a minimization of the direct calls made to the mainframe database; it reduces the load on mainframes and decreases MIPS costs. More recently, performance can also be enhanced further using distributed caching systems that keep repeatedly accessed data memory-based and nearer to the applications, reducing latencies and backend burdens, generally adding to operational efficiencies. This structured caching method leads an organization through a resilient data-retrieval process that optimizes performance and cost-effectiveness efficiently.

2.3. Workload Segmentation

Modernization of mainframe systems involves a careful categorization of workloads based on needs and operational priorities [6]. There are three major categories that a workload can fit into: critical, semi-critical, or non-critical. Critical workloads, including real-time customer transactions and fraud detection, will demand high availability, low latency, and strong data consistency. This kind of workload is offloaded to caching solutions and secondary databases for best performance while taking the load off the mainframes. Semi-critical workloads, like report generation and batch processing, can be offloaded to secondary databases that synchronize periodically to maintain data consistency without swamping the system. Non-critical workloads, such as retrieving archived data and infrequent queries, are the best-suited ones for offloading to cloud-based systems or other cost-effective storage platforms. This classification allows financial institutions to focus modernization efforts in a manner that ensures critical services are maintained while optimizing resources for less demanding operations.

2.4. Monitoring and Optimization

It is now time to monitor the system's transactions for some months to see how the new architecture behaves. First, the check will ensure that the secondary database or cache is serving mainly read traffic. Monitoring tools will be placed to measure the number of calls, together with the successes at each layer, to understand the performance and savings occurring and further optimizations to be performed. From this analysis, further system improvement can be made, and/or legacy flows can be progressively retired when the assurance in the secondary database and caching layers increases. This type of iteration will make the cutover painless and limit the associated risk of disrupting ongoing business operations.

2.5. Transitioning Write Operations to the Secondary Database

Once the secondary database and caching layers prove reliable for read operations, developers will start identifying all write flows in the system systematically. The developers and architects will document these flows and then prioritize those that can be transitioned to the secondary database with the least impact on operations. The write flows will gradually be shifted to the secondary database, which continues to synchronize with the primary for continuity. As more write operations are transitioned and trust in the secondary database builds, it will be able to reduce dependency on the mainframe database. Finally, the primary or mainframe database can be deprecated to reduce costs and dependency on legacy mainframes significantly. This step-by-step process ensures a smooth transition to modern infrastructure, minimizes risks, and assures the integrity of data with no disruption in services to end-users.

This architecture outlines a step-by-step approach to reduce mainframe dependency. Step (1) involves applications initiating data requests, which pass through the API layer (2) that determines the appropriate data source. The API first queries the distributed cache (3) for critical use cases to retrieve data quickly. If the cache is unavailable or outdated, the request is forwarded to the secondary database (4), which handles most read operations. Real-time synchronization between the primary database (5) and the secondary database is facilitated using Kafka to ensure consistency. The primary database handles all write operations and legacy flows as a fallback for rare data retrieval cases. Finally, step (6) involves a batch reconciliation process at the end of each business day to update the secondary database with any changes from the primary database, maintaining consistency across the system. (7) Legacy applications or systems continue writing into mainframe systems to avoid disruption.

