WebSphere Application Server

IBM Batch Modernization

A survey across several broad topics

WP101783 at `ibm.com/support/techdocs`

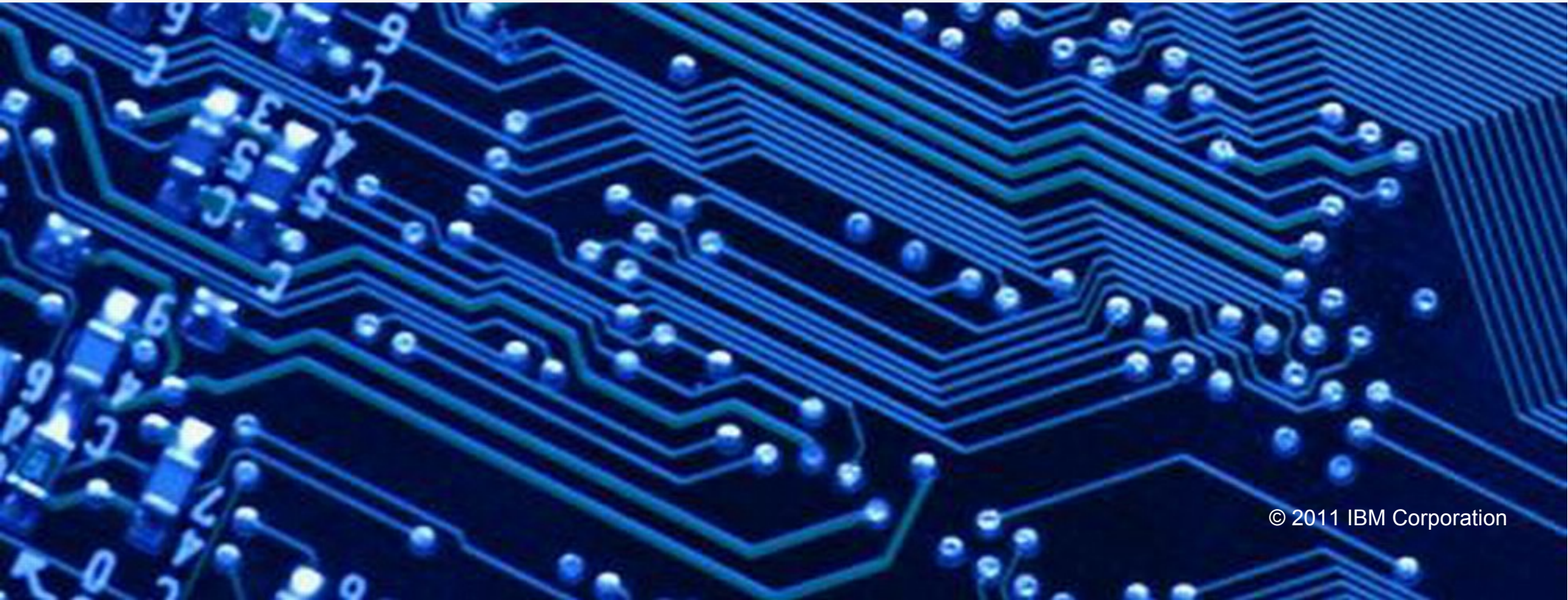


Table of Contents for this Presentation

Click to go directly to that page

Preview of the Message

Java batch is a growing reality with good reasons to pursue

Java batch technologies have different degrees of capability

Be aware of the "custom middleware trap"

