Rethink in Action

Proof Positive of Continuous Modernization of Business-Critical COBOL Systems

A CobolCloud Customer Success Story

Section 1. The Customer.

At the request of the organization involved, neither the company name nor the business sector are disclosed.

The scale of the application is immense, serving multiple geographies, supporting many clients, serving millions of users, and its strategic function is imperative to the success of the organization in question. Conceived in the late 1970s and early 1980s as a mainframe-based application, the system has remained continuously in production ever since, delivering critical calculation services. Over the years, its technical environment, interfaces, and execution models have evolved repeatedly. Since 2013, we have supported this organization on this journey.

This article is therefore more of a sequence of stories – rather than that of a single project - capturing the evolving improvement and modernization of the system over decades, protecting operational continuity, business consistency, and architectural control.

Shared Service

The application (or "platform") was designed as a centralized calculation environment shared across multiple organizations. Each client transmits its own datasets ahead of processing cycles, while centrally managed calculation rules and customer-specific adaptations are applied consistently across the platform. Results were then returned to each organization.

The platform progressively expanded to support calculations affecting several million individuals across Europe, and whose operational continuity became business-critical.

The application's core calculation function evolved into a COBOL codebase of a million lines, initiated by JCL-controlled process flows. Structurally, the application allowed customer-specific customization through dedicated COBOL modules capable of adapting centralized calculation rules without fragmenting the core application itself

Operations were supervised directly from the mainframe environment by teams who understood not only the application itself, but also its execution behavior, operational constraints, and production dependencies. The teams continuously separated two dimensions: preserving and evolving the business rules embedded in the application; and maintaining

architectural control over the execution environment.

Business complexity naturally continued to grow over time, but architectural debt remained under control - one of the key reasons the platform evolved so successfully over time.

Section 2. Opportunities and Solutions

Throughout its lifespan, the business criticality of this platform continued, as did an ambition to deliver best possible service. The IT leaders took periodic strategic decisions to support the platform's continued evolution and innovation.

Opportunity 1 – Expanding and Simplifying Access

In the early 2000s, the primary requirement was no longer extending functionality but simplifying operational access to the platform while preserving execution consistency.

Solution 1 – Citrix and Browsers

A Citrix-based infrastructure was introduced, allowing applications to be centralized and accessed through web browsers rather than local workstation installations. Internal operational tools progressively followed the same trajectory: calculation launches, supervision interfaces, customer-specific adjustments, and monitoring functions became accessible through browser-based operational environments. Daily operations no longer required direct interaction with the mainframe itself, while the underlying execution model remained unchanged.

One architectural decision proved particularly important over the long term.

Opportunity 2 – Improving Mainframe production efficiency

A dedicated simulation platform was introduced to validate rule changes before production deployment. Instead of building a separate implementation, the simulation environment reused the exact same COBOL source code as production.

Validation therefore relied on identical execution behavior rather than approximation.

Solution 2 – Rule Validation Portability breakthrough

This simulation environment was progressively deployed outside the mainframe, initially on SCO Unix systems, while still relying on the same COBOL application core.

At that stage, the architecture had already started evolving toward a distributed and hybrid execution model. This became the first major step toward progressively reducing dependency on the mainframe environment while preserving what mattered most, namely business logic, execution semantics, operational behavior, and the expertise accumulated around the platform itself.

Distributed components communicated with the central platform through controlled file exchanges and messaging mechanisms, introducing flexibility without sacrificing determinism.

Opportunity 3 – Embrace an Open Strategy

As part of an increasing strategic and cultural move towards open systems, execution environments progressively moved from the mainframe toward distributed Unix systems and later Linux environments.

Solution 3 – Preserve business logic through platform portability

Throughout these transitions, the COBOL application core itself remained largely unchanged. JCL-controlled execution chains were progressively transformed into shell-based orchestration while preserving execution ordering, operational semantics, and production behavior.

Over time, additional application branches originating from separate mainframe environments were consolidated into the same distributed infrastructure. Modules that had evolved independently for years eventually operated side by side within a unified execution environment.

At that stage, the physical location of execution had become secondary. The priority was no longer tied to a specific infrastructure platform, but to preserving identical execution behavior, operational consistency, and deterministic results across environments. This continuity became one of the factors that made long-term transformation possible without destabilizing production operations.

Opportunity 4 – Achieving a dream of an elastic infrastructure.

The next major transition concerned infrastructure elasticity. However flexible the open systems-based architectural model had become, it still relied on fixed, on premises server infrastructure that represented a fixed asset cost and offered limited scalability or flexibility.

Exploring cloud computing potential was considered the next ambition.

Solution 4 - Genuine cloud flexibility without compromise.

The platform’s workload profile was particularly well suited to cloud execution models, with processing activity varying significantly throughout each monthly cycle.

At that stage, the production environment relied on a large combination of physical and logical servers hosted inside a traditional data center architecture. Maintaining infrastructure continuously sized for peak execution periods had progressively become difficult to justify both operationally and economically.

Cloud elasticity introduced a more coherent execution model, allowing compute capacity to adapt dynamically to actual workload intensity. AWS was ultimately selected as the target infrastructure for this transition.

The application itself was progressively transferred toward a cloud-oriented execution model through containerized deployment approaches, while preserving the existing COBOL application core, execution semantics, and operational behavior. This transition was completed several months ahead of the original schedule.

Opportunity 5 - Embracing new processor architectures.

Shortly afterward, AWS Graviton processors became available as a potential execution target. Operational efficiency and better throughput were a natural consideration for next-level innovation.

