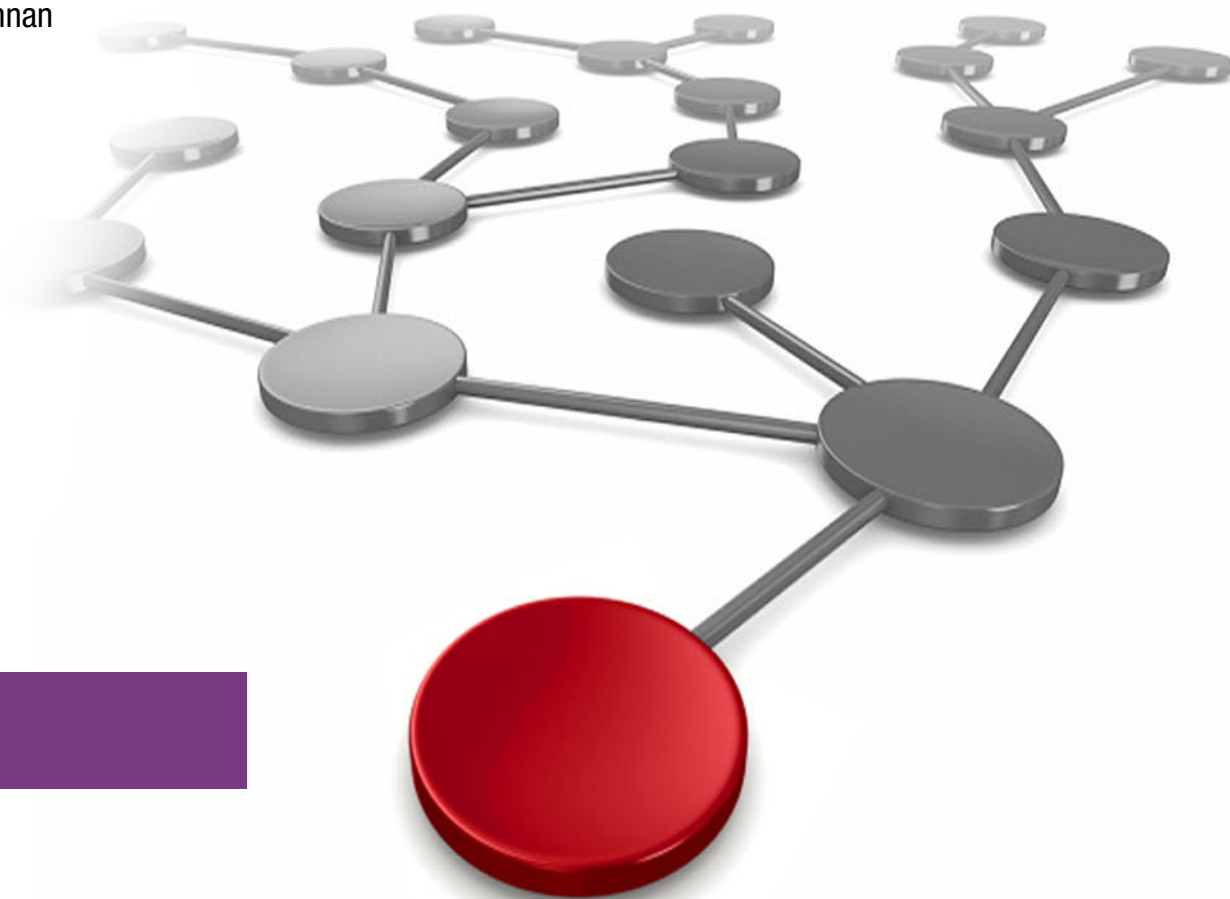


Mainframe Batch Modernization and Transformation

Pete McCaffrey

Mythili Venkatakrishnan



IBM Z

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

CICS®


Db2®

IBM®

IBM Consulting™

IBM Z®

Redbooks®

Redbooks (logo) ®

z/OS®

The following terms are trademarks of other companies:

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.



Mainframe batch modernization

Batch processing is used for some of an organization's most critical and time-sensitive business operations. It is used across many industries including, memo-posting of banking transactions in high volume, settlement and clearing, claims processing, risk assessments, payroll handling, ledger postings and reconciliations, and operational activities such as backups, reporting, data transfer, and archival. Batch jobs perform many read, write, and sort activities on sequential and VSAM files without manual intervention or input required unless, of course, a job does not end successfully. Batch window processing often takes place during off-hours or non-peak hours, and it is not unusual for OLTP to wait on the completion of the batch processing, as OLTP start of day (input) often requires the results of batch processing.

Within the enterprise domain, there are various applications that require bulk processing to perform critical business operations. These operations involve:

1. **Automated Processing:** Complex processing of large amounts of data that does not require user interaction. This includes time-based events such as daily, weekly, and month-end calculations, notices, or correspondence.
2. **Periodic Business Rules:** Application of repetitive and complex business rules across extensive datasets. Examples include insurance benefit determination or rate adjustments.
3. **Information Integration:** Processing and integration of data from internal and external systems, involving formatting, validation, and transactional processing into the system of record. Batch processing is commonly used to handle billions of transactions daily in enterprises.

