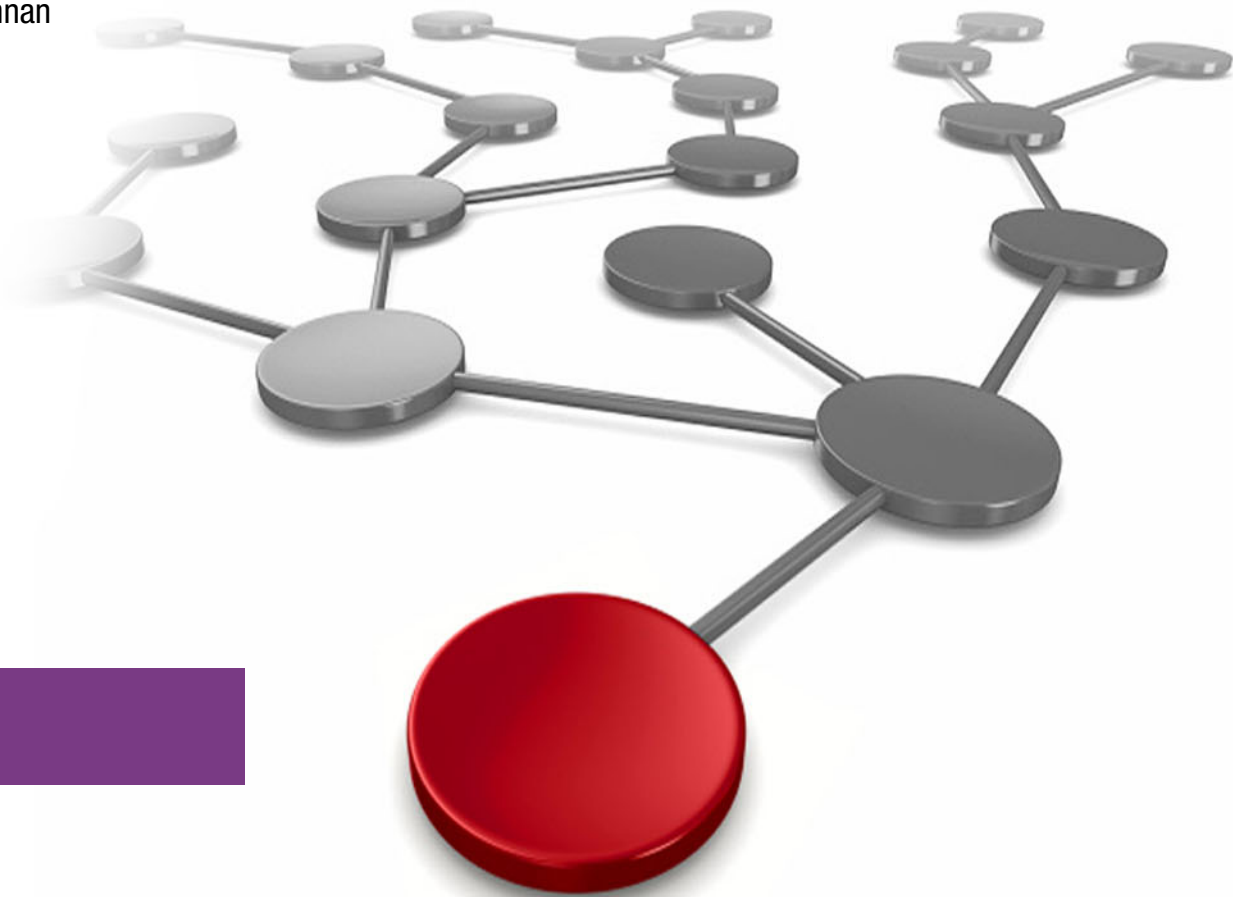


# Mainframe Batch Modernization and Transformation

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 and Transformation

Batch processing is used for some of an organization's most critical and time-sensitive business operations. It is used in numerous tasks, such as 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 archiving. Batch jobs perform many read, write, and sort activities on sequential and VSAM files with no need for input or manual intervention unless a job ends with an error. Batch window processing often takes place during non-peak hour. Often, online transaction processing (OLTP) must wait for the completion of the batch processing because OLTP input often requires the results of batch processing.

Within the enterprise domain, various applications require bulk processing to perform critical business operations. These operations involve the following characteristics:

### **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.

### **Periodic Business Rules**

Application of repetitive and complex business rules across extensive data sets. Examples include insurance benefit determination or rate adjustments.