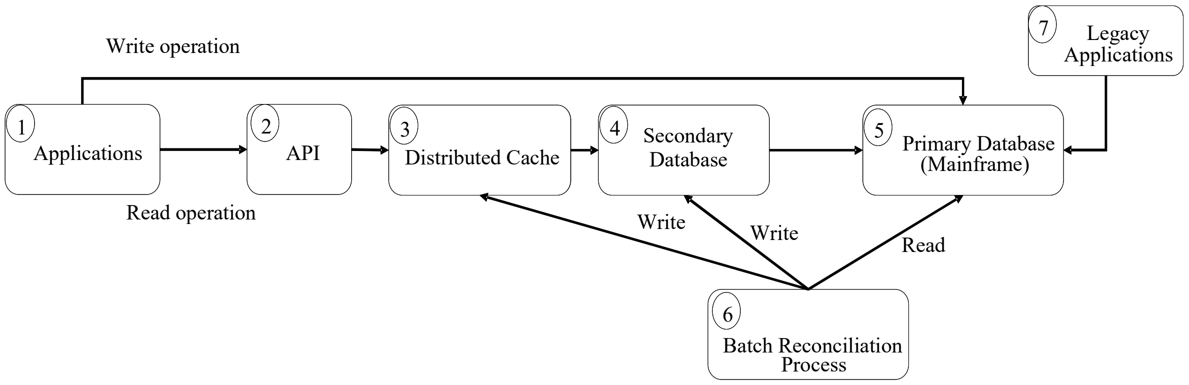

Fig. 1 Proposed architecture for gradual mainframe dependency reduction in banking systems

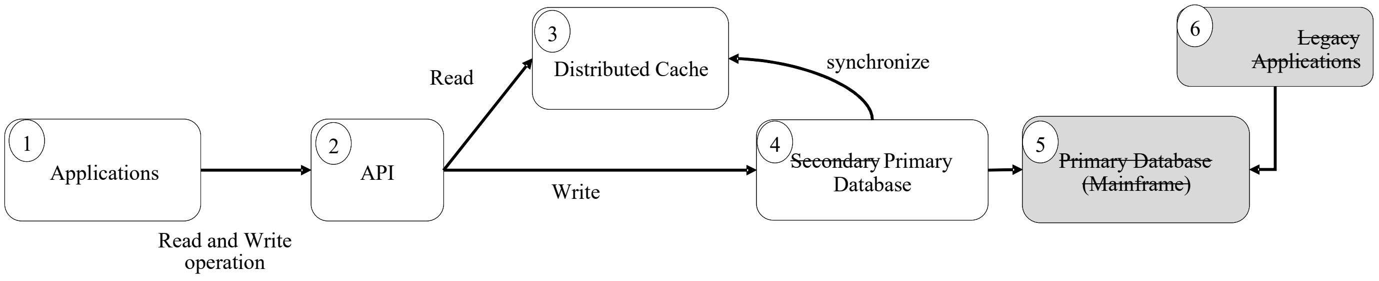
3 Distributed Cache
Read
synchronize
6 ~~Legacy Applications~~
1 Applications
2 API
Write
4 ~~Secondary~~ Primary Database
5 ~~Primary Database (Mainframe)~~
Read and Write operation

Fig. 2 Final phase architecture for eliminating mainframe dependency in banking systems

This diagram represents the final phase of transitioning from mainframe dependency in banking systems. In this phase, applications (1) interact with APIs (2), which direct requests to a distributed cache (3) for critical and frequently accessed data. The secondary database (4) now serves as the primary source for all read-and-write operations, completely replacing the original mainframe database. Real-time data consistency is maintained within the modernized infrastructure, eliminating reliance on legacy applications and the mainframe database (5). This architecture demonstrates an entirely modernized system that is cost-efficient, scalable, and capable of meeting evolving customer demands without the constraints of legacy systems.

3. Results and Discussion

This architectural strategy slashes the cost of mainframes in the financial institution world without giving up system reliability and performance. A bank, for instance, would conventionally rely on its mainframe to process millions of customer transactions on any given day. Using a caching layer between clients and the organization's backend systems, frequently accessed data, such as account balances and recent transaction history, could be cached. Caching offloaded more than 60% of read operations from the bank's mainframe, thus shaving mainframe transaction processing costs by approximately $2 million annually. Additionally, the bank used a second-tier distributed database to store historical transaction data older than two years. The migration freed expensive mainframe storage and improved analytics and regulatory reporting query performance. For instance, compliance reporting is faster and cheaper, pulling data from the second-tier distributed database instead of the mainframe.

