WebSphere Application Server:

Enabling incremental Java batch modernization

Speaker Name and Title

Agenda

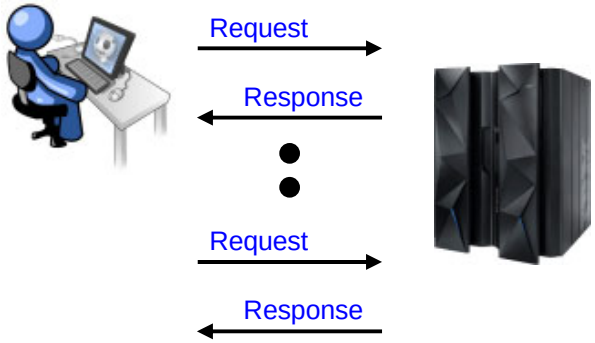
- Batch modernization defined
- WebSphere Java Batch Overview
- Unique value of WebSphere Java Batch on z/OS
- Enabling incremental adoption
- Summary and Wrap-Up

What is batch modernization?

What is Batch (or Bulk) Processing?

Many definitions exist ... They have in common that there is minimal human interaction and expectation of results at a future time rather than immediately.

Online Processing



In general:

- Interaction is *one-for-one* ... that is, request with matching response
- Expectation is for response to follow request in a *near-immediate* time frame
- **Examples:** Inventory query, Website shopping transaction, eBanking account withdrawal

Bulk Processing

In general:

- Interaction is *one-for-many*... that is, initial request results in many results from processing
- Expectation is for results to finish within some determined *non-immediate* time frame
- **Examples:** Month end tax calculation, Period end statements and reports, Data transformation, Data analysis

High — Interaction and Responsiveness — Low

What is driving batch modernization?

Companies are facing significant business pressures to deliver services with utmost efficiency and resource usage, while also reducing costs

✓Improve Process Execution

- There is a continuous need to streamline the delivery and execution of services that support real time, or ***Online***, customer interactions as well as business critical bulk data processing, or ***Batch***, workloads.

✓Maximize Resource Utilization

- Companies must **leverage common skills and technologies** to move away from organizational silos that drive up development and maintenance expenses.

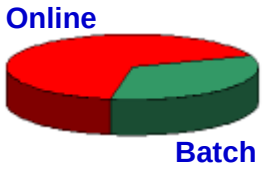
✓Reduce Costs

- Emphasis continues to be on reducing costs attributed to business supporting services, while at the same time delivering maximum performance.

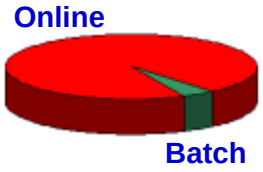
Batch processing is evolving

Windows of time which used to be dedicated to batch processing are shrinking, while batch workloads continue to grow as online transactions increase

In the past ...



Today ...



24 x 7 x 365 Access

Users of your online systems expect availability at all hours.

Users from other parts of the world means availability is expected around the clock.

Mobile Users

Users are no longer tied to a desk and a computer. Today users have access to mobile computing devices that are with the user wherever they may be. Day or night, home or office.

The need to process batch work has not gone away.

The need to perform batch concurrent with online has emerged.

Redundant staff and IT are under review

It's not *just* a shrinking batch window... it's also the pressures to reduce duplication in staff and infrastructures supporting batch and online environments

Batch Support Staff

Online Support Staff

- Batch Infrastructure
- Online Infrastructure
- Batch Applications
- Online Applications
- Batch Development Tools
- Online Development Tools
- Homegrown Middleware Infrastructure

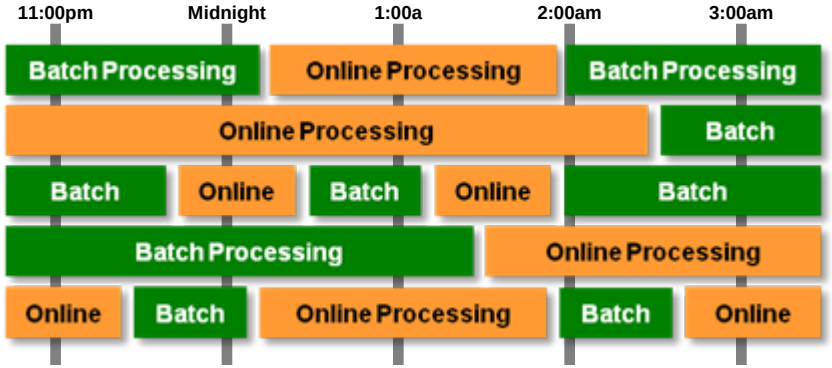
=

- Increased Costs
- Slow Time to Market
- Skills Imbalance

Maintaining homegrown infrastructures for separate online and batch processing results in increased costs and decreased business focus

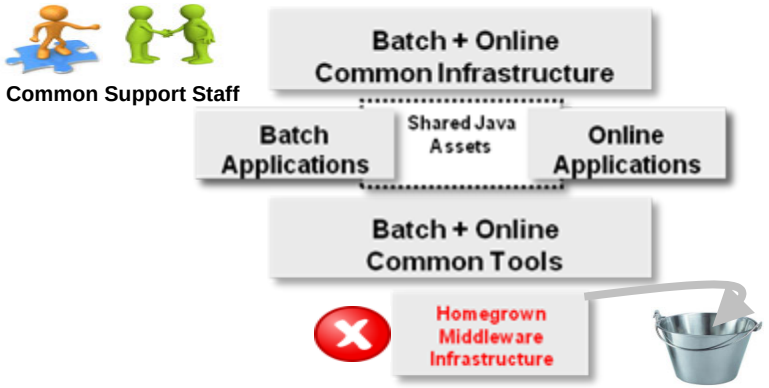
WebSphere Batch addresses these issues

Integrated Batch and Online



Online and Batch execution within a common runtime, managing workloads by priority to respond to capacity fluctuations and meet SLAs

Shared Services and IT Infrastructure

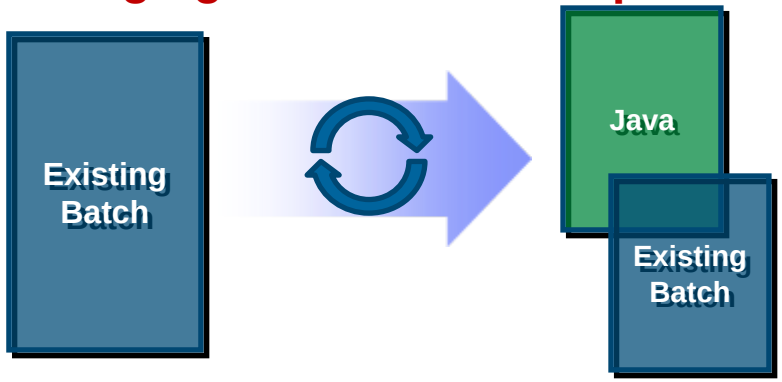


Consolidation around common tools and technologies enable shared services, code re-use and support optimizations

Better Resource and Skills Utilization

Evolving batch processing to leverage Java enables alignment with online application skills, and more efficient skills utilization

Enablinging Incremental Adoption

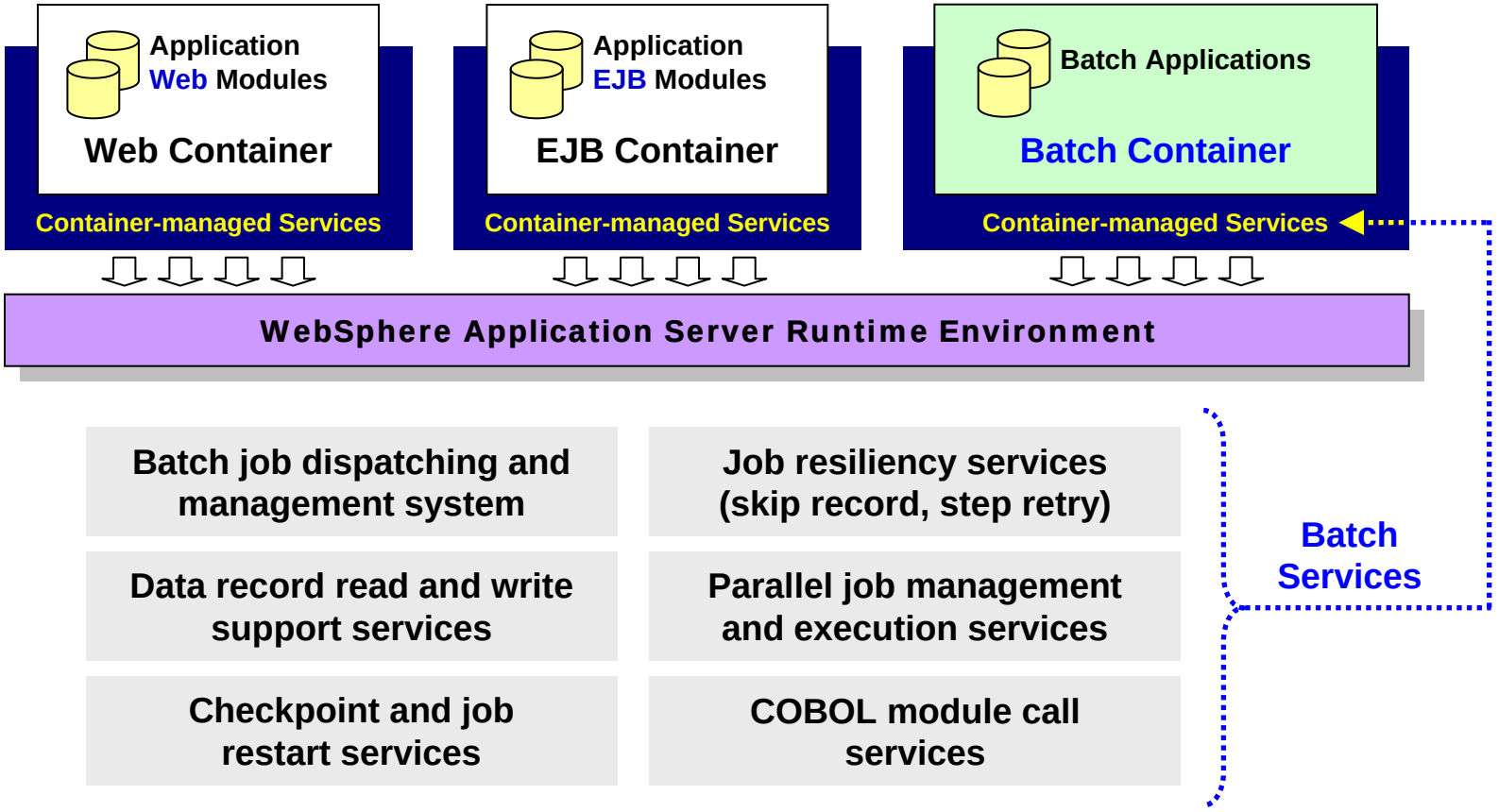


Integration with existing processes and technologies allow for preserving investments and incremental modernization of batch