### **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.

An ideal IT environment might seem to rely solely on OLTP in which user actions immediately trigger updates and deliver instant output. However, some functions in an information system 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 topics about 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 an IBM z/OS® environment, dedicated subsystems with special job scheduling software, such as [IBM Z® Workload Scheduler](#), manage batch processing.

The Job Entry Subsystem (JES) is a component of the [IBM z/OS](#) mainframe operating system that is responsible for managing batch workloads.

Job processing in a system is typically divided into several phases to enable parallelism through pipelining. These included phases are provided in the following list:

- ▶ **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.** After a job is scheduled, it enters the execution phase, where it runs and performs the specified tasks. This phase involves running the program or batch commands that are specified in the job control statements.
- ▶ **Output Processing.** After a job completes its execution, the output that is 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 status. For example, jobs that are running are placed on the execution queue. These queues, including execution queues and other relevant queues, are stored in the JES multi-access spool (MAS). The MAS is a shared storage area accessible by all systems in a group of interconnected mainframe systems called a Sysplex.

To protect against unexpected hardware or software failures, JES uses a mechanism that is called a checkpoint. A checkpoint backs up essential information about running jobs and their associated output. If a failure occurs, the checkpoint can be used to restore the jobs and their output to ensure that progress is not lost and to allow 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 needs of the business, it uses modern technologies to optimize processing.

Many batch processes include multiple critical paths that consist of dependencies between batch jobs leading to critical end points. IBM [Workload Manager](#) (WLM) works 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 that 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

Batch processes commonly used in industries today include the following examples:

- ▶ Core industry-specific processes including:
  - memo-posting of banking transactions
  - settlement and clearing
  - claims processing
  - risk assessments
  - ledger postings
  - 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 the 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 by using programming languages like COBOL. However, it is important to understand 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 remains a valuable and strategic approach. This type of processing often involves integrating external data feeds. These data feeds need to be reconciled in the correct sequence with OLTP activity to generate consistent end-of-day results. By using batch processing, organizations can efficiently handle the high-volume data processing that is required for these reconciliation and consolidation tasks.

## Designing an application function as a batch process

The following list provides examples as to why you might design an application function as a batch process:

- ▶ Dependency on unavailable resources

If real-time processing of the business logic is not possible due to dependencies on information or resources that are not available then, then 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 more 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 that are 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 using batch processing for these scenarios, organizations can optimize resource usage, handle large data volumes, meet specific timing requirements, and possibly improve efficiency for various business operations.

## Batch analysis and assessment

IBM offers a range of batch analysis and assessment tools and expertise to help businesses optimize mainframe usage for batch processing. These tools and services are designed to help clients identify inefficiencies and fully use 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 delivers insight for IT Operations Managers, Performance Experts, and Capacity Planners to make informed decisions about their infrastructure and application performance including batch.

You can use structured metric files (SMF) and other structured data sources on IBM Z and other platforms to help curate enterprise-wide IT usage information more quickly and efficiently into actionable reports that can be used to identify opportunities for batch improvements.

IBM Z Performance and Capacity Analytics can be used with [IBM Application Discovery and Delivery Intelligence \(ADDI\)](#). The combination can be used to pinpoint batch applications with high resource consumption and to run 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 that are needed with no guarantee that the updates are compatible. 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 usage, 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. These ongoing investments enable clients to optimize costs that are associated with existing applications and the deployment of new or modernized applications.

By using current software and hardware, clients can use 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 and has 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 that is 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, and the use of VSAM Record Level Sharing (RLS), allow for improved batch concurrency with OLTP.

IBM continues to enhance and optimize programming languages like COBOL and PL/I to fully use 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 that is designed to optimize COBOL modules without requiring source code recompilation. It is built to support the latest IBM Z hardware and can help reduce 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 the same way as the originals but use fewer system resources, which can reduce costs.

When used with the latest IBM Enterprise COBOL compiler for z/OS, ABO becomes a complementary tool. Organizations can use the COBOL compiler during active application development or maintenance phases to optimize specific parts of the application. ABO can then be used to optimize the remaining parts to help maximize the return on investment for IBM Z.

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

Overall, the investments, optimizations, and comprehensive toolsets that are provided by IBM can help clients effectively address cost and performance challenges in batch processing and maximize the benefits of their mainframe deployments.

# Java batch on the mainframe

This section provides an outline of some of the benefits of using Java batch.

IBM mainframes support and run batch processes that are written in the Java language. Java Batch can be useful when there is a need to transform existing batch processing or accelerate the change to more real-time processing.