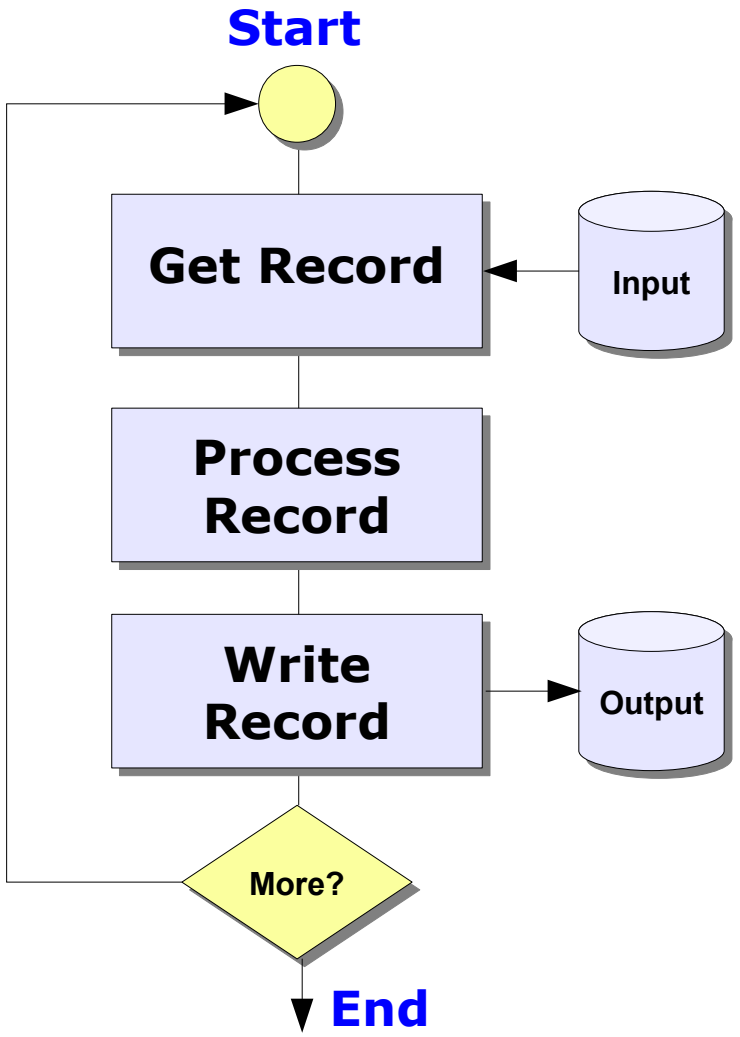
IBM's Java batch story:

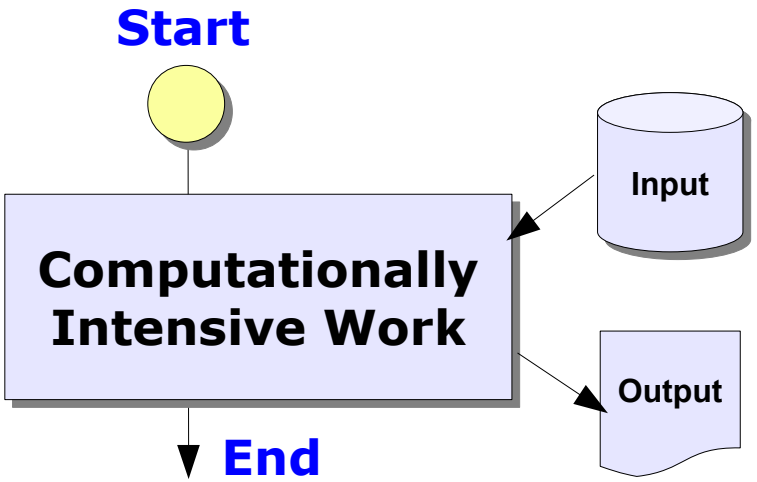
- ***Two products: Feature Pack for Modern Batch and Compute Grid***
- ***Feature Pack is no charge and functionally a subset of Compute Grid***
- ***Compute Grid is full function priced program product***
- ***Both based on WebSphere Application Server***
- ***Both available across all platforms***
- ***Compute Grid on z/OS has additional abilities to exploit the platform***

Setting Context

Batch Processing

Has been around for a very long time. Traditionally it's used to process large amounts of data in a repetitive way. But there's also "compute intensive" as well:





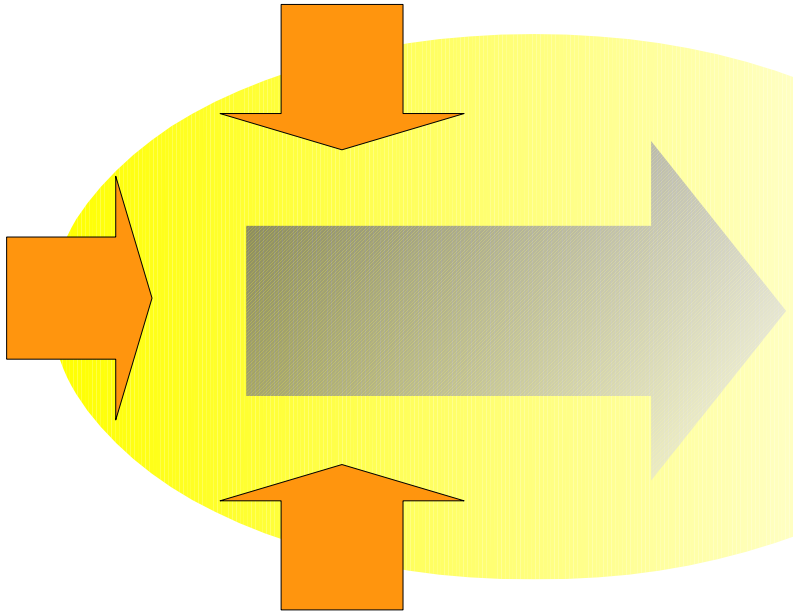
Both types are still very heavily used and under review as part of a broader "modernization" effort

"Modernization"

A fancy term for the reality of constantly reviewing existing processes in light of the pressures facing you and your business objectives:

Cost Pressures

Competitive Pressures



Response

- **Change**
- **Improve**
- **Update**
- **Modernize**

Time Pressures

Common Motivators to Consider Java Batch

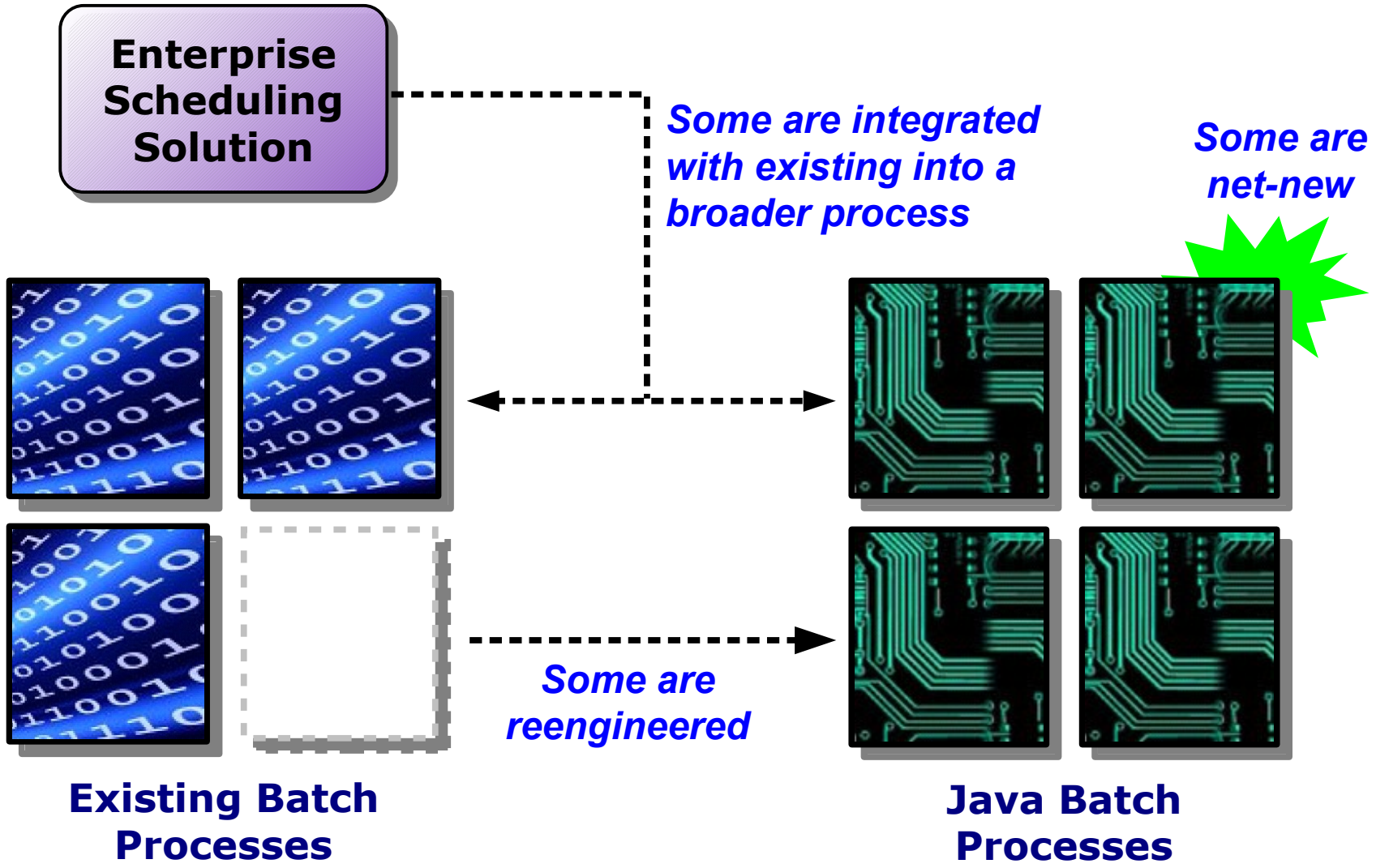
A summary of things we see:

- **Leverage the specialty processors on System z**
  - Java processing can be offloaded to the zAAP, lowering the overall cost profile of running batch processing on the mainframe
- **Focus development skillset around Java**
  - As Java skills become more prevalent and COBOL skills less, the motivation is to focus development effort around a common programming skill
- **Reuse business logic between OLTP and batch**
  - To create a more singular design and code stream, share skills across different classes of workloads, and to streamline the build / test / deploy process
- **Integrate batch processing with OLTP**
  - As a means of extending batch window and to share / balance activity and system resources within a common execution environment
- **Expose batch processes as SOA services**
  - As a means of integrating batch into a broader SOA architecture

Thoughts or discussion?

Not a "Rip-and-Replace"

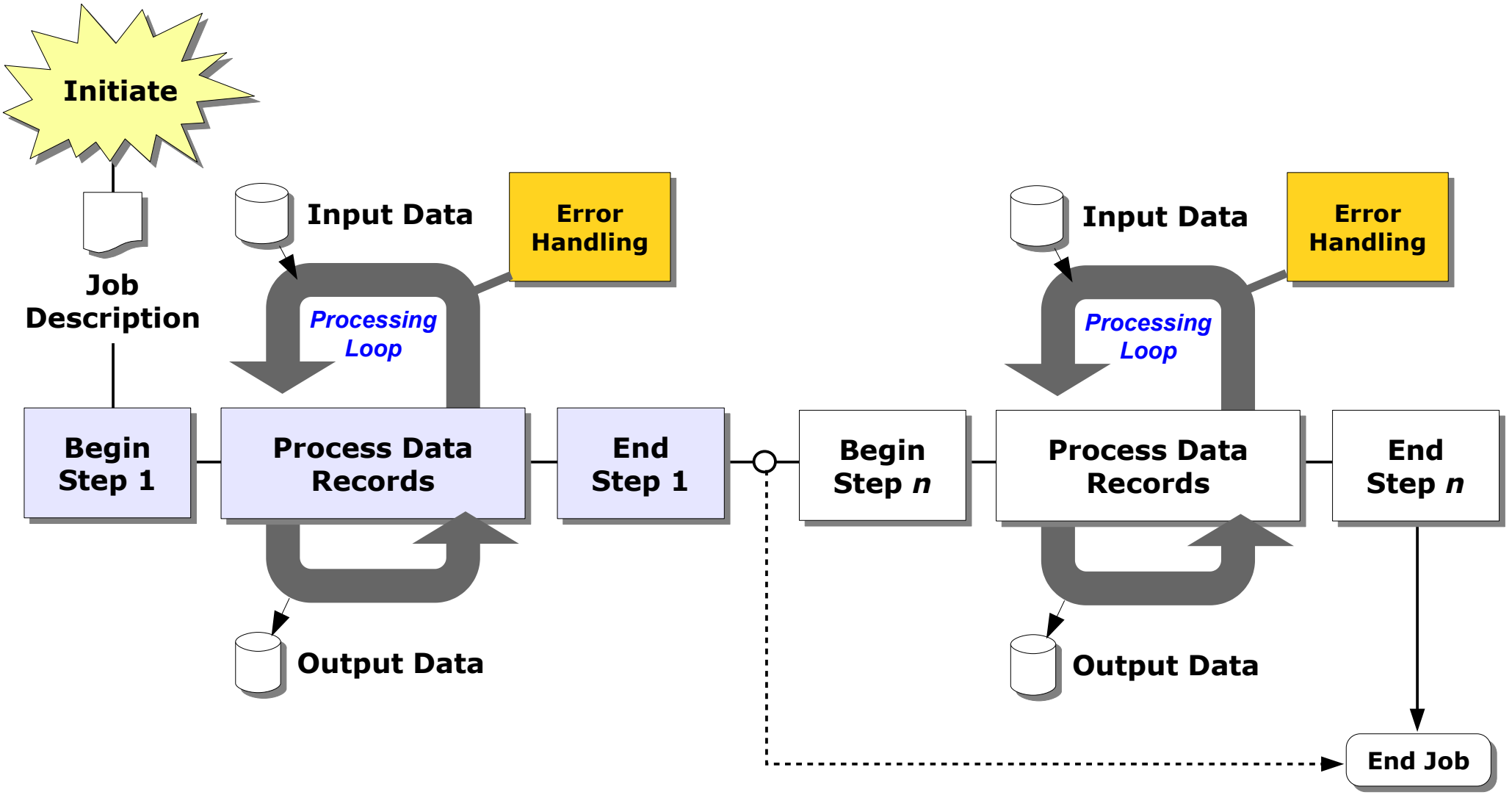
We wish to emphasize that we're speaking of a reasoned evolution here, where the needs of the business are key:



The Batch Lifecycle

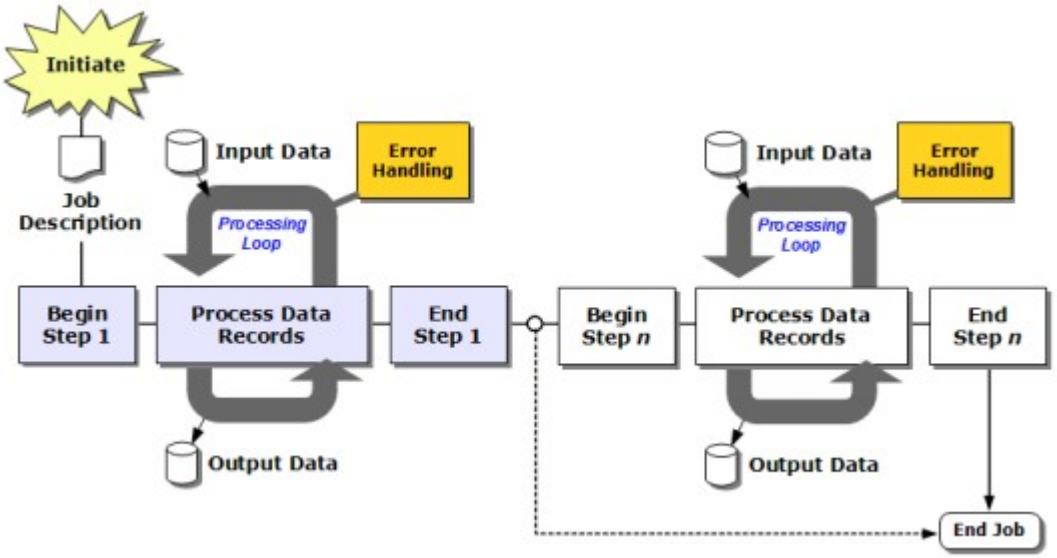
The Anatomy of a Batch Job

This is a very schematic representation of what takes place:



Further Considerations

Some things to think about as you ponder Java batch processing:



How is the job initiated?	**Any transactional updates taking place?**	**Do you need integration with enteprise schedulers?**
What will be mechanism for overall job description	**Two-phase commit required?**	**What accounting information is needed?**
Need for conditional step processing?	**How is checkpoint processing handled?**	**How is WLM classification performed?**
What coordination is done among all jobs?	**Can you suspend processing?**	**Does process lend itself to parallelization?**

