IBM

Efficient, low-risk modernization of core banking applications

Java alongside COBOL preserves the best of both the IBM zSystems and distributed worlds

by Tom Farre

7-minute read

Atruvia AG provides banking-as-a-service solutions and application development services to Germany’s cooperative banks. More than 800 banks rely on Atruvia’s agree21 core banking process and other applications along with its network of more than 30,000 ATMs and self-service terminals across the country.

With some 81 million customer accounts, the client banks generate a massive volume of transactions, which requires a powerful, resilient and security-rich IT infrastructure. Atruvia accomplishes this with eight IBM® z15® systems across four data centers that process 80 billion core banking transactions annually, with peaks of 12,000 transactions per second. The IBM Information Management System (IMS) manages the transactions and data is stored in the IBM Db2® for z/OS® database.

On the distributed side, customer engagement applications are implemented in Java®, either classically or as microservices, hosted on x86 Linux® systems on the Red Hat® OpenShift® container platform. Around 1,200 microservices support a sales platform and other processes, which consume core banking services via a proprietary API layer.

Historically, Atruvia implemented core business transactions with Java/HTML5 at the front end connected to a mid-tier layer in plain Java and IMS COBOL at the back end. With many COBOL programmers retiring

"We see Java on IBM zSystems as a key technology in driving competitive advantage for our clients."

Pascal Meyer, Senior Enterprise Architect, Atruvia AG and banking customers expecting fast delivery of new omnichannel services, Atruvia executives decided on a creative approach to application modernization that would maintain the security and performance of IMS while offering the advantages of a popular programming language.

Rather than a high-risk and costly "lift and shift" strategy, their solution was to selectively refactor in place. They would bring the latest version of Java into the existing IMS/COBOL runtime on the IBM zSystems® platform for incremental app modernization.

"Our core banking applications are permanently in evolution," explains Pascal Meyer, Senior Enterprise Architect at Atruvia. "For requests that don't justify full reimplementation, we wanted to introduce Java into the IMS COBOL environment. A key objective was to make our banking applications

Using the Java in IMS capability on the IBM zSystems platform, Atruvia has Java-enabled

85%

of core banking transactions

Colocating some data-intensive distributed Java applications with IBM zSystems results in

3 times

faster performance

Java alongside COBOL within IMS

boosts

application developer productivity

available in a way that would be more familiar to the latest generation of developers."

Adding Java to IMS alongside COBOL could also reduce application complexity. Atruvia's classic architecture makes it necessary to include "compensation" logic within every component to ensure that operations distributed across the various components can be rolled back to the last consistent state if an error or interruption occurs. Bringing Java code inside the IMS environment would make the extra logic superfluous, thereby reducing complexity, costs and delays.

"In terms of the application architecture, Java opens up the possibility of using a broader range of frameworks and protocols," adds Thomas Bauer, Team Leader and IT Architect at Atruvia. "For example, RESTful web services are more natural in Java than in COBOL, and certain functions that are difficult to implement in a pure COBOL environment are available as prebuilt artifacts in Java. We wanted to cut time-to-market by deploying modern, reusable software components—and, at the same time, protect our investment in existing business logic by Java-enabling it."

As Java was already a strategic platform at Atruvia, the organization was keen to enable greater portability of its code across both distributed and IBM zSystems platforms—whichever offered the best price-performance. This approach can also ease the transition to cloud computing. Since OpenShift runs on all platforms, Atruvia can move workloads to the cloud when that makes sense.

Plus, Java on IBM zSystems alongside COBOL would enable developers to enrich core banking functionality in a seamless, low-risk manner by replacing COBOL subroutines with Java without having to rewrite large programs. And it would be easier for software architects within the distributed environment to call core transaction services directly from IMS applications.

"In the longer term, our goal is to be more platform-independent by having Java building blocks that can run anywhere," says Meyer. "Besides reducing architectural complexity, mixing COBOL and Java inside IMS would enable us to build new application components tightly integrated with existing components. In this way, we can gain the advantages of a modern hybrid app—namely, short time-to-market, increased flexibility and greater ease of development—without losing the performance benefit of IMS where it really counts in core banking transactions."

Setting the stage for creative application modernization

Atruvia personnel worked closely with IBM Systems developers in a multiyear project to optimize a common runtime for Java constructs within IMS production environments—before the common runtime was generally available. Based on IBM Semeru Runtime Certified Edition for z/OS, Version 11, the common runtime is now standard within the latest release of IMS. And IBM and Atruvia are committed to supporting the latest Java version.

The common runtime has enabled Atruvia to move toward a 64-bit future

by allowing 31-bit COBOL applications to communicate with 64-bit Java applications. With Java and COBOL interoperable within IMS, the company can modernize and revitalize its core banking applications without impacting performance or reliability.

In practical terms, Atruvia focuses on two objectives for Java on the IBM zSystems platform. First, developers are building hybrid Java-COBOL applications for classic IMS processing and batch workloads that invoke IBM MQ for z/OS and connect with IBM Db2 for z/OS through Java database connectivity (JDBC). Second, developers are migrating native Java code from the distributed environment to IBM zSystems when appropriate.