Solution 5 - Adopting ARM64.

The entire CobolCloud distribution was rebuilt on the ARM64 architecture within only a few hours. The flexibility and portability of CobolCloud technology enables rapid changes of deployment model without fuss. This rapid transition was made possible by years of engineering work focused on portability across successive infrastructure and processor generations.

Once the CobolCloud platform itself had been rebuilt on ARM64, existing customer applications could immediately be recompiled identically on the new architecture and evaluated under real execution conditions.

The objective was to validate that future processor evolution could be adopted without revisiting the existing COBOL application core. Initial tests confirmed the expected execution behavior, demonstrating that newer processor architectures no longer represented a structural constraint for the platform.

Opportunity 6 – Rethinking the Execution Model

After decades of continuous operation, the next optimization challenge had shifted away from the business logic itself toward the execution architecture surrounding it. The original platform architecture relied heavily on sequential execution patterns and extensive intermediate file exchanges. Senior engineers recognized the opportunity to increase execution workflow by removing the limitations imposed by the current sequential execution process.

Solution 6 – Parallel and in-memory execution

The next transformation phase progressively introduced a revised execution process, embracing

- In-memory execution pipelines
- Reduced intermediate data persistence
- Optimized data locality
- Horizontally parallelized execution flows.

Through this transition, context no longer needed to be repeatedly serialized and deserialized between execution stages, remaining available throughout the execution pipeline itself.

The main transformation concerned execution orchestration, data flow management, workload distribution, and parallel execution coordination. A low-level C component managed execution orchestration and context continuity, while Java-based ingestion services dynamically instantiated execution pipelines directly from centralized storage layers.

Depending on available infrastructure capacity, up to sixty-four execution pipelines could operate simultaneously.

This transition from a vertically sequential execution model toward a horizontally parallelized architecture fundamentally changed how the platform utilized infrastructure resources, while preserving the integrity of the existing COBOL application core. The COBOL modules themselves remained largely unchanged – retaining the same trusted, proven business logic of the platform.

These transformations were designed and validated by the same operational teams responsible for maintaining the platform daily, ensuring that execution integrity remained fully controlled throughout the transition.

Section 3. Results.

As the system evolved in stages over time, so did the business outcomes of the solutions. With six technology strategies, implemented over time, the platform – and the organizations it serves – has witnessed continuous improvement, innovation, and ever-improving quality of service.

Overall, the organization can boast tremendous improvements.

A. Simulation execution times, which previously required asynchronous execution and approximately thirty seconds to complete, were reduced to near-instantaneous responses for smaller workloads.
B. For larger customers, the combination of reduced intermediate data persistence and parallelized execution pipelines significantly transformed full calculation cycles.
C. Under optimized execution conditions, end-to-end processing times decreased from approximately eight minutes to around eight seconds.
D. The first production migration phase, involving the transfer of the operational platform from the traditional data center environment toward AWS infrastructure, was completed approximately six months ahead of the original schedule.
E. The resulting infrastructure flexibility allowed the platform to adapt more efficiently to workload variations throughout each monthly processing cycle.

These improvements were achieved but through a considered, incremental, progressive redesign of the execution architecture accumulated over many years.

Continuous Evolution

This story showcases a platform that continued evolving across multiple generations of infrastructure, execution models, and operational requirements, one which integrated important changes progressively over time in response to an ever evolving IT strategy.

What ultimately emerged was less a specific modernization project, and more a long-term software engineering discipline focused on preserving continuity while continuously rethinking and evolving execution models. It is the embodiment of the concept and value of continuous modernization.

---

www.cobolcloud.io

Section 4. Summary

CASE STUDY HIGHLIGHTS

Customer and Application Profile
A mission-critical calculation platform originally designed in the late 1970s, continuously operated and evolved for more than four decades, delivering production results for several million individuals across Europe. The system in question was a large-scale COBOL-based calculation engine combining online supervision and extensive batch processing, with strong requirements for determinism, auditability, execution consistency, and operational continuity.

Scope and Strategy
The platform is maintained and evolved by a dedicated European team responsible for daily production operations, maintenance, performance optimization, and functional adaptations driven by regulatory and business changes. Originally deployed on IBM mainframe infrastructure, the platform progressively evolved through distributed Unix and Linux environments before reaching cloud-ready execution models, without rewriting its core business logic.

Continuous Modernization and Architectural Evolution
The application stack was validated on AWS infrastructure, including ARM64 (Graviton) processors, allowing rapid adoption of new hardware architectures while preserving execution semantics, operational behavior, and business integrity. Execution models were progressively redesigned to reduce I/O pressure, minimize intermediate file exchanges, optimize data locality, and support horizontally parallelized execution pipelines adapted to elastic cloud infrastructure.

Measured outcomes
Simulation and production execution times were reduced from minutes to seconds, while maintaining deterministic execution and operational continuity. In parallel, the migration of the existing production workload from the data center to AWS was completed ahead of initial projections. A continuous, long-term evolutionary approach to application health has delivered multiple new improvements to successfully support business operations for decades, without any issues.

CobolCloud Partnership

Since 2013, CobolCloud (and its predecessor company) have supported the customer's gradual transformation from mainframe-based COBOL operations to a modern, cloud-ready environment. Rather than replacing core business logic, we helped preserve stability while enabling architectural evolution, including migration to distributed systems, AWS cloud adoption, and ARM64/Graviton portability. Their role has been collaborative and continuous, supporting customer-led re-architecture with software adjustments that improved performance, scalability, and flexibility without disruptive, high-risk transformation.