4. Conclusion

Financial institutions cannot afford to lag in mainframe use to remain competitive in an increasingly dynamic and technology-driven environment. This paper presents a strategic framework for legacy system modernization, focusing on workload segmentation, secondary databases, and caching solutions that will help increase performance while reducing costs and ensuring scalable efficiency. Segmentation into critical, semi-critical, and non-critical workloads will enable effective prioritization of modernization work while ensuring service continuity for the mission-critical operations of the institution. It offers a structured methodology towards modernization in steps. It also brings significant relief for the mainframes by offloading the read operations to the secondary databases with the appropriate caching solutions, which thereby reduces operational costs, including the MIPS charges. When the functionality supports writing on secondary, it can be sustainable in the long term while reducing dependence on legacy infrastructure piece by piece. The hybrid model presents a no-risk strategy that maintains data integrity and allows financial institutions to meet customer and regulatory demands. This, in turn, leads to the final stage when the financial organizations move to a modernized infrastructure where the mainframes are replaced in all operations by secondary databases and caching solutions. This sets the stage for a cost-effective, flexible architecture to support scalability for future innovations. In this respect, by applying the given framework for modernization, financial institutions will be able to reach operational excellence, increase customer satisfaction, and become leaders in a volatile technology environment.

However, the study does acknowledge a number of limitations. One of the key challenges is assuming that all financial institutions or banks are equally ready to implement secondary database and caching solutions. Technological maturity, resource availability, and organizational priorities vary greatly. Smaller banks or those with very tight budgets may find it hard to allocate resources for incremental modernization, which may delay or complicate the process. Another limitation involves potential integration challenges. Seamlessly shifting workloads from mainframes into modern systems requires full compatibility between legacy systems and new technologies. For example, keeping primary and secondary databases in real-time sync can be complicated and error-prone, especially within high-transaction environments. Furthermore, basing one’s system on caching systems introduces vulnerabilities such as cache consistency issues and single points of failure, leading to interruption of operations.

Another limitation is the assumption that workload segmentation can be sharply divided into critical, semi-critical, and non-critical. Operations may fall within more than one category or even change in criticality over time, thus requiring constant re-evaluation and re-adjustment, which may be burdensome for the organization.

In the future, the framework would need comprehensive case studies involving the number of financial or banking institutions to assess its real-world applicability and effectiveness in terms of implementation. The development of adaptive strategies for overcoming integration challenges and the refinement of workload segmentation methods would also contribute to strengthening the robustness and applicability of the proposed framework. Adding these directions would fine-tune the approach to the divergent needs of financial institutions, allowing for a seamless transition into modernized systems.

References

[1] Gloria Mentonelli, Why COBOL Still Dominates Banking—and How to Modernize, Castsoftware, 2024. [Online]. Available: https://www.castsoftware.com/pulse/why-cobol-still-dominates-banking-and-how-to-modernize

[2] Alvaro Ruiz, Key Strategies and Approaches for Mainframe and Core Banking Modernization, Accenture Banking Blog, 2024. [Online]. Available: https://bankingblog.accenture.com/strategies-mainframe-core-banking-modernization

[3] Spas Tyurkedzhiev, Migrating the COBOL Legacy to Modern Systems and Their Challenges, Dreamix, 2022. [Online]. Available: https://dreamix.eu/insights/migrating-the-cobol-legacy-to-modern-systems-and-their-challenges/

[4] Million of Instruction Per Second, ScienceDirect. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/million-of-instruction-per-second#

[5] T. Sharvari, and K. Sowmya Nag, "A study on Modern Messaging Systems- Kafka, RabbitMQ and NATS Streaming," *Arxiv*, pp. 1-5, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Bing Hu, and Nicholas Mason, "Large Scale Analytics for Workload Segmentation," *Journal of Management & Engineering Integration*, vol. 16, no. 1, pp. 19-26, 2023. [CrossRef] [Google Scholar] [Publisher Link]