WebSphere Batch Overview

Batch is fundamental to the WAS Runtime

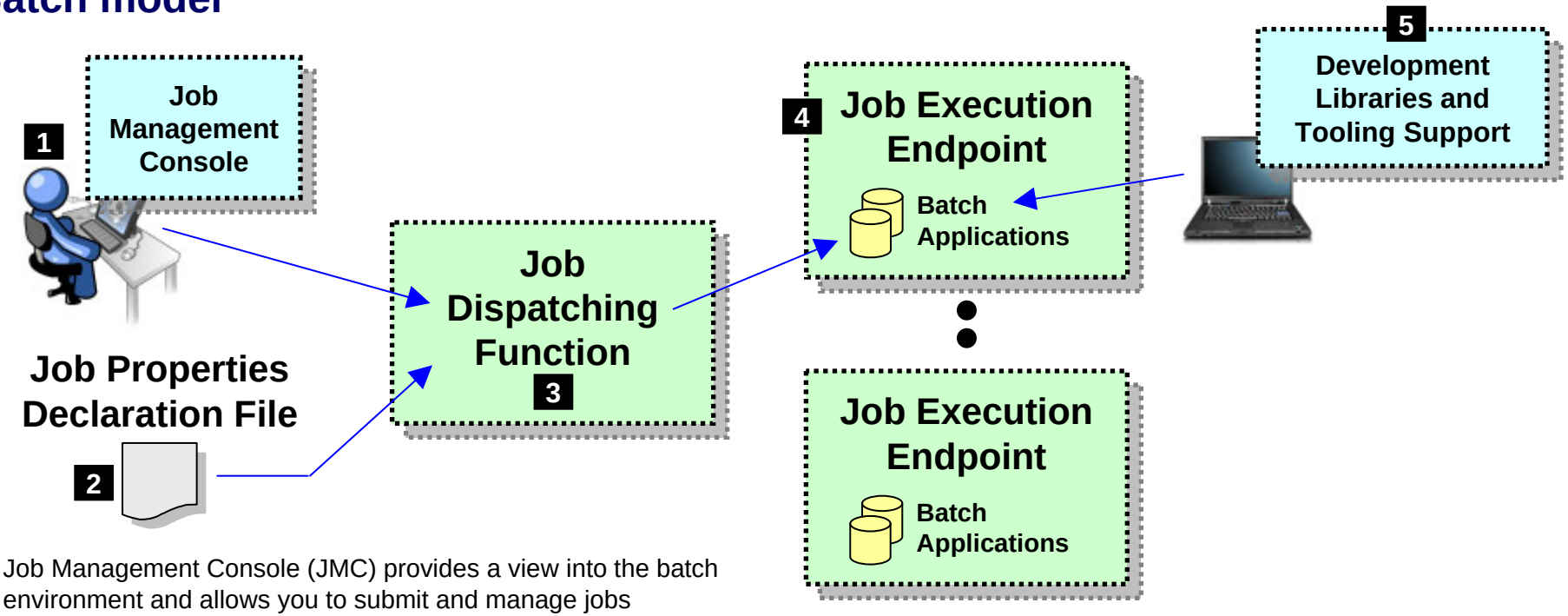
Think of IBM WebSphere Java Batch function as a "batch container" operating alongside the other containers of WAS itself:



Note: WebSphere batch capabilities are fully integrated in WebSphere Application Server (WAS) as of v8.5. For installations prior to WAS v8.5, the WebSphere Compute Grid product is required.

Batch Management and Execution Model

This picture illustrates some of the key components of the WebSphere Java Batch model



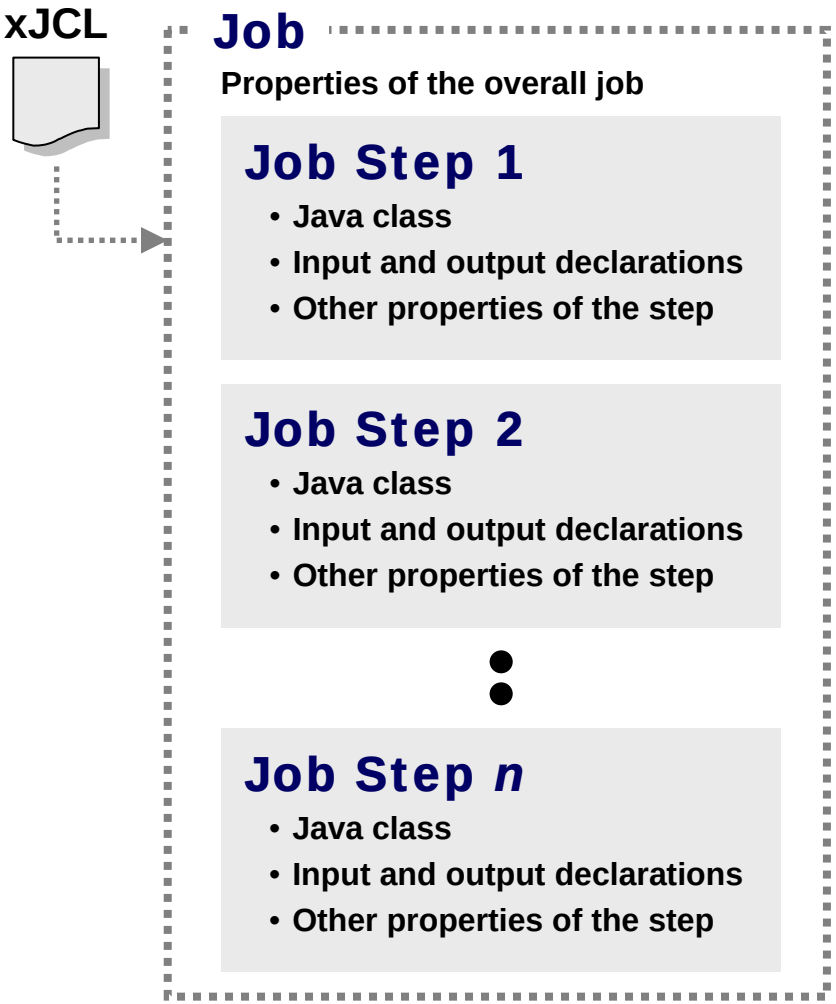
1. Job Management Console (JMC) provides a view into the batch environment and allows you to submit and manage jobs
2. Job declaration file (xJCL) provides information about the job to be run, such as the steps, the data input and output streams and the batch class files to invoke
3. The Job Dispatching function interprets the xJCL, dispatches the job to the endpoint where the batch application resides, and provides ability to stop and restart jobs
4. The Execution Endpoint is a WAS server in which the deployed batch applications run
5. The development libraries and tooling assist in the creation of the batch applications

A comprehensive Java batch execution platform

Built on the proven Java runtime environment of WebSphere Application Server

Batch Job and Batch Job Steps

A batch job consists of one or more steps executed in order specified in xJCL:



- The xJCL is submitted through the Job Management Console
  - Interfaces provided: HTTP browser, command Line, Web Services, RMI
- The Job Dispatching function interprets xJCL and determines which endpoint has batch application class files deployed
- The Dispatching Function invokes the job and passes to the endpoint an object containing all the properties in xJCL
- Steps are executed in order, with conditional step processing if declared
- The Dispatching Function maintains awareness of the job state
- When the job ends, the job output file is accessible through Job Management Console

The Job Definition

Job Declaration File - xJCL

Conceptually, xJCL is just like normal // JCL -- it describes the job to be run and the context in which the job is to operate. The difference is xJCL is written in XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
<job name="MyJob" ... ">
  <substitution-props>
    <prop name="inputDataStream" value="/tmp/input-text.txt" /
    <prop name="outputDataStream" value="/tmp/output-text.txt"
    <prop name="checkPoint" value="10" />
  </substitution-props>