Java Batch provides an option for modernizing batch operations and can enable the integration of more real-time processing capabilities. Also, the availability of Java programmers in the market is typically 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 use the lower-cost specialty processors that are known as IBM Z Integrated Information Processors (zIIPs) to help manage the costs that are associated with running batch processes in parallel with OLTP workloads. The second advantage is that running batch during the day on zIIP processors can shorten the batch processing window that is required by preparing relevant information and 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 sometimes faster than, compiled languages. Because bat run time. The IBM Z processor chips have instructions that are specifically designed to assist JIT-compiled code, which can provide additional performance benefits. In contrast, COBOL code that has not been recompiled for a long time might not fully use these chip instructions, which makes it less optimal in terms of performance compared to Java in certain scenarios.

Overall, using Java Batch on IBM mainframes offers opportunities for modernizing batch processes, by using 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.

The following list includes some of the facilities and features that are provided by the JZOS Batch Launcher and Toolkit:

- ▶ Run Java applications on z/OS seamlessly in an MVS batch job step or started task
- ▶ Simple but flexible configuration of the Java execution environment
- ▶ Access to data sets by using 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 can be made easier through various enhancements. The combination of the launcher, data access capabilities, system services, and environmental improvements has simplified the process, which specifically benefits Java application developers.

With these enhancements, managing Java batch jobs on z/OS is now like managing batch applications that are 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. JIB enables a unified approach to managing batch jobs and ensures 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 and 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 and enables reliable and streamlined execution of batch jobs.

## Using Jakarta or Spring Batch frameworks

Both Jakarta Batch and Spring Batch are 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 that use XML and Java application artifacts, which promotes reuse across different jobs. Jakarta Batch includes the following key features:

- ▶ checkpoint and restart functionality
- ▶ step composition for reuse and restart ability
- ▶ XML configuration for externalized configuration
- ▶ support for parallel processing of data through partitions

Spring Batch is a lightweight and comprehensive batch framework that allows developers to build robust batch applications for enterprise systems. It uses the productivity and ease-of-use associated with the Spring Framework. It includes the following features:

- ▶ 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 is important to note that although 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 builds upon the characteristics of Spring Framework and includes additional enterprise services for batch processing. Also, Spring Batch does not include scheduling capabilities but can be used alongside a scheduler.

Overall, both Jakarta Batch and Spring Batch can serve as valuable tools for Java developers that work 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

This section includes discussions of modernizing batch reporting, the IBM migration utility and moving from traditional ETL to read-transform-serve.

### Modernizing batch reporting

Report generation is a common batch use case, and there are various report generation tools for the mainframe. Cost of ownership and functional limitations for some of these tools can prompt businesses to evaluate other options.

In one example, IBM provides [IBM Migration Utility for z/OS](#) for converting report generation to use IBM mainframe COBOL. Some of the benefits are included in the following list:

- ▶ 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.
- ▶ Cost savings by eliminating licensing costs for separate report generation software.

### IBM migration utility

The IBM Migration Utility gives you the following choices:

- ▶ You can continue developing programs that use 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 the COBOL code.

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

## Change traditional ETL to Read-Transform-Serve

ETL is a data integration process that is used to combine data from various sources into a consistent data store, typically a data warehouse or target system. It originated in the 1970s with 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 that is processed through ETL grows, it leads 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 using 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 by using existing tools, technology, and utilities. This approach minimizes processing costs and potential security risks that are associated with moving large amounts of data.

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

As businesses accelerate their digital initiatives to better address customer expectations, there is a requirement to run core business processes increasingly in real-time. This includes the following processes:

- ▶ loan origination and approval
- ▶ clearing and settlement
- ▶ payments authorization
- ▶ fraud detection
- ▶ inventory management
- ▶ asset management
- ▶ transportation booking

Today, many of these processes are run 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.

In some strategies you can use different 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

IBM Db2 Analytics Accelerator (IDAA) is a high-performance component that is 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 can reduce 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 the 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, 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 high-performance in-memory caches that use a Java-based runtime environment. IBM zDIH 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. It 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 route 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.

### Clearing and settlement use case

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, performance challenges might exist because of resource and dataset access contention. In addition, running both batch and OLTP at the same time might increase demand for mainframe general-purpose capacity.

Depending on the implementation, using 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 by using the in-memory caches for manipulation and avoid intermediary I/O to files. Alternatively, 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 because of a reliance on third-party files. You can create a Java Batch program that uses zDIH as a data source with other data sources that are needed to complete the batch processing. You can use JDBC or standard Java interfaces for the zDIH caches. You run this batch job in parallel to the OLTP window. Because the job is running against zDIH instead of the core business systems, you avoid potential performance issues, and you optimize costs. Because zDIH is zIIP eligible, the production datasets are not impacted.