“Many of our batch jobs were spread across multiple platforms on both IBM zSystems and the distributed environment, with different schedulers that we needed to coordinate,” says Meyer. “This approach also required data exchange and/or data sharing. Today, by executing Java jobs directly on IBM zSystems, we need only a single scheduler so there is less complexity. We also achieve better performance because there is no need to move or convert data, and because the Java code sits right next to its data rather than needing to access it over a network.”

Atruvia is also building a set of generic Java services that can be exposed as APIs to developers of front-end applications running in its distributed environment. For example, a developer could use an API to call up the current balance of a customer’s checking account for display in a mobile app. And adding Java to the IBM zSystems platform makes it easier to present new services based on existing functionality.

“Developers in the distributed world will not know if they are calling Java services or original IMS transactions,” says Meyer. “Everything will be accessed in the same easy and consistent manner, making it faster and simpler to build new front-end applications that call the robust underlying transactions on IBM zSystems. This supports our banking clients as they look to reach out to their employees and customers with innovative services delivered through web, mobile and whatever new channels may emerge in the future.”

Java within IMS also gives Atruvia the option to migrate older COBOL code, though developers do so selectively, such as when building new business logic. "There is no pressure to modernize just for the sake of modernization," explains Bauer. "We continue to value the performance and robustness of COBOL and IMS at the back end, and we continue to plug those technologies into graphical front ends running on distributed systems."

In this way, the IBM zSystems platform running IMS continues as the focal point for business logic. It feeds directly into a variety of channels for consumption of that logic, such as teller applications, ATM systems, customer-facing web applications and mobile apps.

"The IBM zSystems platform running IMS remains our strategic choice because it is the most secure and reliable platform that we know of, and because it offers exceptional performance for our core banking systems. With Java in IMS, we have the best of both worlds."

Pascal Meyer, Senior Enterprise Architect, Atruvia AG

Faster, simpler and more sustainable

Due to the pioneering collaboration between IBM and Atruvia, Java is now an established, production-ready component of the IBM zSystems toolbox. This significantly refreshes the IMS environment, preserving its traditional qualities of performance and robustness while enabling faster development using more widely available and sustainable programming skills. It also allows developers to enrich existing core banking functions in a low-risk, efficient manner.

“We see Java on IBM zSystems as a key technology in driving competitive advantage for our clients,” says Meyer.

“With easier development and reuse of existing components, it enables us to deliver new functionality at higher speed and lower cost, supporting our banking clients as they seek to bring new services to market more rapidly.”

In fact, Atruvia has already Java-enabled around 85% of its core banking IMS transactions—some 400 million Java transactions per day with peak throughput reaching 12,000 transactions per second. Twelve IMS systems with approximately 200 million instructions per processor second (MIPS) support these business-critical transactions.

By reducing the complexity of the application environment, Java within IMS has increased the efficiency and performance of end-to-end business transactions. Before, enterprise batch processing was handled by a spread of multiple platforms. Now, all processing, regardless of the language, can simply be integrated into a single batch step by combining COBOL and Java.

Additionally, Atruvia developers can easily port Java code across its distributed and IBM zSystems environments, whichever platform offers the best price-performance. For example, Atruvia has reduced latency by colocating some distributed Java workloads to the IBM zSystems platform. "After all, the best I/O is no I/O," quips Meyer. He reports threefold performance gains from colocation of some data-intensive apps instead of running them across the network. Clearly, a wholesale migration to the cloud is not the only path to application modernization.

In addition, there are financial incentives to move Java workloads from general processors (GPs) to IBM Z Integrated Information Processors (zIIPs), on which the licensing fees are lower. Although the MIPS required increase—because Java consumes more resources than COBOL—the overall costs are reduced.

Other efficiencies come from rich Java libraries that let programmers avoid writing custom code for common functions such as compressing and decompressing data. Similarly, Atruvia can integrate third-party Java software into its IBM zSystems environment, potentially avoiding in-house development altogether. And as new hardware-based cryptographic, compression or networking features become available on the platform, Java for IBM zSystems will give applications transparent access to the new capabilities.

"As a provider of core banking systems to more than 800 banks with tens of millions of end users, we cannot afford to compromise on performance and reliability, nor can we afford to introduce risk in our development practices," says Meyer. "The IBM zSystems platform running IMS remains our strategic choice because it is the most secure and reliable platform that we know of, and because it offers exceptional performance for our core banking systems. With Java in IMS, we have the best of both worlds."

About Atruvia AG

Formed by the 2015 merger of Fudicia IT AG and GAD eG, Atruvia (external link) is the digitization partner of Germany's 820 cooperative banks. Its banking IT solutions and services range from data center operation and the agree21 banking-as-a-service process to app development and support. With administrative headquarters in Karlsruhe and Münster and branches in Munich, Frankfurt and Berlin, Atruvia employs almost 8,400 people and reported group sales of around EUR 1.77 billion in 2020.

Solution components

- IBM Db2® for z/OS®
- IBM IMS Solutions for Java Development
- IBM Information Management System (IMS)
- IBM MQ for z/OS
- IBM Semeru Runtime Certified Edition for z/OS, Version 11
- IBM zSystems®
- IBM Z Integrated Information Processor (zIIP)
- IBM z/OS
- Red Hat® OpenShift®