  <job-step name="MyStep1">
    <classname>com.ibm.ws.batch.MyStep1</classname>
    <batch-data-streams>
      <bds>
       <logical-name>inputStream</logical-name>
         <props>
           <prop name="PATTERN_IMPL_CLASS" value="com.ibm.webs
           <prop name="FILENAME" value="${inputDataStream}" />
         </props>
       </bds>
    </batch-data-streams>
  </job-step>
     :
  <job-step ...>
  </job-step>
</job>
```

Comparable to the JCL "JOB" card. It sets job-level information along with some substitution properties Substitution properties may be overridden at submission time.

Comparable to the JCL step. It names the Java class that implements the step function.
The "Batch Data Stream" implementation is specified. In this example it defines the class that implements the input stream.

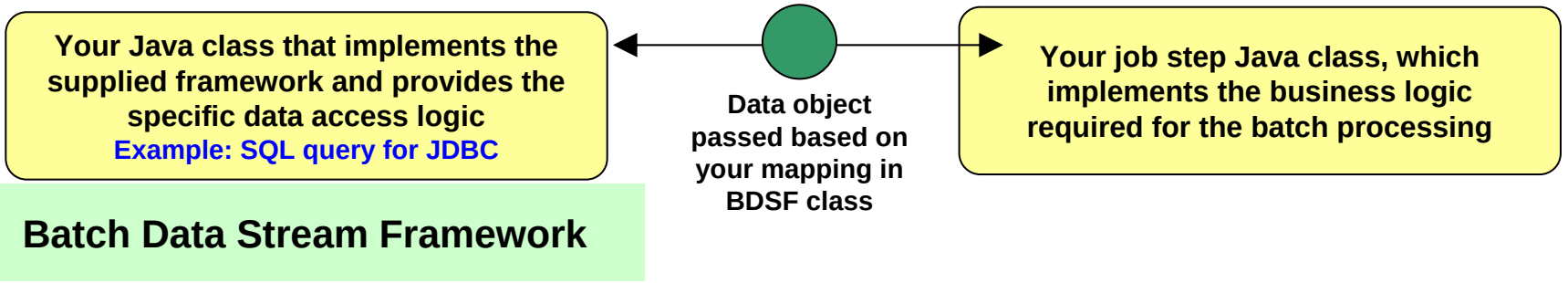
If job consists of more steps they are specified in sequence.

This tells WebSphere Java Batch *what* to run and *how* to run it

Note: this is a trimmed version of actual xJCL

Batch Data Stream Framework (BDSF)

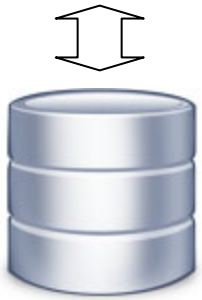
A key service provided by the batch container – it abstracts data read and write operations so your code may focus on the business logic:



Batch Data Stream Framework

Supplied "patterns" for data access:

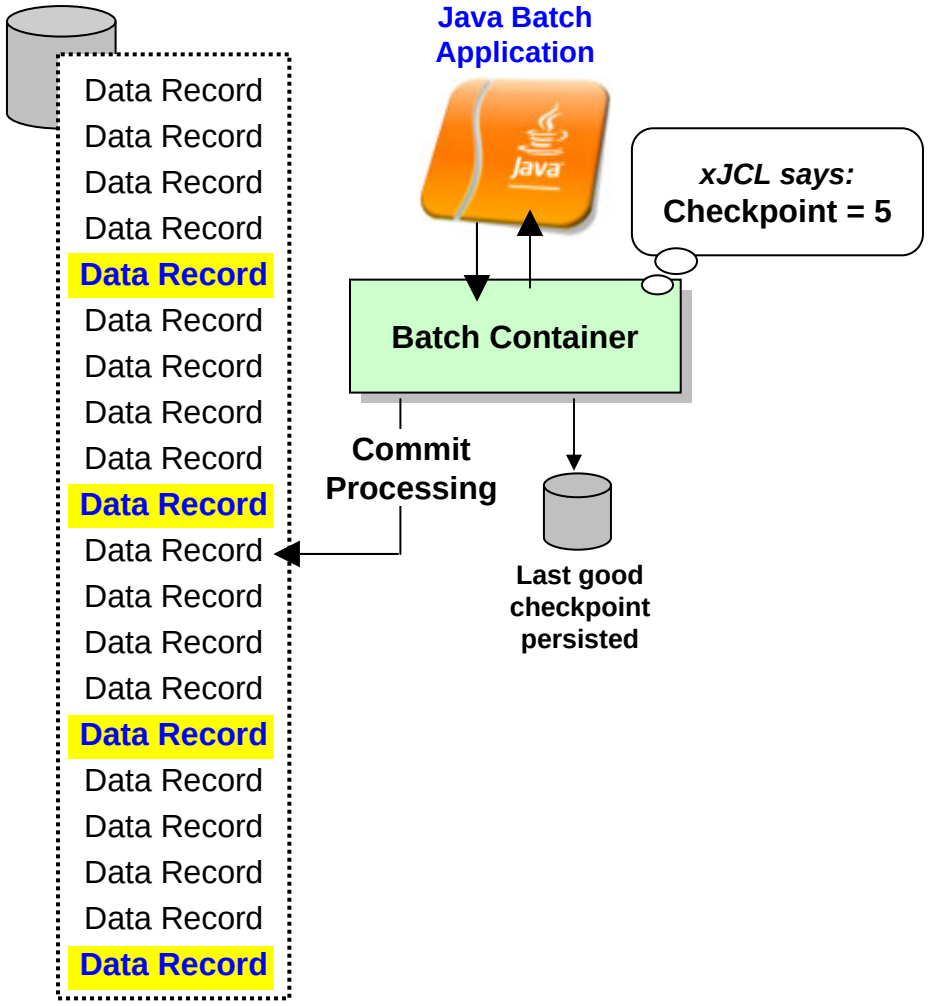
- **JDBC read or write operations**
- **JPA read or write operations**
- **File read or write operations**
- **z/OS Data Set read or write operations**



- Batch Data Stream retrieves result set from data persistence store (DB, file, etc.)
- Batch Data Stream maps data fields to data object
- For each record in result set, BDSF invokes your job step, passing a data object mapped to your specifications
- Your job step code stays focused on business logic, not Java stream handling and data object formatting

Transactional Checkpoint Processing

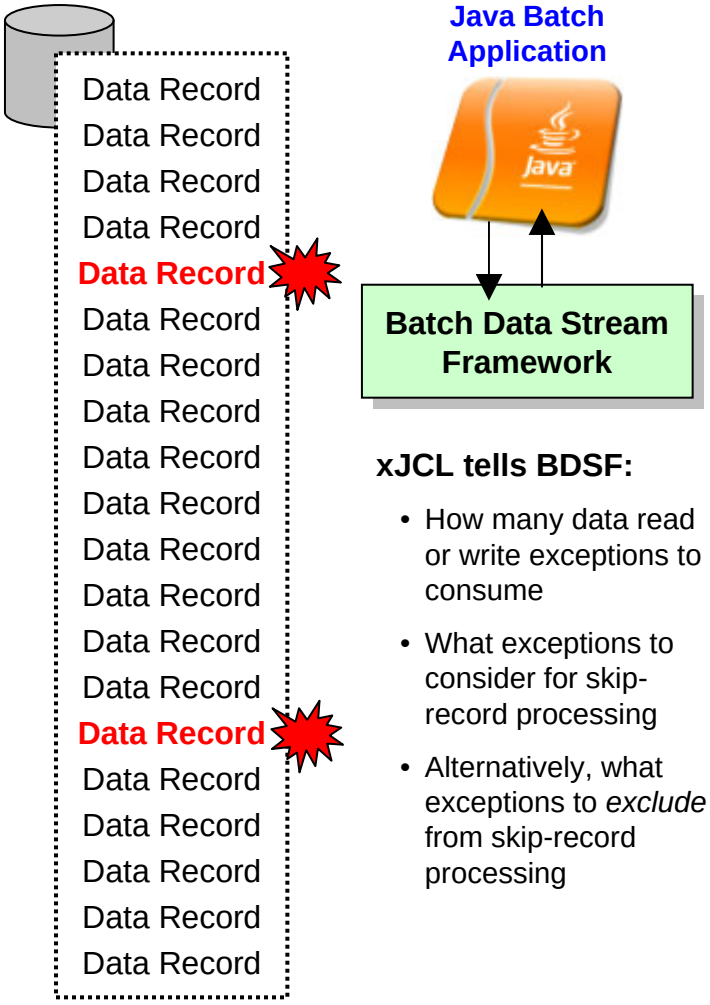
The batch container provides the ability to checkpoint at intervals based on either record count or time. The container keeps track of last checkpoint.



- Checkpoint interval (record or time) specified in the xJCL
- This is a function of the batch container, *not* your application code
- As checkpoint intervals are reached, container commits and records the checkpoint attained
- In the event of a failure, job may be restarted at the last good checkpoint
- Set the checkpoint interval based on your knowledge of balance between recoverability and efficiency

Skip-Record Processing

Provides a container-managed way of tolerating data read or write errors so the job itself may continue, while information about data errors can be logged.



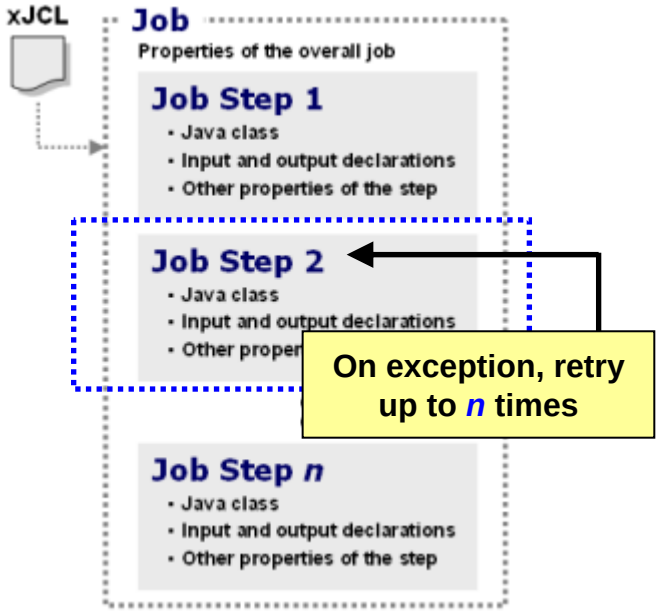
xJCL tells BDSF:

- How many data read or write exceptions to consume
- What exceptions to consider for skip-record processing
- Alternatively, what exceptions to *exclude* from skip-record processing

- Objective: allow job to continue if a data read or write exception occurs in BDSF
  - Why fail a million-record job just because of one or two read or write exceptions? Better to complete the job and allow auditors to go back and investigate the few exceptions.