While an ideal IT environment may seem to solely rely on online transaction processing (OLTP), where user actions immediately trigger updates and deliver instant output, there are functions in an information system that are not suitable for OLTP. Therefore, during the design phase of an information system, it is crucial to clearly identify functions that should be implemented as batch processes versus real-time processes.

This IBM Redpaper publication examines varieties topics regarding batch modernization:

- ▶ "Batch scheduling and processing"
- ▶ "Common batch use cases"
- ▶ "Designing an application function as a batch process"
- ▶ "Batch analysis and assessment"
- ▶ "Batch optimization"
- ▶ "Java batch on the mainframe"
- ▶ "Common batch use cases and modernization techniques"

- ▶ “Moving from batch to more real-time for business processing workflows”
- ▶ “Modernizing loan processing to be more real-time”
- ▶ “Summary”

Batch scheduling and processing

In a z/OS environment, dedicated subsystems (JES2) in combination with special job scheduling software, such as [IBM Z Workload Scheduler](#), manage batch processing.

The *Job Entry Subsystem (JES)* is a component of IBM's z/OS mainframe operating systems that is responsible for managing batch workloads.

Job processing in a system is typically divided into several phases to enable parallelism through pipelining. These phases include:

- Input Processing: In this phase, jobs are read and interpreted by the system. This involves parsing job control statements, allocating resources, and scheduling the jobs for execution.
- Execution: Once a job is scheduled, it enters the execution phase, where it runs and performs the specified tasks. This phase involves executing the program or batch commands specified in the job control statements.
- Output Processing: After a job completes its execution, the output generated by the job is processed. This can involve storing the output on Direct Access Storage Devices (DASD) for future reference or generating reports.

During each phase, jobs are organized into queues based on their current status. For example, jobs that are currently executing are placed on the execution queue. These queues, including execution queues and other relevant queues, are stored in the JES (Job Entry Subsystem) multi-access spool (MAS). The MAS is a shared storage area accessible by all systems in the Sysplex (a group of interconnected mainframe systems).

To protect against unexpected hardware or software failures, JES uses a mechanism called a checkpoint. A checkpoint backs up essential information about currently executing jobs and their associated output. If a failure occurs, the checkpoint can be used to restore the jobs and their output, ensuring that progress is not lost and allowing for recovery.

IBM Z Workload Scheduler is a powerful workload automation solution that enables organizations to automate, plan, and control complex systems' workloads. It provides centralized management of workflows across multiple platforms and business applications, offering a single point of control. With the ability to adjust real-time workflow execution based on business needs, it leverages modern technologies to optimize processing.

In many batch processes, there are multiple critical paths comprised of dependencies between batch jobs leading to critical end points. IBM's [Workload Manager](#) (WLM) works in conjunction with IBM Z Workload Scheduler to effectively manage these scenarios. WLM allows for the classification and prioritization of late-running critical path batch jobs, ensuring they are assigned higher priority access to CPU resources. This capability ensures that critical batch processes receive the necessary resources to meet their deadlines and maintain optimal performance.

Common batch use cases

Examples of batch processes commonly used in industries today include:

- ▶ Core industry specific processes including memo-posting of banking transactions, settlement and clearing, claims processing, risk assessments, ledger postings and reconciliations
- ▶ Generating reports of all daily processed data
- ▶ Printing or sending bi-weekly account statements to all customers of a bank
- ▶ Paying salaries for all employees
- ▶ Archiving historical data at the end of month
- ▶ Optimizing databases
- ▶ Extract, Transform, Load (ETL) data
- ▶ Creating backups of files and databases, for Disaster Recovery purposes
- ▶ Processing files with large amounts of data from business partners

In business process-oriented batch processing, application programming logic is commonly implemented, often using programming languages like COBOL. However, it is important to note that not all batch jobs require an application program. Many batch processes involve the use of utility programs that perform specific tasks such as data extraction, sorting, formatting, database reorganization, backup creation, file replication, or data transmission.

Bulk processing of large transaction volumes during non-office hours continues to be a valuable and strategic approach. This type of processing often involves integrating external data feeds, which need to be reconciled in the correct sequence with online transaction processing (OLTP) activity to generate consistent end-of-day valuations and balances. By leveraging batch processing, organizations can efficiently handle the high-volume data processing required for these reconciliation and consolidation tasks.

Designing an application function as a batch process

Reasons to design an application function as a batch process include:

- ▶ **Dependency on Unavailable Resources:** If real-time processing of the business logic is not possible due to dependencies on information / resources that are not available at that time, the transaction can be captured and stored for later processing in a batch program. The results of the transaction can still be reflected in real-time inquiries, even if the final processing and reconciliation occurs during batch execution. An example of this is memo posted core deposits processing in banking.
- ▶ **Repetitive Processing:** Batch processing is efficient when a large amount of repetitive processing needs to occur. Calculations that involve the same static data elements, such as tax percentages used for employee salary calculations in a large company, can be performed more efficiently in batch rather than as separate OLTP transactions.
- ▶ **Large Data Scanning:** When scanning large data files or database tables sequentially, batch programs with asynchronous read-ahead functions are more efficient than synchronous OLTP processing, which typically reads small data blocks.
- ▶ **Deferral of CPU Cycles:** Batch processing allows for the deferral of CPU cycles to times when the system is not heavily used by online users. For example, preparing printed invoices for orders placed during the day can be done during off-peak hours.