For the clearing and settlement use case, you must first understand what information is needed to process, how much data is being processed, what transforms are required, and what the overall processing flow is. There might 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 that can do the processing by 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.

The graphics in Figure 1 illustrate optimizing intraday batch with zDIH.

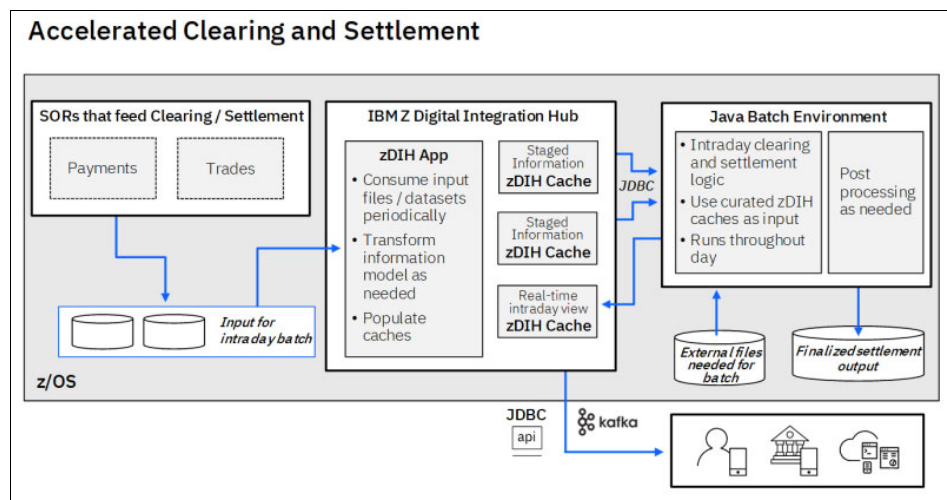


Figure 1 Accelerated clearing and settlement



## Loan processing use case

Figure 2 provides 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.