- Skip-Record processing allows BDSF to keep exception and *not* surface it to your application
  - This takes burden off your application code to explicitly handle data read or write exceptions that may occur
- A "skip-record listener" may be called so your code may log information about skipped records
- xJCL properties allow you to specify how many records may be skipped and what exceptions to include or exclude from consideration
- When skip limit is reached, further exceptions are surfaced to application. That may result in job failing and going into a restartable state
  - Normal restart-at-checkpoint would occur

Retry-Step Processing

Provides a means of retrying a job step in the event of an exception thrown. If successful on retry then the job continues and your processing completes.



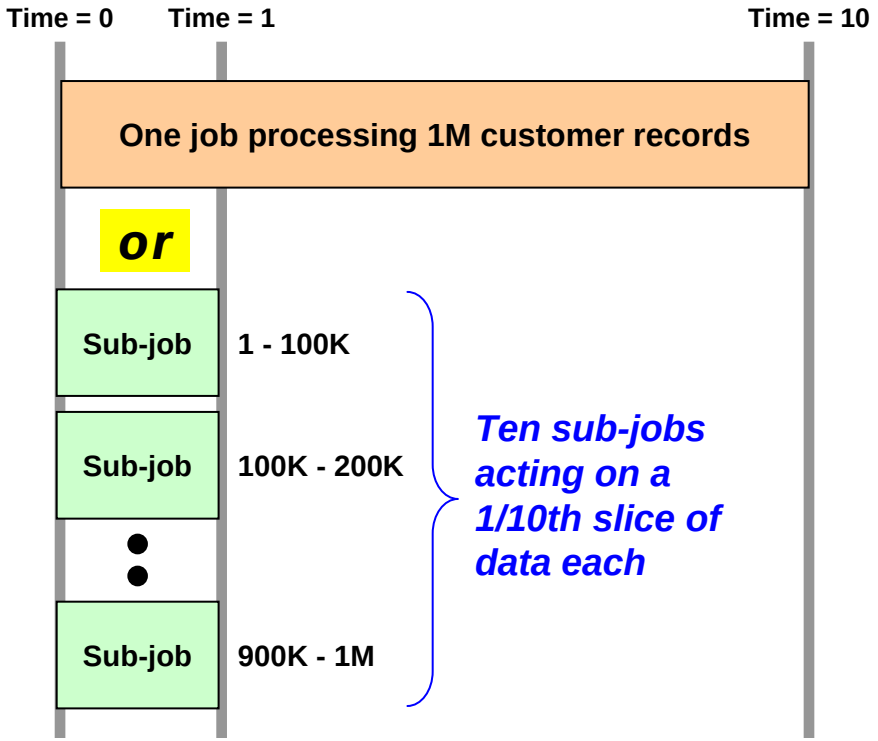
xJCL tells Container:

- How many step retries may be attempted
- What exceptions to consider for retry-step processing
- Alternatively, what exceptions to *exclude* from retry-step processing
- Whether to process a delay before attempting a retry of the step

- Objective: retry step in attempt to allow overall job to continue and complete when an unanticipated exception is thrown
- This is at a level higher than skip-record ... this is if an unhandled exception is thrown when the job step function is called
- Batch container falls back to last good checkpoint and restarts from there
- A "retry-step listener" may be called so you can perform custom action upon retry-step processing
- xJCL properties allow you to specify how many retry attempts will be performed and what exceptions to include or exclude from consideration
- When retry limit is reached, job will go into restartable state

Parallel Job Manager

The Parallel Job Manager (PJM) provides a way to "parameterize" logic so parallel sub-jobs may act on a slice of the overall batch job data:



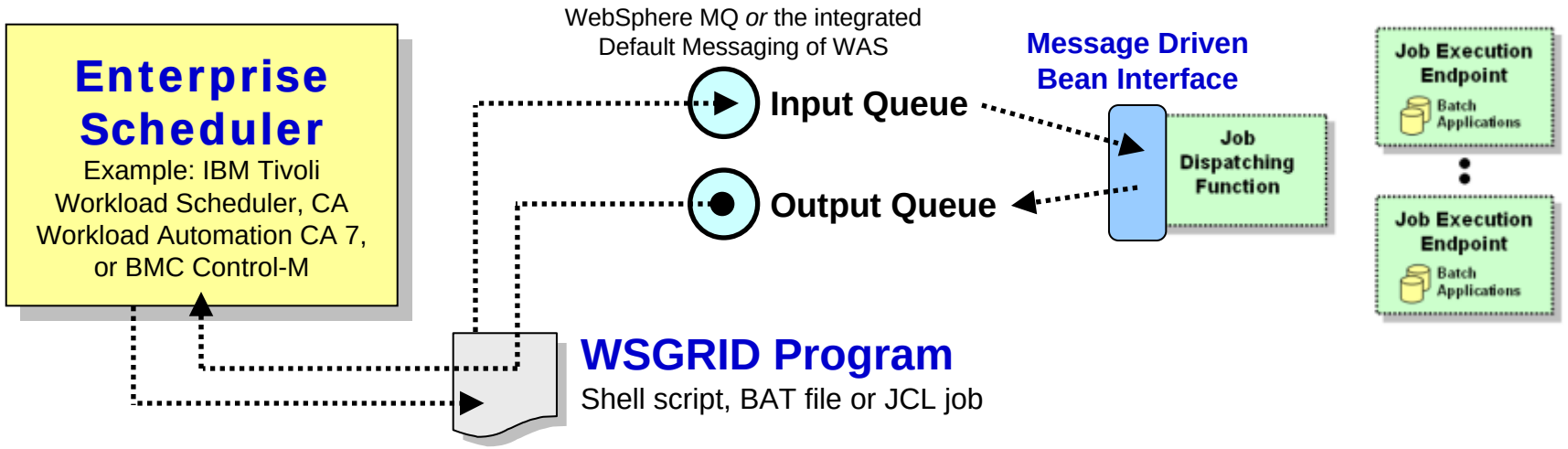
- xJCL specifies whether job is to be run in parallel, and if so how:
  - One JVM, multiple threads
  - Multiple JVMs
- Your "parameterizer" code is called at start so data range may be segmented into sub-job slices
- Job is submitted, then PJM dispatches "sub-jobs" to act on each data range
  - "Parameterizer" code constructs data range query strings to be used by each sub-job
- PJM manages "top-job" and all subordinate "sub-jobs" to completion

Objective is reduction in overall job completion time

Which shortens overall batch window if other jobs are dependent on this job for completion

Integration with Enterprise Schedulers

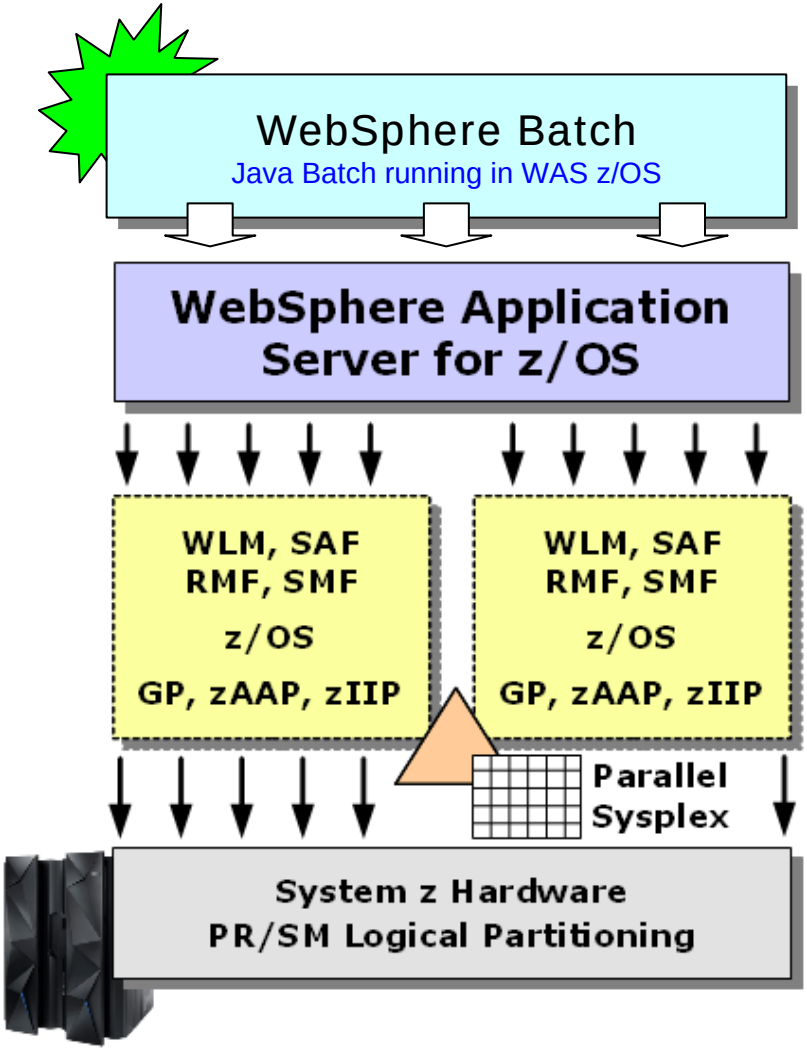
The Job Dispatching Function has a Message Driven Bean (MDB) interface. IBM supplies a program that integrates schedulers with WebSphere Java Batch:



- WSGRID is seen by Scheduler as any other batch job it starts and monitors
- WSGRID interacts with Job Dispatching, submitting the job and processing Java batch job output back to STDOUT or JES Spool if z/OS
- WSGRID program stays up for life of job in WebSphere Java Batch
- To the Scheduler, WGRID *is* the Java Batch job ... but behind WSGRID is all the WebSphere Java Batch function we'll discuss

WebSphere Batch on z/OS