- **Building Periodic Interface Files:** In industries like banking, there is a frequent need to create interface files with logical groupings of records to be sent to partner information systems. Sending one consolidated file with many records periodically can be more efficient and reliable than sending each record in real-time.

By utilizing batch processing for these scenarios, organizations can optimize resource usage, improve efficiency, handle large data volumes, and meet specific timing requirements for various business operations.

Batch analysis and assessment

IBM offers a range of Batch analysis and assessment tools as well as expertise to help businesses optimize mainframe MIPS (Million Instructions Per Second) usage for Batch processing. These tools and services are designed to assist clients in identifying inefficiencies and fully leveraging the capabilities of their systems to optimize performance and reduce the cost of batch processing.

One such tool is the [IBM Z Performance and Capacity Analytics](#) which deliver insight for IT Operations Managers, Performance Experts and Capacity Planners to make informed decisions about their infrastructure and application performance including batch.

Leveraging SMF and other structured data sources on IBM Z and other platforms, enterprise-wide IT utilization information is curated quickly and efficiently into actionable reports that can be leveraged to identify opportunities for batch improvements.

IBM Z Performance and Capacity Analytics can be used in combination with [IBM Application Discovery and Delivery Intelligence \(ADDI\)](#) to pinpoint batch applications with high resource consumption and execute an application discovery process to identify and determine the impact of changes with full understanding of dependencies and risk.

With millions of lines of code, hundreds of dependencies, and dated documentation, developers can spend weeks or months trying to understand all the changes needed with no guarantee the updates won't break something. Discover dependencies in a click, make changes with confidence, and keep your documentation current and accurate. With ADDI, you can expect developer productivity to increase by 20% to 30%.

The goal of these IBM offerings is to help businesses meet their target window for running batch processing, optimize mainframe MIPS utilization, and achieve better performance and cost efficiency in their batch operations.

Batch optimization

IBM has made significant investments and innovations in the IBM Z platform, making it highly regarded in the industry for batch processing. These ongoing investments enable clients to optimize costs associated with existing applications and the deployment of new or modernized applications.

By leveraging current software and hardware, clients can take advantage of various performance optimization tools, specialty processors, and consumption-based pricing models that provide a more cloud-like experience.

IBM z/OS serves as the foundation for batch processing, with a strong track record of excellent I/O performance. When combined with IBM z/OS Workload Manager (WLM),

specific prioritizations can be applied to further optimize I/O operations. High-speed compression technology reduces the amount of data read and written by batch jobs, and high-performance disk and tape technology accelerates I/O activities. Moreover, optimized database design techniques in IBM Db2 and IMS, along with the use of VSAM Record Level Sharing (RLS), allow for improved batch concurrency with online transaction processing (OLTP).

IBM continues to enhance and optimize programming languages like COBOL and PL/I to fully exploit the capabilities of the IBM Z architecture. These programming languages are supported by tools that mitigate risks and enable modern programming techniques, facilitating interoperability with other languages such as Java.

[IBM Automatic Binary Optimizer for z/OS \(ABO\)](#) is an advanced optimization technology designed to optimize COBOL modules without requiring source code recompilation. It is specifically built to support the latest IBM Z systems and offers significant benefits in terms of reducing processing time, CPU usage, and operating costs associated with running critical COBOL batch applications.

ABO optimizes COBOL modules directly, resulting in improved performance while retaining strict compatibility with the original modules. The optimized modules behave exactly the same way as the originals but consume fewer system resources, leading to cost savings.

When used alongside the latest IBM Enterprise COBOL compiler for z/OS, ABO becomes a complementary tool. Organizations can leverage the COBOL compiler during active application development or maintenance phases to optimize specific parts of the application. ABO, on the other hand, can be employed to optimize the remaining parts, maximizing the return on investment for IBM Z systems.

By employing ABO and the latest COBOL compiler, organizations can enhance the performance, efficiency, and cost-effectiveness of their COBOL batch applications running on IBM Z, without the need for extensive source code changes or recompilation.

Overall, IBM's investments, optimizations, and comprehensive toolset provide clients with the means to effectively address cost and performance challenges in batch processing and maximize the benefits of their mainframe deployments.

Java batch on the mainframe

In this section, we outline some of the benefits of using Java batch.

IBM mainframes support and execute batch processes written in the Java language. There are various motivations for leveraging Java Batch, particularly when there is a need to transform existing batch processing or accelerate the transition to more real-time processing.

Java Batch provides an additional option for modernizing batch operations and can enable the integration of more real-time processing capabilities. Additionally, the availability of Java programmers in the market is generally higher compared to COBOL programmers, allowing organizations to tap into a larger talent pool.

There are 2 significant advantages of using Java Batch. One is the ability to leverage the lower-cost specialty processors known as zIIPs (IBM Z Integrated Information Processor) to help manage the costs associated with running batch processes in parallel with online transaction processing (OLTP) workloads. Another advantage is that running batch intra-day on zIIP processors can shorten the batch window processing required by preparing relevant

information / data during the day without impact to OLTP which typically has limited use of zIIPs.