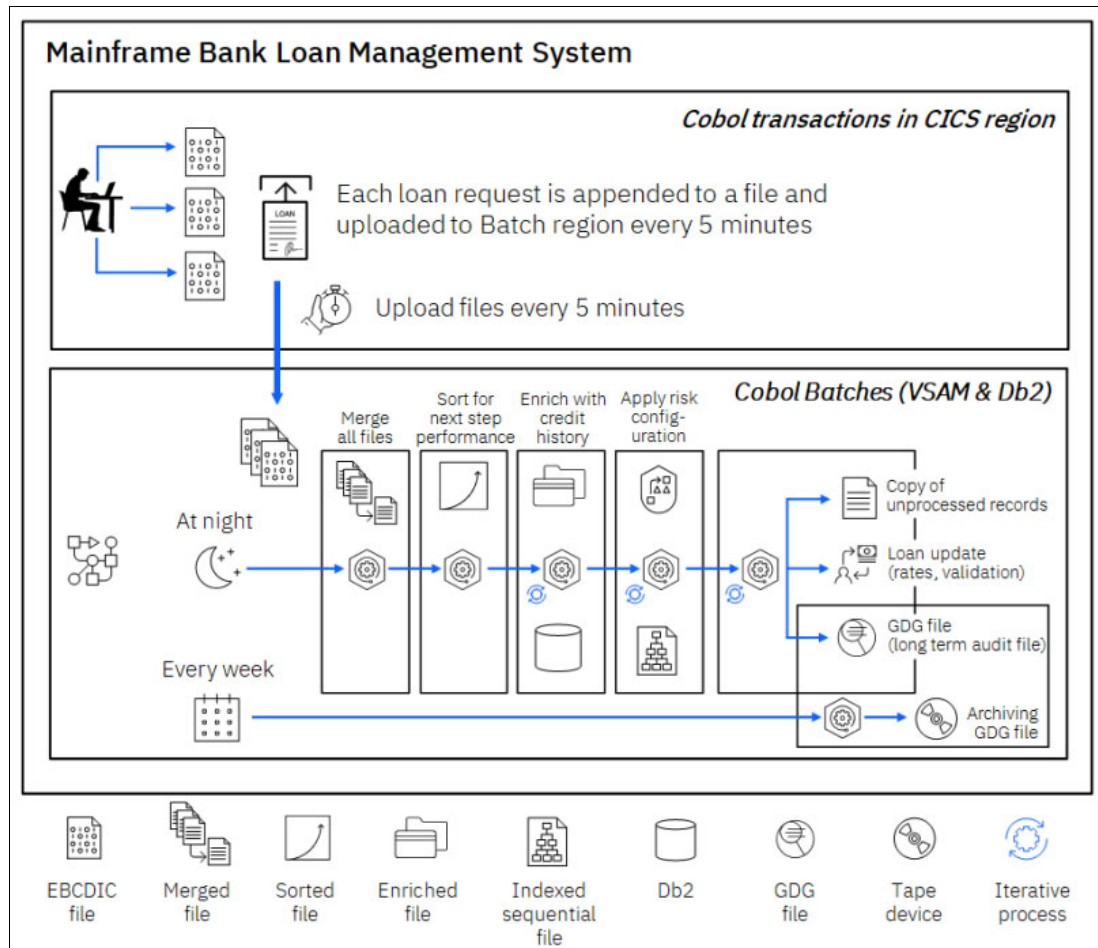


Figure 2 Mainframe Batch architecture example.

The loan processing batch workflow that is described consists of three programs with specific functions:

### 1. Upload batch program

This program runs every five minutes and sends temporary files that are generated by loan request transactions to the Batch region by 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 runs 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:
    - i. Unprocessed or rejected record. Contains copies of records that require further processing due to parsing errors, missing elements, or other failures.
    - ii. Db2 table updates. Updates the records for each customer that requests a loan in a Db2 table, indicating the status of the loan request: rejected, approved, pending. For approved requests, it also includes loan proposal details such as rates and duration.
    - iii. 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:
- a. Tape backup. Saves selected GDG data to tape devices for legal reasons, ensuring data preservation and compliance.
  - b. Pruning. Removes GDG files that were 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 uses Java Batch and IBM Digital Integration Hub (zDIH) to enable efficient and parallel processing. Figure 3 on page 14 shows an example of loan processing modernization. The following list provides an overview of the suggested approach:

1. Real-time processing
  - Instead of waiting for end-of-day processing, loan requests are processed during the day as they arrive. 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 using 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 used in a continual run approach to retrieve enriched personal information, such as credit history and other loan details that was previously 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

After the loan processing is completed, the Java Batch application creates the necessary output files based on the processing results. This might 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 using 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.

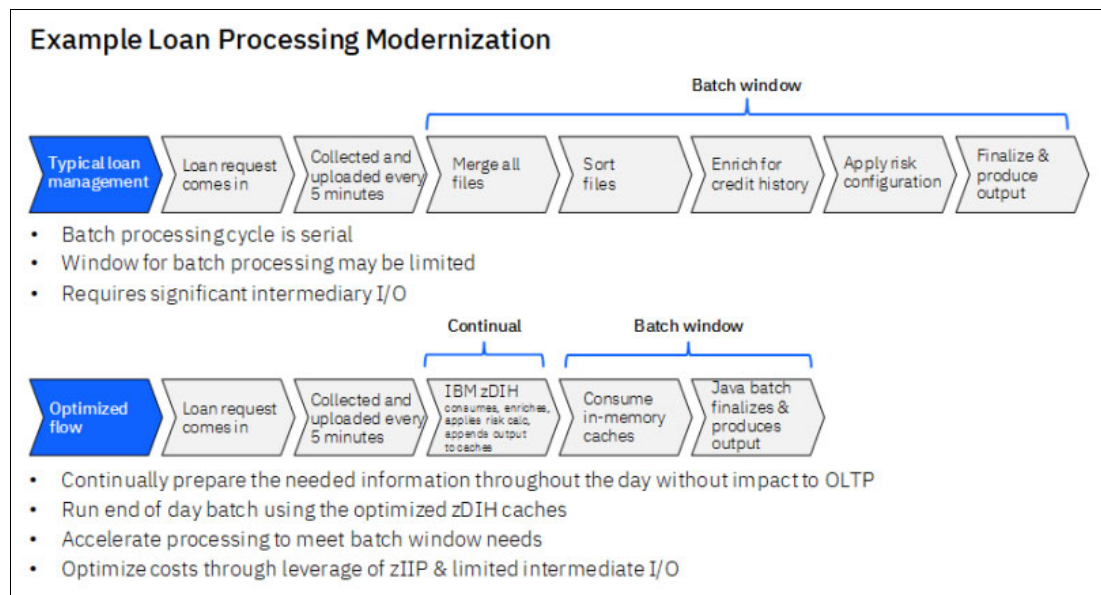


Figure 3 Loan processing modernization

The transition from traditional batch processing to more processing during the day is prevalent across various industries. The following examples describe how different sectors can benefit from using 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 intraday 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 move 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 intraday 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 employing 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. Also, 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 Follis  
**IBM Architect WebSphere Application Server on z/OS and Java Batch**





REDP-5719-00

ISBN 0738461393

Printed in U.S.A.

Get connected