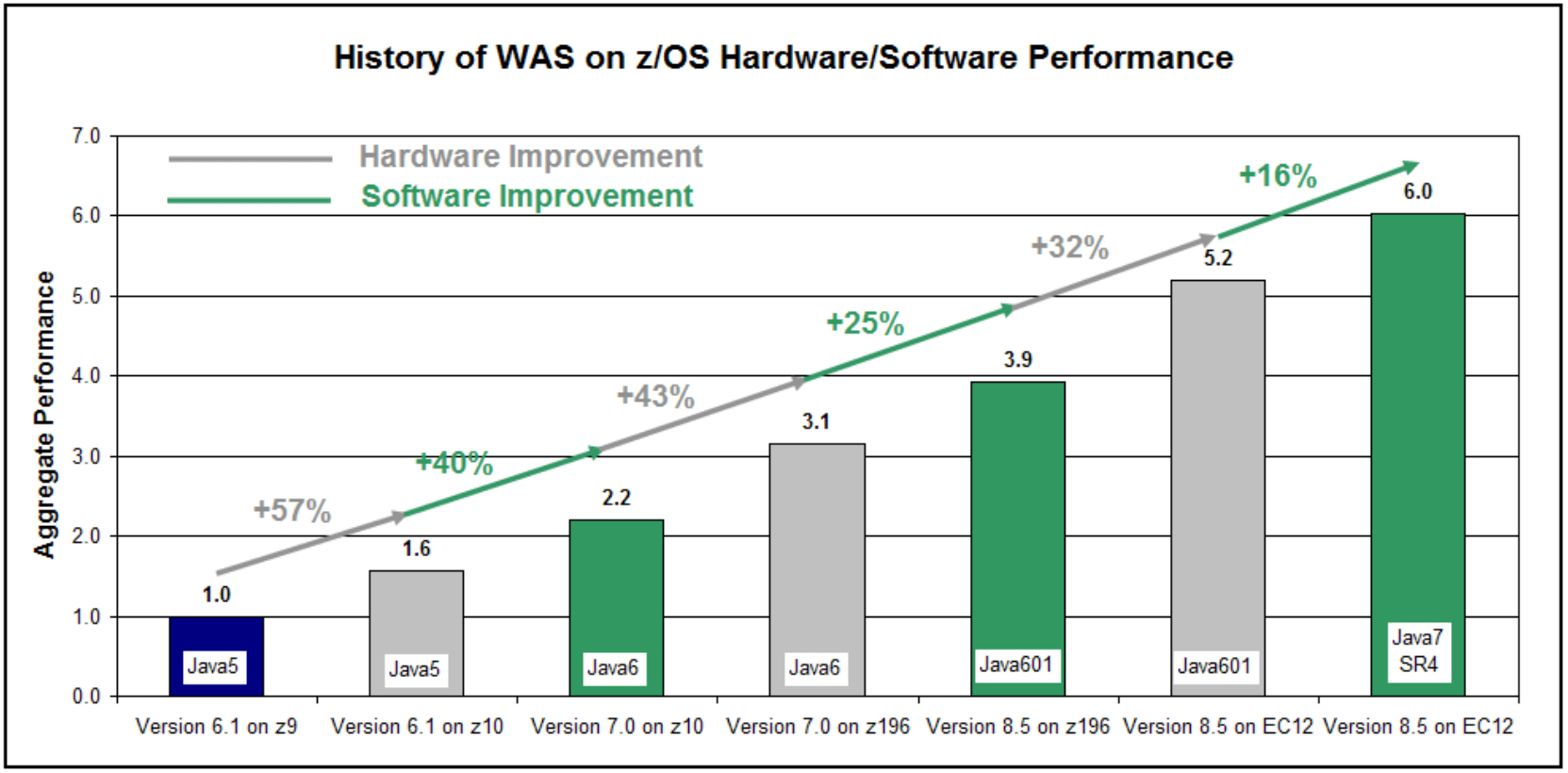
Unique Value of WebSphere Batch on z/OS



- All the WAS z/OS value statements apply to WebSphere batch on z/OS as well, such as:
  - Reliability and availability
  - Shared data in Parallel Sysplex
  - Mapping parallel jobs onto clustered WAS servers in Sysplex with shared data infrastructure
  - Batch job classification with WLM
  - SMF recording of batch activity
  - Integration with enterprise scheduler functions with MQ BINDINGS
  - Local communications when co-located with applications and deata

Performance of Java and WAS on z/OS

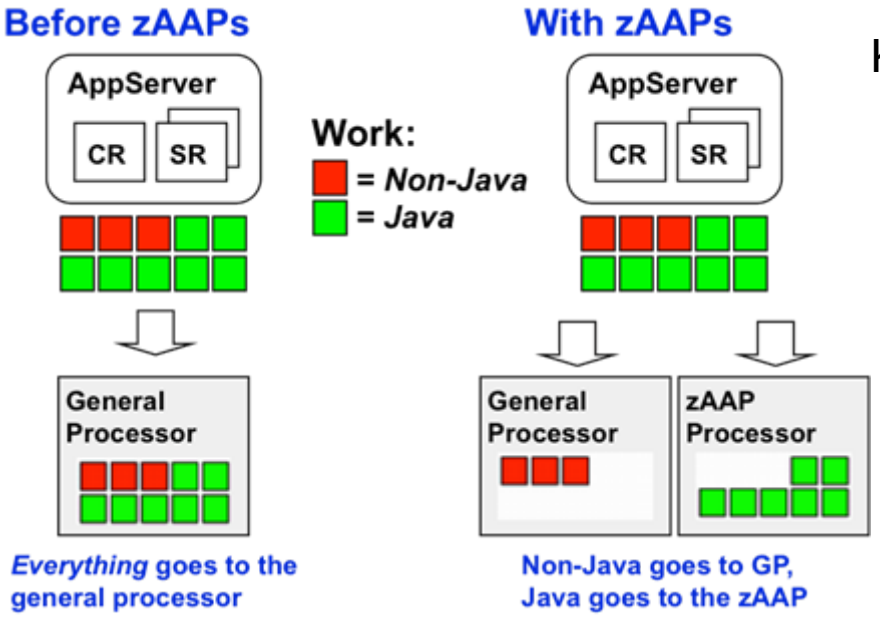
Aggregate HW, SDK and WAS Improvement: WAS 6.1 (Java 5) on z9 to WAS 8.5 (Java 7) on zEC12



~6x aggregate hardware and software improvement comparing WAS 6.1 Java5 on z9 to WAS 8.5 Java7 on zEC12

Java Offload to zAAP Specialty Engines

zAAP engines are Java offload engines. They enhance the financial picture of the z/OS platform, and they free up GP for other key subsystem processing



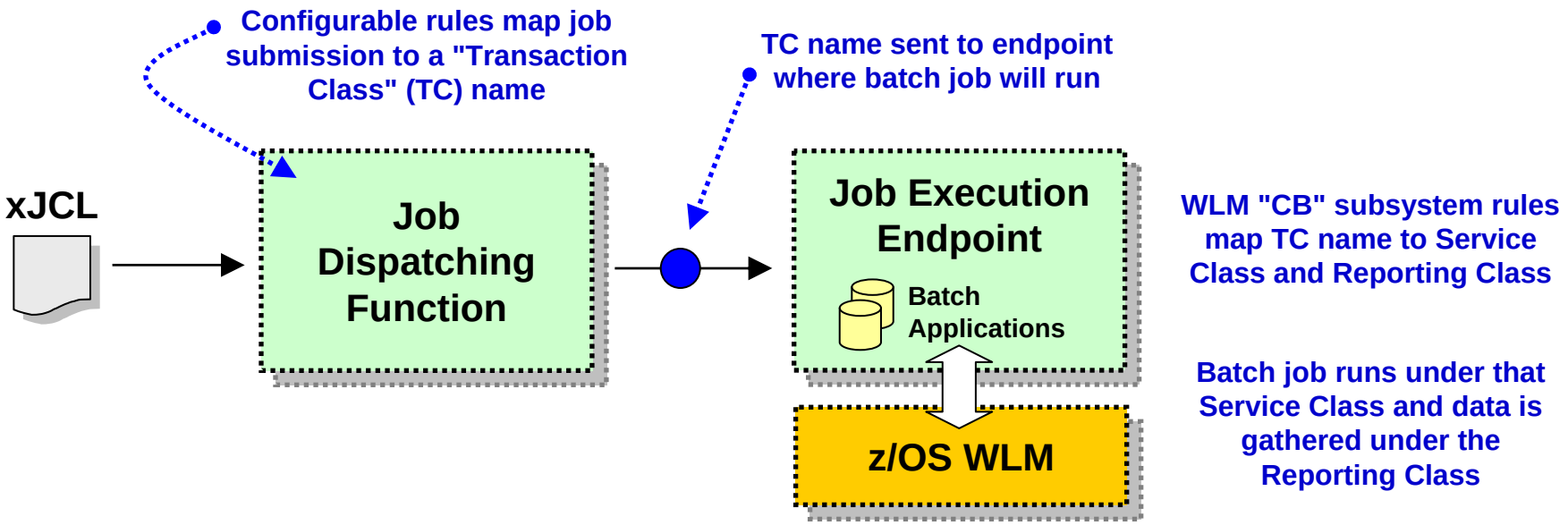
Keys to understanding value of zAAPs:

- zAAP processors have a considerably lower acquisition cost compared to GPs
- Offloading Java to zAAP frequently allows growing non-Java work to live within existing GPs, thus avoiding capital acquisition
- Monthly license charges based on capacity of the system can be influenced by the presence of zAAPs, which do not count towards charges

This is really a function of the Java SDK and the dispatcher of z/OS. The zAAP-enabled Java SDK is packaged with WAS z/OS, so WAS automatically takes advantage of zAAPs if they're present and configured

Workload Manager (WLM) Classification

The submitted job can be tagged with a WLM "transaction class," which may then be used to map the job to a WLM Service Class or Reporting Class:

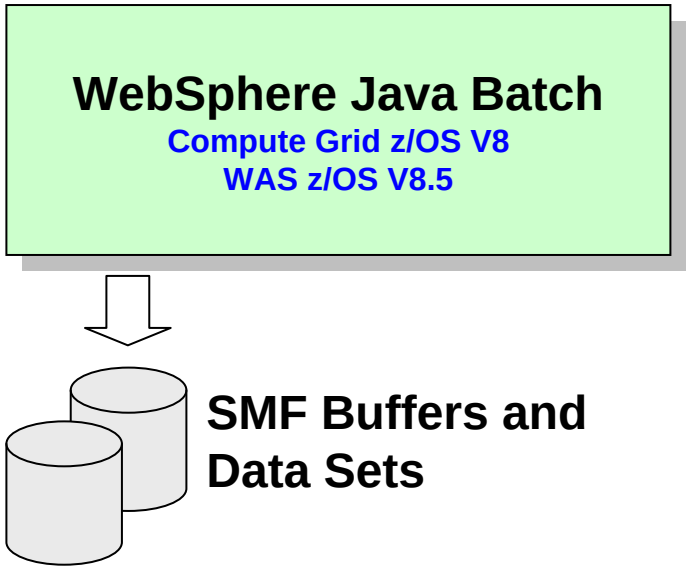


Classifying to a Service Class allows WAS z/OS to place work into separate servant regions based on Service Class

Classifying to a Reporting Class allows WLM to gather system information for all work running under that Class

SMF 120.9 Activity Recording

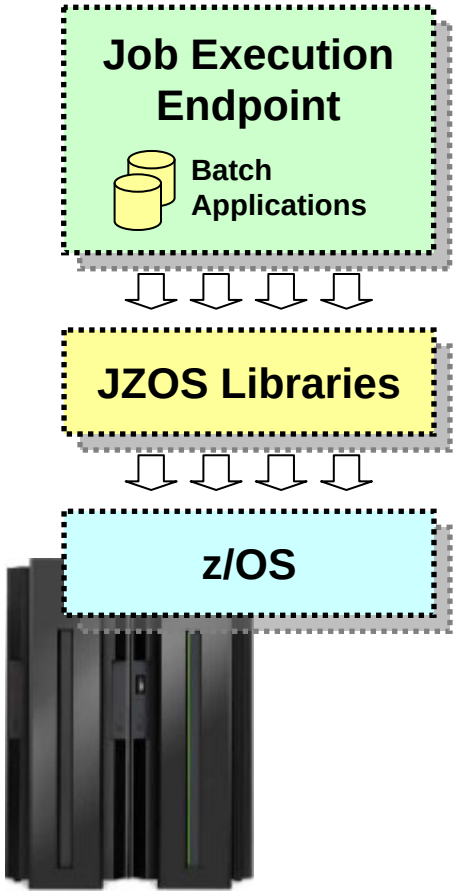
WAS z/OS supports the use of activity recording using the SMF 120.9 record. WebSphere Java Batch extends the record with batch activity information:



- Job activity records allow you to understand how your system is being used and to provide chargeback data
- Activity recording available on all platforms, but only z/OS uses SMF, which is an extremely efficient logging mechanism
- Provides historical records for usage analysis and batch capacity planning
- Information captured:
  - Job submitter
  - Date and time of submission
  - Final job state
  - Total CPU used for job
  - General processor used for job
  - zAAP usage derived: Total - GP = zAAP

Use of JZOS Services

JZOS is a set of functions that make using Java on z/OS much easier and useful. The JZOS class libraries may be used in batch application development:



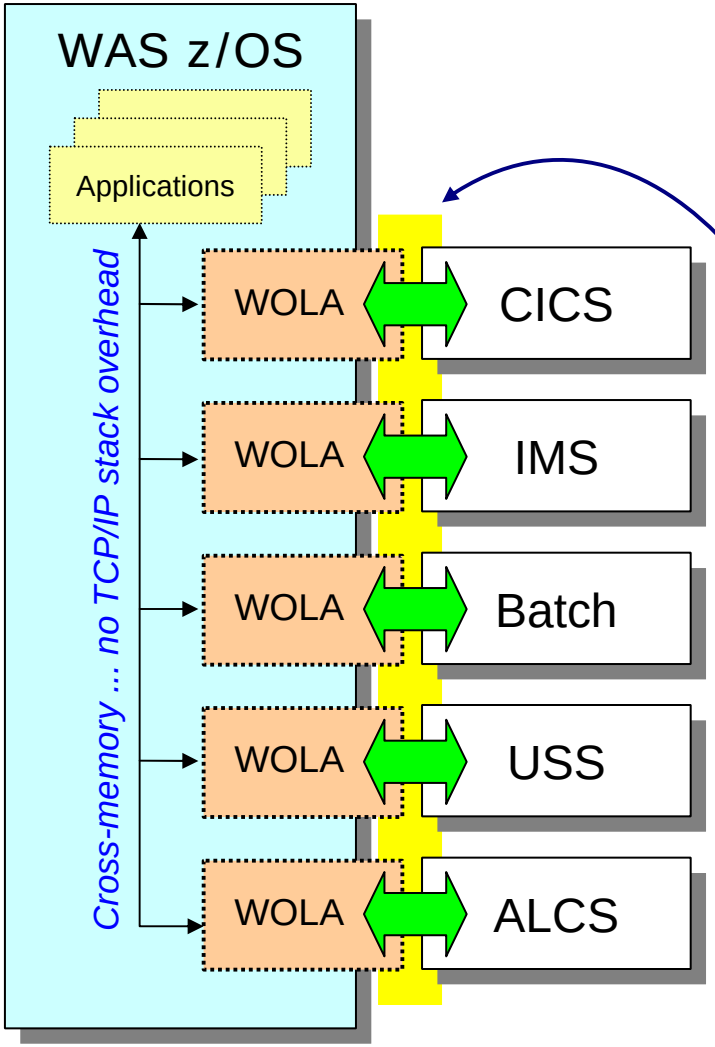
Examples of some z/OS services available:

- **DfSort** - interface for invoking DFSORT
- **MvsConsole** - class with static methods to interface with the MVS console.