Depending on the specific scenario, Java can deliver performance comparable to, or in some cases faster than, compiled languages. Since batch processing is iterative in nature, Java classes are well-suited for optimization through the Just-in-Time (JIT) compilation that occurs at runtime. The IBM Z processor chips have instructions specifically designed to assist JIT-compiled code, providing additional performance benefits. In contrast, COBOL code that has not been recompiled for a long time may not fully utilize these chip instructions, making it less optimal in terms of performance compared to Java in certain scenarios.

Overall, leveraging Java Batch on IBM mainframes offers opportunities for modernizing batch processes, utilizing a larger talent pool, optimizing performance through JIT compilation, and making cost-efficient use of specialty processors for batch and OLTP workloads.

Getting started with Java Batch

Java batch on z/OS is facilitated by [JZOS Batch Launcher and Toolkit](#).

The JZOS Batch Launcher is a native launcher for running Java applications directly as batch jobs or started tasks. Java applications can be fully integrated as job steps to augment existing batch applications.

The JZOS Toolkit is a set of Java classes that give Java applications direct access to traditional z/OS data and key z/OS system services.

Here are some of the facilities and features provided by the JZOS Batch Launcher and Toolkit:

- ▶ Run Java applications on z/OS seamlessly in a MVS batch job step or started task.
- ▶ Simple but flexible configuration of the Java execution environment.
- ▶ Access to data sets via JCL DD statements.
- ▶ Java interfaces to many z/OS specific APIs and features including SMF, Catalog Search and Logstreams.
- ▶ Invoke z/OS Access Method Services (IDCAMS).
- ▶ Read and write traditional MVS data sets from Java.
- ▶ Communicate with the MVS system console.
- ▶ Pass condition codes between Java and non-Java job steps.
- ▶ Access z/OS Workload Manager (WLM) services.

Running Java on IBM z/OS as batch jobs has been made easier through various enhancements. The combination of the launcher, data access capabilities, system services, and environmental improvements has simplified the process, specifically benefiting Java application developers.

With these enhancements, managing Java batch jobs on z/OS is now like managing batch applications written in other compiled languages like COBOL or PL/I. This consistency in management processes allows for seamless integration and operation within the broader z/OS batch processing environment.

Moreover, Java Interlanguage Batch (JIB) provides support for interoperability between COBOL, PL/I, and Java, enabling a unified approach to managing batch and ensuring the smooth execution of batch jobs across different programming languages. In cases where

there is interaction with Db2 for z/OS, additional features such as maintaining transactional data integrity and rollbacks across a unit of work are possible.

Overall, the advancements in running Java as batch jobs on z/OS, along with the interoperability support offered by JIB, provide application developers with the necessary tools and capabilities to effectively manage and integrate Java batch processing within the z/OS environment, enabling reliable and streamlined execution of batch jobs.

Using Jakarta or Spring Batch frameworks

Both Jakarta Batch and Spring Batch are popular frameworks that support Java Batch on the mainframe, providing developers with a set of predefined code and features to expedite and simplify Java batch application development.

Jakarta Batch offers a comprehensive specification that includes an XML-based job specification language (JSL), a Java programming model, and a runtime environment tailored for batch applications on the Java platform. It enables developers to compose jobs using XML and Java application artifacts, promoting reuse across different jobs. Key features of Jakarta Batch include checkpoint and restart functionality, step composition for reuse and restart ability, XML configuration for externalized configuration, and support for parallel processing of data through partitions.

Spring Batch, on the other hand, is a lightweight and comprehensive batch framework that allows developers to build robust batch applications for enterprise systems. It leverages the productivity and ease of use associated with the Spring Framework, providing features such as logging, transaction management, job processing statistics, job restart, skip logic, and resource management. Spring Batch supports both simple and complex use cases, from basic file-to-database tasks to high-volume data processing and transformation. It also offers advanced optimization and partitioning techniques for efficient processing of large volumes of data.

It's important to note that while both frameworks provide powerful batch processing capabilities, they have some distinctions. Jakarta Batch focuses on defining the specification for batch processing with features like XML-based job specifications and partitioned processing. Spring Batch, on the other hand, builds upon the Spring Framework's characteristics and includes additional enterprise services for batch processing. Additionally, Spring Batch does not include scheduling capabilities but can be used alongside a scheduler.

Overall, both Jakarta Batch and Spring Batch serve as valuable tools for Java developers working on mainframe batch processing, offering pre-existing code and features that accelerate development, enhance productivity, and ensure reliable execution of batch applications.

Common batch use cases and modernization techniques

In this section we will cover the following topic:

- ▶ “Modernizing batch reporting”
- ▶ “IBM migration utility”
- ▶ “Transition traditional ETL to Read-Transform-Serve”

Modernizing batch reporting

Report generation is a common batch use case, and there are a variety of report generation tools for the mainframe. There are often cost of ownership considerations and functional limitations in some of these tools that cause businesses to evaluate other options.

In one example, IBM provides a [Migration Utility](#) for converting report generation to leverage IBM mainframe COBOL. The benefits are:

- ▶ COBOL I/O handling is more efficient
- ▶ COBOL sorting and searching is more efficient
- ▶ COBOL better coexists with other languages and environments
- ▶ All COBOL debugging tools can be used for debugging
- ▶ More people are available for program support
- ▶ You can save money by eliminating licensing costs for separate report generation software.

IBM migration utility

The IBM Migration Utility give you the following choices:

- ▶ You can continue developing programs using your existing report generation format. The only thing that changes is JCL. That is, you use the Migration Utility translator and COBOL compiler in the place of report generation.
- ▶ You can convert the report generation programs to COBOL. You can convert existing and newly developed programs. After converting, you maintain COBOL code.

IBM Consulting™ also offers a service for migration of report generation to COBOL.

Transition traditional ETL to Read-Transform-Serve

ETL, which stands for extract, transform, and load, is a data integration process used to combine data from various sources into a consistent data store, typically a data warehouse or target system. It originated in the 1970s alongside the rise of databases and has become the primary method for processing data in data warehousing projects.

ETL plays a crucial role in data analytics and machine learning workflows. By applying a set of business rules, ETL cleans and organizes data to meet specific business intelligence requirements, such as monthly reporting or advanced analytics. It improves data quality, establishes consistency, and can enhance back-end processes and user experiences.

Typically, organizations employ ETL to:

- ▶ Extract data from mainframe systems
- ▶ Cleanse the data to enhance its quality and ensure consistency
- ▶ Load the data into a target database

As the volume of data processed through ETL has grown, it has led to increased processing costs and potential security vulnerabilities when data is moved off-platform. There is an opportunity to optimize performance and reduce ETL requirements by leveraging mainframe data for inquiry, analytics, and reporting without the need to transfer it away from the platform. Performance can be enhanced by streamlining information requirements and utilizing existing

tools, technology, and utilities. This approach minimizes processing costs and potential security risks associated with moving large amounts of data.

Moving from batch to more real-time for business processing workflows

As business accelerate their digital initiatives to better address customer expectations there is a requirement to execute core business processes increasingly in real-time. This includes processes like loan origination and approval, clearing and settlement, payments authorization, fraud detection, inventory management, asset management, transportation booking to name just a few. Today, many of these processes are executed through nightly batch processing. Business demands are driving the need for more frequent updates throughout the day in parallel with OLTP and more visibility to intermediate information trending toward real-time.

There are resource and cost optimization challenges associated with more frequent batch processing concurrent with OLTP. Application dependencies and parallel access to and update of the same data in both OLTP and batch can create resource contention. In addition, dependencies in the workflow where batch depends on input from OLTP and vice versa need to be evaluated to ensure that any optimizations introduced align with business requirements.

We will explore strategies where you can use a variety of techniques for optimizing batch and reducing contention like accessing near real-time information for inquiry only transactions, complex queries, analytics, and reporting. Some technology enablers include [IBM Z® Digital Integration Hub](#), [IBM Db2 Analytics Accelerator](#) and Java Batch.

IBM Db2 Analytics Accelerator (IDAA)

IBM Db2 Analytics Accelerator is a high-performance component tightly integrated with Db2 for z/OS®. It delivers high-speed processing for complex Db2 queries to support business-critical reporting and analytic workloads. The accelerator transforms the mainframe into a hybrid transaction and analytic processing (HTAP) environment.

It drives out cost and complexity and enables analytics on transactional data as it is generated. This can reduce the need for batch reporting while providing the more real-time access to transactional data without impacting performance of the OLTP environment.

IBM Z Digital Integration Hub

[IBM Z Digital Integration Hub \(zDIH\)](#) provides a real-time flow of information between core business systems, which are hosted on the IBM Z platform (z/OS), and consuming applications, which are hosted on hybrid cloud. IBM zDIH offers optimized performance and minimal impact to the core applications when applications need real-time mainframe information at scale.

IBM zDIH includes highly performant in-memory caches that use a Java-based runtime environment. It includes a Developer Kit that auto-generates the caches and applications for accelerated low-code adoption. IBM zDIH comes with templates, and samples to accelerate the integration with IBM CICS® and IMS applications through application exits combined with log streams on IBM z/OS, direct writes from the applications, or by using IBM MQ on z/OS.

IBM zDIH runs predominantly on IBM zIIP processors and helps optimize costs through separating inquiry workload from update traffic (CQRS) and reducing the need for core business applications to recompute information that has not changed in response to inquiries. In addition, unpredictable or spiky inquiry traffic can be handled with efficiency and without impact to your core business applications.

IBM zDIH can directly flow information to enterprise-wide architectures for event processing (through Kafka), API management, or mediation on public cloud. Kafka topics can be configured to be updated after any information in zDIH changes. Those topics can be used by public cloud (SaaS) solutions that derive significant value from real-time characteristics, and potentially improve the time to value of cloud-native applications.

Example business use case: Clearing and settlement

Today, most financial institutions perform clearing and settlement as part of an overnight batch process. Increasingly, there is a requirement for more frequent “intra-day” clearing and settlement to reduce risk or achieve regulatory requirements. One approach is to run batch processing for clearing and settlement during the day in parallel to OLTP – however, there may be performance challenges due to resource and dataset access contention. In addition, it may drive additional mainframe general purpose capacity to run both batch and OLTP at the same time.

Depending on the implementation, leveraging IBM Z Digital Integration Hub (zDIH) can be part of a solution architecture that helps you achieve your objectives for accelerated and optimized Clearing and Settlement.