- **MvsJobSubmitter** - class for submitting batch jobs to JES2 or JES3 from a Java program
- **PdsDirectory** - class for opening a PDS directory and iterating over its members.
- **WtoMessage** - simple data object/bean for holding a WTO message and its parameters.
- **ZUtil** - static interface to various z/OS native library calls other than I/O.

WebSphere Java Batch and JZOS are not mutually exclusive ... the JZOS class libraries may provide exactly what you need for your batch application to access z/OS functions and services

WebSphere Optimized Local Adapters

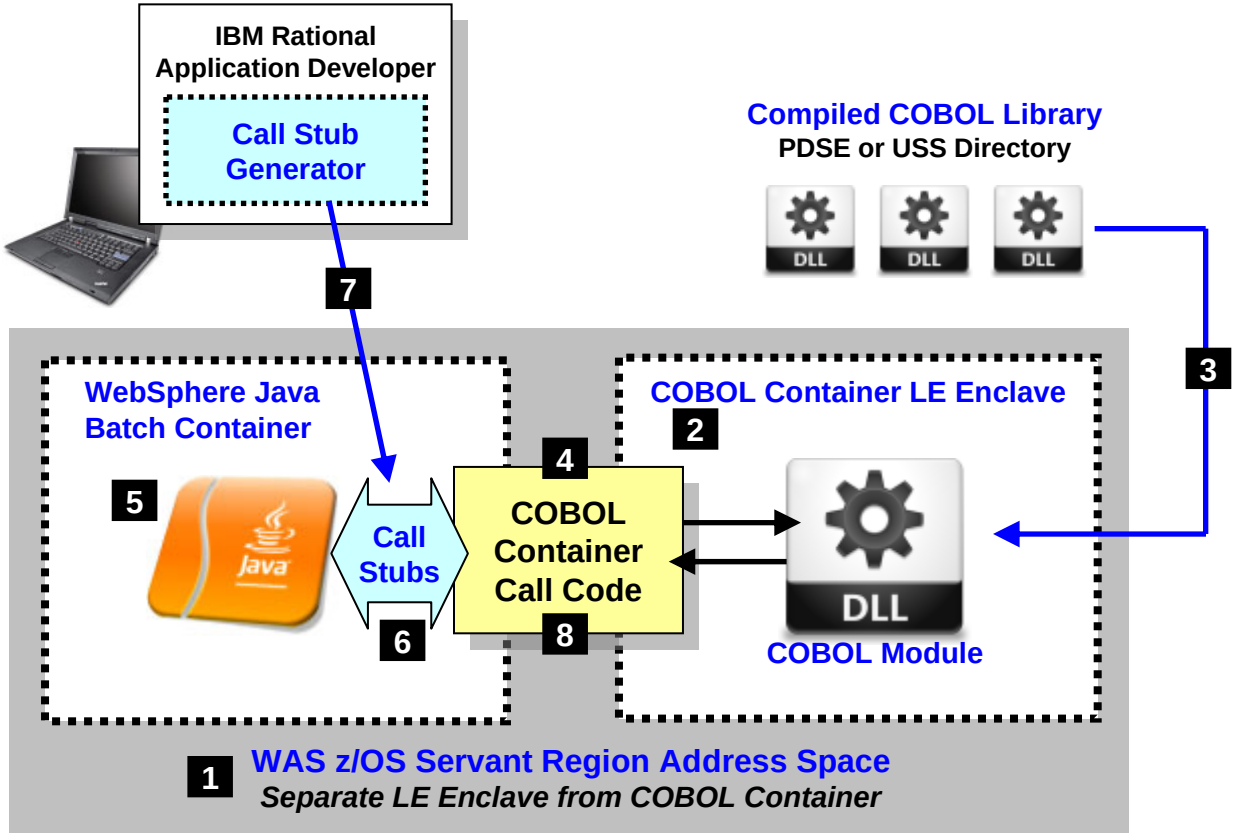
WOLA provides an efficient low-latency mechanism to exchange data bi-directionally between WAS z/OS and other address spaces:



- Leverage WOLA to call out from batch programs to co-located applications
- Very efficient byte-array transfer
- Bi-directional
  - Outbound -- Java in WAS invokes program in external
  - Inbound -- Program in external invokes Java in WAS
- Two phase commit, identity assertion
- Supplied JCA resource adapter for applications going outbound
- Supplied native APIs for cases where their usage is indicated
  - COBOL, C/C++, PL/I, High Level Assembler
  - 31-bit and 64-bit modules

COBOL Container

The COBOL Container provides a way to call and execute COBOL modules in the WAS z/OS server address space ... a *very efficient* way to call COBOL

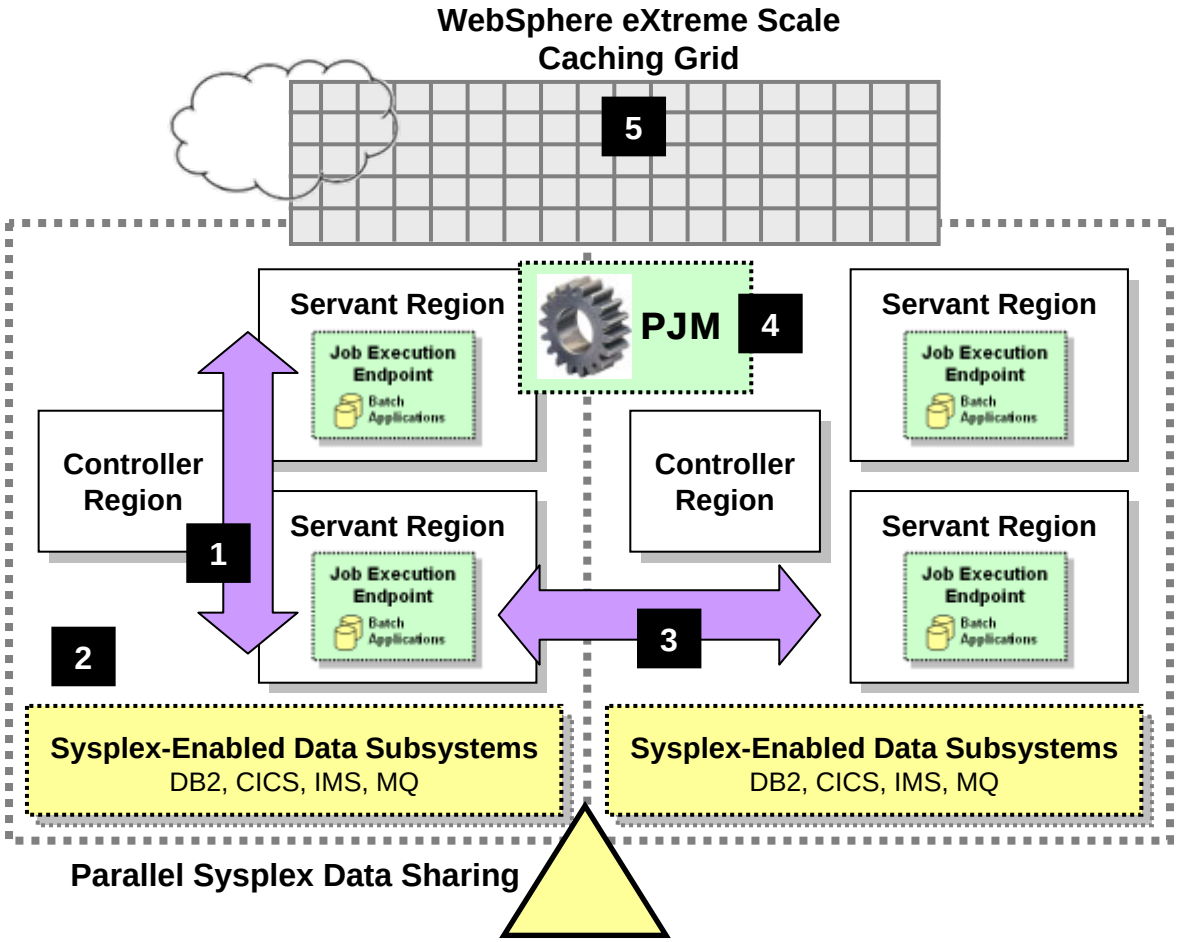


1. Batch application runs in the WAS z/OS servant region address space
2. The COBOL container is created as a separate LE enclave in the address space
3. COBOL DLLs are accessed using STEPLIB or LIBPATH
4. COBOL Container code provides the "glue" between the Java environment and the native COBOL
5. Java batch code uses supplied class methods to create the container and use it
6. Call stubs provide an easy way to call the COBOL DLL and marshal data back and forth
7. The call stubs are generated by a supplied utility that uses COBOL source to understand data bindings
8. JDBC Type 2 connections created in the Java batch program may be shared into the COBOL module in the COBOL Container

Lines of code needed to invoke COBOL many times less than other means of calling COBOL from Java

Scaling the Java Batch Solution on z/OS

There are several ways in which a WebSphere Java batch solution can be scaled up to provider greater batch throughput and shorter execution windows:



1. Vertical

WAS z/OS servant regions provide a type of "vertical cluster," giving you additional batch compute resources

2. Capacity on Demand

CPU processors may be dynamically added to a z/OS LPAR, increasing the capacity for processing work

3. Horizontal

WAS z/OS clustering on top Parallel Sysplex provides near-linear scalability up to 32 nodes with a central data sharing model

4. Parallel Processing

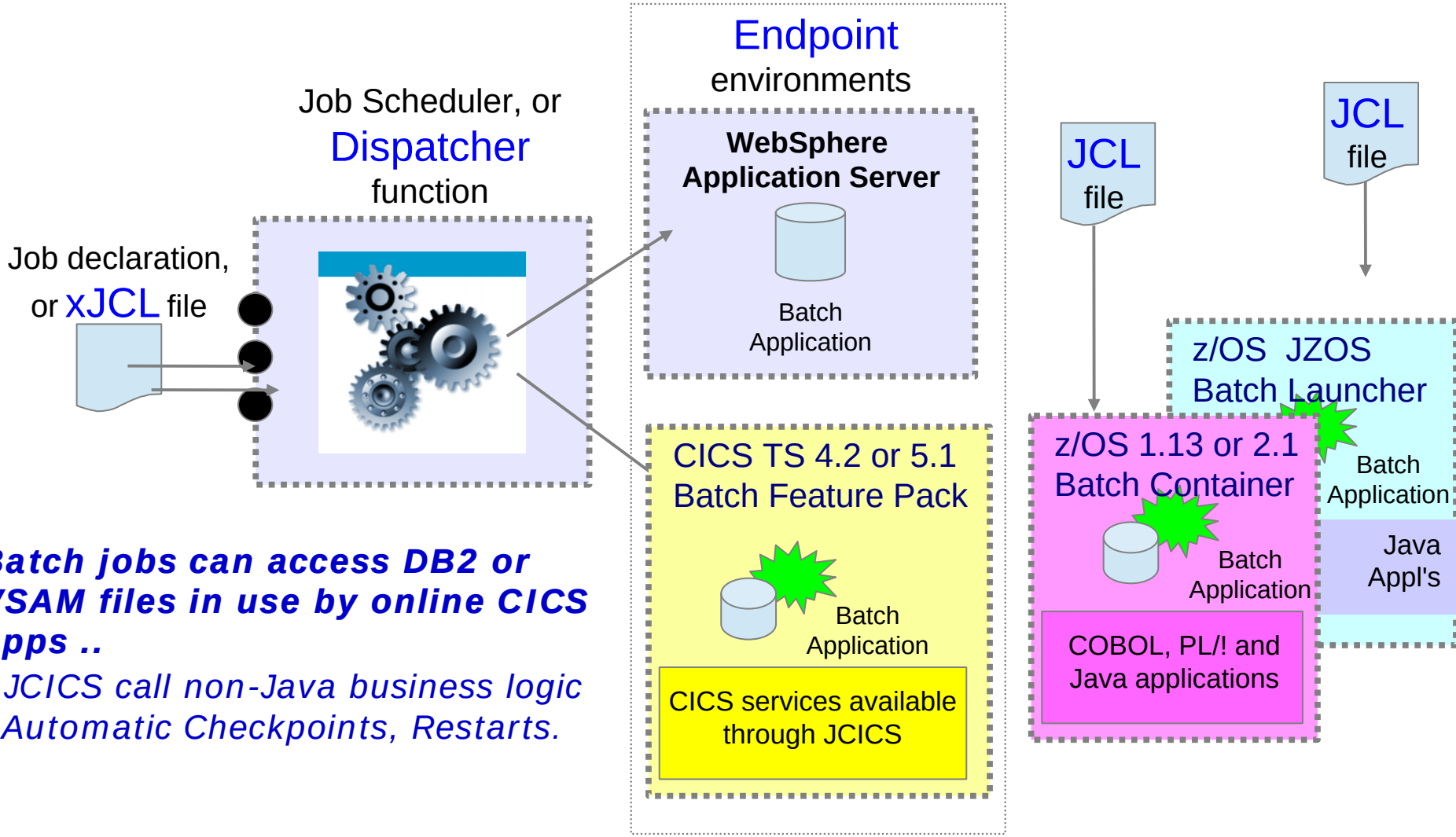
The Parallel Job Manager may be used to partition data into sub-jobs, which may then be run on multiple threads, different servants, or different servers on other LPARs.

5. Data Caching

WebSphere eXtreme Scale provides a data caching grid from which Java batch may fetch and store data

"Batch Containers" are not Limited to WAS