One option, depending on volumes, is to read input of relevant batch files into memory. The logic to perform the settlement could be built using the in-memory caches for manipulation and avoid intermediary I/O to files, or aspects of the logic can be incorporated into the zDIH applications. The information could then be made available for inquiry through the day, but then also persisted to disk for post processing.

Another scenario is if a client implements zDIH but there is still a need for some batch processing, for example a reliance on 3rd party files. You can create a Java Batch program that uses zDIH as a data source along with other data sources needed to complete the batch processing. You would simply use JDBC or standard Java interfaces for the zDIH caches. You run this batch job in parallel to the OLTP window but since it's going against zDIH instead of the core business systems you avoid potential performance issues and optimize costs since zDIH is zIIP eligible and the production datasets are not impacted.

For the Clearing / Settlement use case, you need to first understand what information is needed to process, how much data, what transforms are required, and what the overall processing flow for the clearing/settlement. There could be a zDIH Java application that reads input batch files periodically into the zDIH caches. Then you would need to add the settlement logic to a Java batch application which can do the processing using the in-memory zDIH caches and persist output to disk as needed. Another option is that the zDIH applications could read input data and perform basic transforms while populating the in-memory caches to further optimize processing for any subsequent Java batch jobs.

In Figure 1, we illustrate optimizing intra-day batch with zDIH.

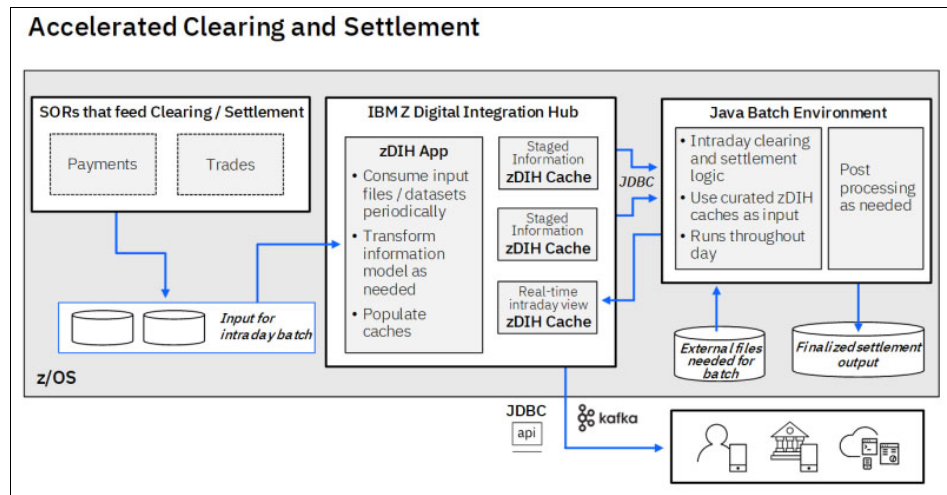


Figure 1 Accelerated Clearing and Settlement

Example Business Use Case: Loan Processing

In Figure 2, we use an example of a typical Batch architecture. This typical scenario shows a mainframe system using z/OS CICS, JCL, Cobol, Db2®, VSAM files, GDG files, and tapes.

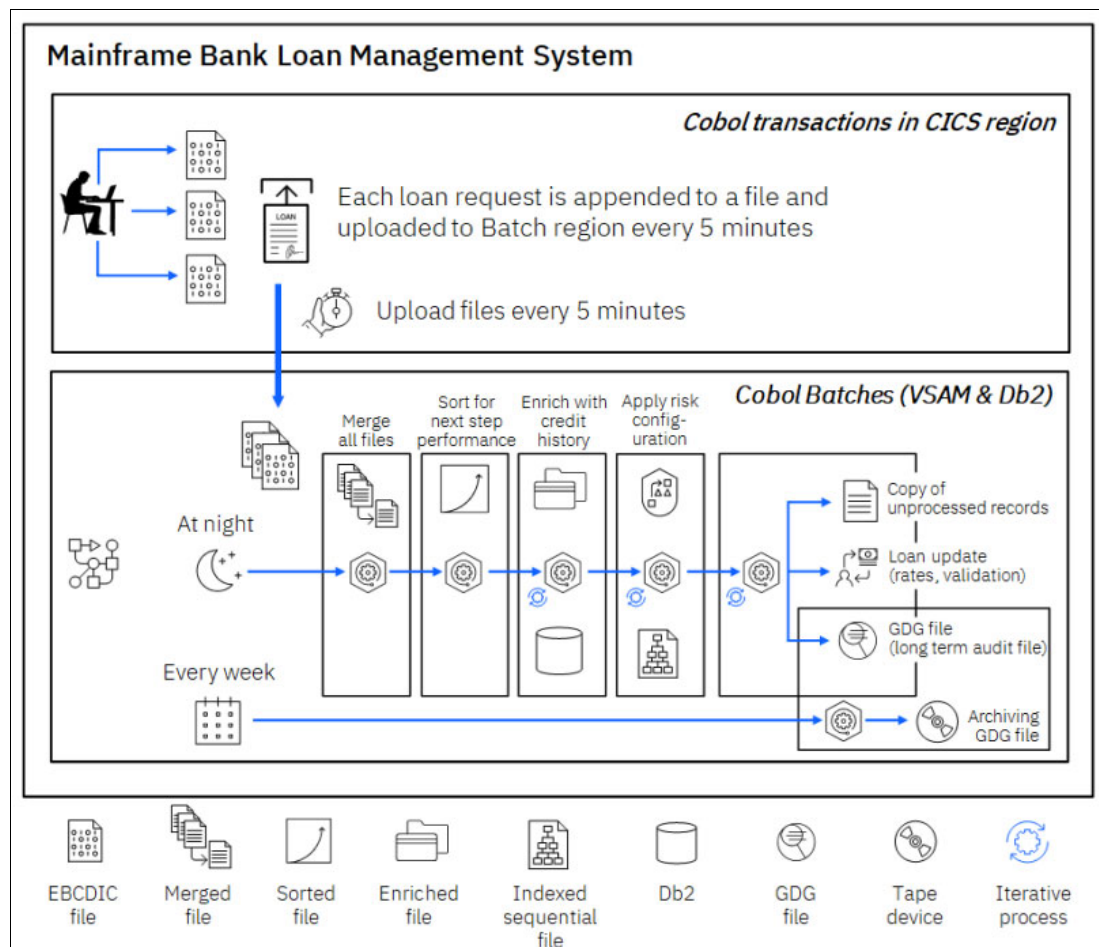


Figure 2 Mainframe Batch architecture example.

The Loan processing batch workflow described consists of three programs with specific functionalities:

1. Upload Batch Program: This program runs every five minutes and sends temporary files generated by loan request transactions to the Batch region using message queuing. Its purpose is to transfer the data from different physical offices to the central batch processing system.
2. Loan Batch Program: This program executes nightly and performs the following steps:
 - a. Merge: Combines all temporary files into a single file for further processing.
 - b. Sort: Sorts the merged file to improve performance and facilitate subsequent processing steps.
 - c. Enrichment: Retrieves additional information from Db2 to enhance each record in the sorted file. This includes credit history and other loan-related details, which are injected into the enriched record.
 - d. Risk Assessment: Enhances each record further by injecting risk assessment information from a VSAM file. This results in an enriched file containing all the required data for performing loan request risk analysis.
 - e. Processing: Processes each record in the enriched file, generating three outputs:
 - Unprocessed/Rejected Records: Contains copies of records that require further processing due to parsing errors, missing elements, or other failures.
 - Db2 Table Updates: Updates the records for each customer requesting a loan in a Db2 table, indicating the current status of the loan request (rejected, approved, pending). For approved requests, it also includes loan proposal details such as rates and duration.
 - Audit Information: Traces who, what, when, and where details into mainframe Generation Data Groups (GDG) files, providing an audit trail for the batch processing operations.
3. Archiving Batch Program: This program runs weekly and serves two purposes:
 - Tape Backup: Saves selected GDG data to tape devices for legal reasons, ensuring data preservation and compliance.
 - Pruning: Removes GDG files that have been successfully archived to free up storage space and maintain a manageable dataset.

Overall, this batch processing workflow manages loan request data throughout the day, consolidates and processes it during the night, and performs archival tasks periodically. It enables efficient loan processing and data management while minimizing concurrency, locks, and consistency issues between batch processing and online transactions.

Modernizing loan processing to be more real-time

The proposed solution architecture for more real-time loan processing leverages Java Batch and IBM Digital Integration Hub (zDIH) to enable efficient and parallel processing. Here's an overview of the suggested approach:

1. Real-time Processing: Instead of waiting for end-of-day processing, loan requests are processed intra-day as they come in. The frequency of processing can be determined by the business requirements.
2. Java Batch Application: A Java Batch application is developed to handle the intra-day loan processing. This application can periodically read the incoming loan request file and

initiate a zDIH batch job to begin processing. By utilizing Java Batch, the application can manage the orchestration and coordination of the loan processing tasks.

3. **Enrichment with zDIH:** As part of the loan processing, zDIH is utilized in a continual run approach to retrieve enriched personal information, such as credit history and other loan details, that has previously been loaded into zDIH. This data can be accessed during the processing phase to enhance the loan request records.
4. **Risk Assessment:** Additional risk assessment information can be incorporated into the loan processing either through batch processes or through zDIH calculations. This further enhances the loan evaluation and decision making process.
5. **Parallel Processing:** If there is no dependency or need for serial processing, the Java Batch program can initiate multiple threads to process the loan request records in parallel. This allows for faster processing and improved scalability.
6. **Output Generation:** Once the loan processing is completed, the Java Batch application creates the necessary output files based on the processing results. This may include files for further processing, updates to relevant databases, loan approval details, and audit information.

By adopting this solution architecture, organizations can achieve efficiency and more real-time loan processing while leveraging the capabilities of Java Batch and IBM Digital Integration Hub. This approach enables parallel processing, enhances loan evaluation through data enrichment, and provides flexibility in determining the frequency of processing to meet business requirements.