There's also a Java batch container for **CICS** ... and **z/OS**



Batch jobs can access DB2 or VSAM files in use by online CICS apps ..

- *JCICS call non-Java business logic*
- *Automatic Checkpoints, Restarts.*

Enabling Incremental Adoption

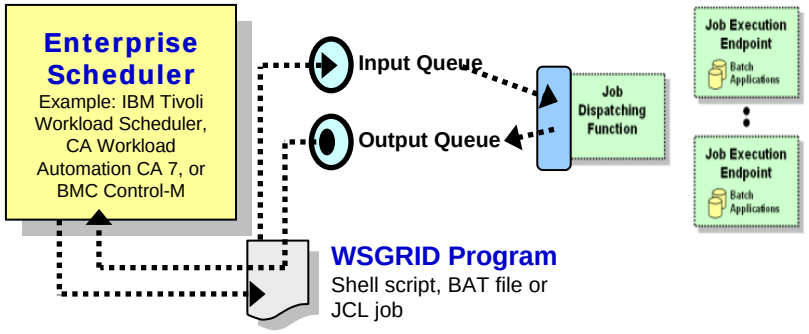
An *evolution* not a *revolution*

A primary benefit of WebSphere batch is the ability to introduce Java batch to existing batch processing over time in an incremental fashion

- Customers have thousands of existing legacy batch applications
- In many cases it would not be feasible to replace significant portions of the existing batch processing at a single time
- WebSphere Batch is designed with key features to facilitate a step-wise approach to Java batch modernization
- Integration with existing processes, applications, operating system services and technologies allows WebSphere batch to be introduced incrementally, while still providing immediate value

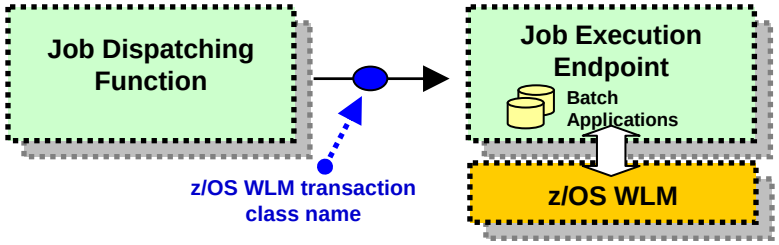
Key capabilities for incremental adoption

Integration with Existing Processing



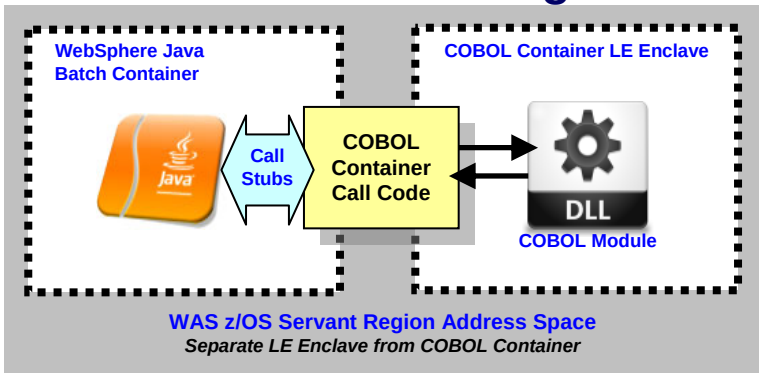
Integration with enterprise schedulers enables WebSphere batch to be introduced without disruption into existing processes

Prioritized Workload Management



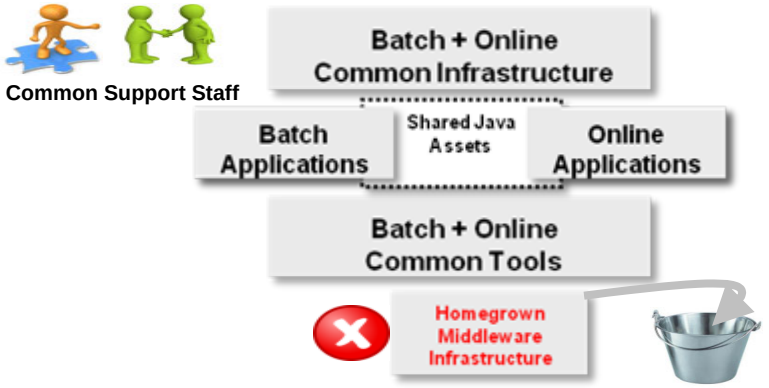
Tight integration with z/OS Workload Manager enables policy based prioritization of batch and online workloads to achieve business goals

Java and COBOL Integration



The close proximity and shared database connections enable an extremely efficient and effective means of calling COBOL from Java

Leveraging Shared Infrastructure



The ability to share services and infrastructure between batch and online processes allows for step-wise convergence of batch and online

Wrap-Up and Summary

WebSphere Batch Summary

Evolve to a single infrastructure for both online and batch that enables you to leverage existing applications and focus resources on business logic

✓ Improve batch execution efficiency

- **Parallel Batch Jobs** – Execute a single large batch job that is broken into chunks and executed concurrently across a grid of resources.
- **Dynamic Online & Batch Runtime** – Dynamically provision resources as capacity changes to meet operational goals.

✓ Leverage common skills and IT resources

- **Replace Homegrown Batch Frameworks** – Migrate from a native batch runtime (e.g. C / C++, PL/I, and COBOL) to Java and reduce costly proprietary batch infrastructures and focus development resources on creating business value.
- **Share business logic across Online and Batch** – Leverage the proven WebSphere platform to share logic across both batch and online, reducing maintenance and development costs.

✓ Optimize the cost of batch and online

- **Leverage System z Specialty Processors** – Offload Java workload from GP to less expensive zAAP processors, to gain processing capacity and
- **Batch as a Service** – Expose business capabilities as a service and leverage usage accounting features for tracking and chargeback, to

Real Customer value from WebSphere Batch

A leading wholesale provider of reinsurance, insurance and other insurance-based forms of risk transfer.

Business Drivers:

- **Incremental adoption** of Java batch job architecture and dependencies
- **Reduce costs** of batch execution by exploiting System z specialty engines
- Take advantage of more readily available **Java skills**

Overview of Usage:

- Existing batch with a very large base of **COBOL batch** (~21,000 COBOL modules and 40,000 batch jobs per day) with considerable interdependencies
- Leverage **COBOL Container** to allow Java jobs to more efficiently call existing COBOL assets
- Share **JDBC T2 connections** between Java and COBOL for one transactional unit of work
- Leverage **WSGRID** to integrate with existing enterprise scheduler

Business Value:

- **Incremental modernization** through integration of COBOL / Java and enterprise scheduler
- **Reduced maintenance and dev costs** by migrating to newer WebSphere technology
- **Streamlined coding efforts**, encouraging technology innovation and faster development
- **Increased overall server performance**, helping reduce associated software licensing fees

Real Customer value from WebSphere Batch

A major global financial institution.

Business Drivers:

- Take advantage of newer development tools and developer skills
- Increase agility ... faster time-to-market for changes

Overview of Usage:

- Month-end, quarter-end and year-end batch jobs
- WebSphere Java batch spread across 6 LPARs, with heavy interaction with DB2
- WSGRID for **integration with existing enterprise scheduler** function
- **Parallel Job Manager** for shortened overall completion time
- Workload balancing via **z/OS WLM integration** for SLA goal mapping

Business Value:

- **Increased application agility**, and time-to-market for requirements changes
- **Reduction in JCL jobs from 22,000 to less than 100**, with most changes now parameter driven
- More flexible and dynamic document generation enabling adherence to regulations
- Approx **70% Java processing offload to zAAP** specialty engines improving G-CPU capacity
- **Scalable** both up and out utilizing WebSphere Java batch as an underlying execution platform

WAS on z/OS and Java Batch Collateral

Topic	Link
Guide to WebSphere on z/OS Collateral – Updated master list of links to collateral	http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102205
WebSphere Java Batch – Overview, z/OS Specifics, Quick Start – Presentation, whitepaper, videos	http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101783
Why WebSphere Application Server for z/OS – Executive Brochure – History of release enhancements – Technical Presentation, videos	http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101532
WAS for z/OS Liberty Profile – Executive Brochure – Quick Start Guide and Samples	http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102110
WebSphere Optimized Local Adapters (WOLA) – Overview, whitepapers, videos – History of WOLA updates	http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101490
Training – z/OS Wildfire Workshops – WAS for z/OS v8.5 – WebSphere Compute Grid (WebSphere Batch)	http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1778
WebSphere on z Virtual User Group – Join the User Group – Download previous webcasts	http://www.websphereusergroup.org/zos