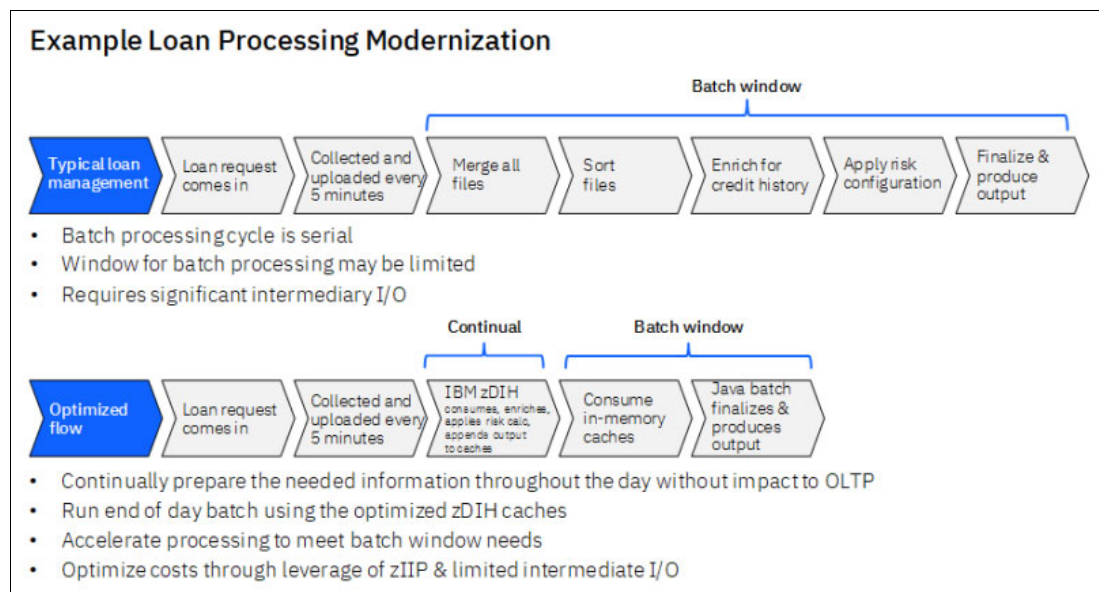


Figure 3 Loan processing modernization

Indeed, the transition from traditional batch processing to more intra-day processing is prevalent across various industries. The following are some examples of how different sectors can benefit from leveraging new technology enablers and optimization techniques to transform their processes:

- ▶ **Payments authorization:** Instead of processing payments in large batches at specific intervals, organizations can adopt real-time or near real-time payment authorization systems. This allows for faster validation and approval of transactions, improving customer experience and minimizing potential fraud.
- ▶ **Inventory management:** Moving from batch-based inventory updates to intra-day or real-time updates enables organizations to have accurate and up-to-date visibility into

their inventory levels. This helps in optimizing stock levels, reducing stockouts, and improving overall supply chain efficiency.

- ▶ **Fraud detection:** Traditional batch-based fraud detection systems can be enhanced by incorporating real-time analytics and machine learning algorithms. By analyzing transactions as they occur, businesses can identify and mitigate fraudulent activities more promptly, minimizing financial losses and protecting customers.
- ▶ **Claims processing:** Insurance companies can transition from batch-oriented claims processing to a more dynamic and agile approach. Implementing technologies like robotic process automation (RPA) and artificial intelligence (AI) can streamline claims intake, assessment, and resolution, resulting in faster claim settlements and improved customer satisfaction.
- ▶ **Transit booking:** With real-time availability and dynamic pricing becoming the norm in the transportation industry, shifting from batch-based booking systems to more intra-day processing allows customers to book, modify, and manage their reservations in real-time. This enables improved customer convenience and responsiveness to changing travel needs.

By leveraging new technology enablers, such as Java Batch and IBM Z Digital Integration Hub, businesses can transform their processes from traditional batch to more intra-day operations. This transformation enables faster decision-making, improved customer experience, enhanced risk mitigation, and overall operational efficiency in various industry sectors. Additionally, cost and performance optimization techniques can be applied to ensure the scalability, reliability, and economic viability of these new intra-day processing solutions.

Summary

Batch processing remains vital in many industries, but organizations can optimize and modernize it based on their business and IT goals. It is important to distinguish functions suitable for real-time, batch, micro-batch intra-day, or other processing methods. By doing so, businesses can streamline batch processing, reduce execution costs, improve security and compliance, and mitigate risks. In addition, transforming core processes like payments, claims processing, and loan approval from batch to more frequent or real-time processing presents significant business opportunities.

Authors

This publication was produced by a team of specialists from around the world working with IBM Redbooks, Poughkeepsie Center.

Pete McCaffrey

IBM, Principle Product Management GTM Lead, IBM Z Software

Mythili Venkatakrishnan

IBM DE, IBM Z Financial Services Sector CTO

Thanks to the following people for their contributions to this project:

Henry Vo, Lydia Parziale, IBM Redbooks Project Leader

IBM Redbooks, Poughkeepsie Center

Kyle Charlet
IBM Fellow, CTO IBM Z Software

David Betten
IBM Senior z/OS Performance Specialist - HiPODZ Team Lead, IBM Z Technology Sales

Suman Gopinath
IBM STSM & Chief Architect, DevOps for IBM Z Hybrid Cloud

David Sudlik
IBM Principal Storage Solution Specialist



REDP-5719-00

ISBN DocISBN

Printed in U.S.A.

Get connected