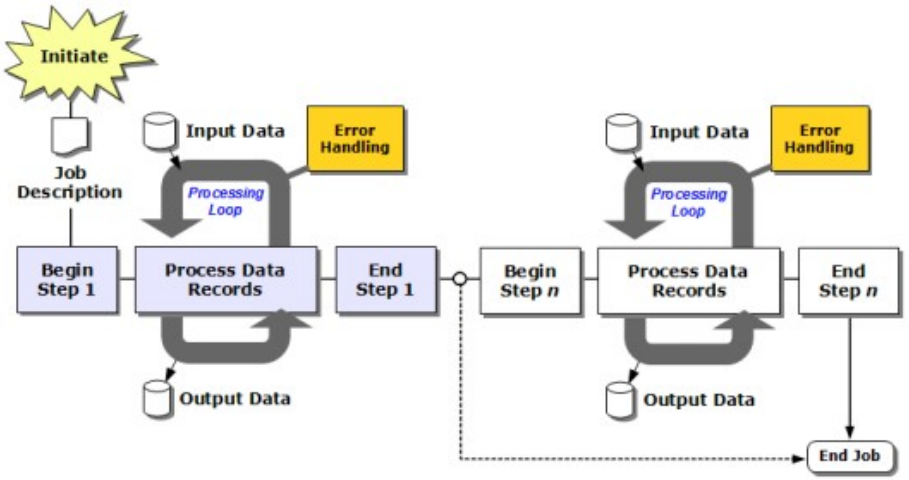
Business Function vs. Support Function

Batch processing does not take place in a vacuum. The core business function being done relies on support functionality to make it work.

Batch code that is doing the core business processing against the data

Batch processing function in support of the core business batch processing

- **This is whatever the batch program does that *directly* contributes to the business objectives**
- **This is function that provides the batch needs we discussed on the previous page:**



For thought or discussion ...

- What is your comfort level with going down the custom middleware path?
- How will you keep your custom middleware from becoming isolated islands of functionality?

We ask because the Java batch solutions have differing levels of support

Java Batch Technology Options

Three Essential Java Batch Models

For consideration and discussion:

JVM Launcher

- Tools that instantiate a JVM and invoke the Java program
- JVM terminates at completion of batch program
- Provide a degree of batch support
- Examples: BPXBATCH or JZOS

Development Framework

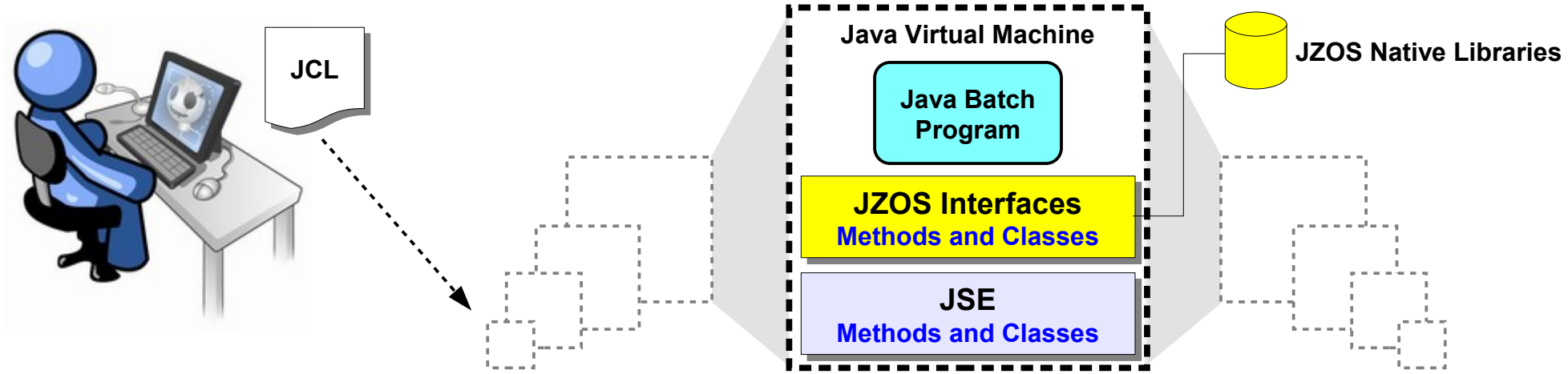
- Provides a set of batch functionality in the form of supporting class libraries and development libraries
- Limited integration with underlying middleware or platform
- Examples: Spring Batch

Execution Platform

- Provides a set of batch functionality in the form of supporting class libraries and development libraries
- Specific integration with underyling middleware or platform
- Examples: Feature Pack for Java Batch, Compute Grid

JVM Launchers

Instantiate a JVM and invoke the specified Java program within the JVM. On z/OS the model looks like this:



JVM Initialized ⇨ Java Batch Run ⇨ JVM Torn Down

Advantages:

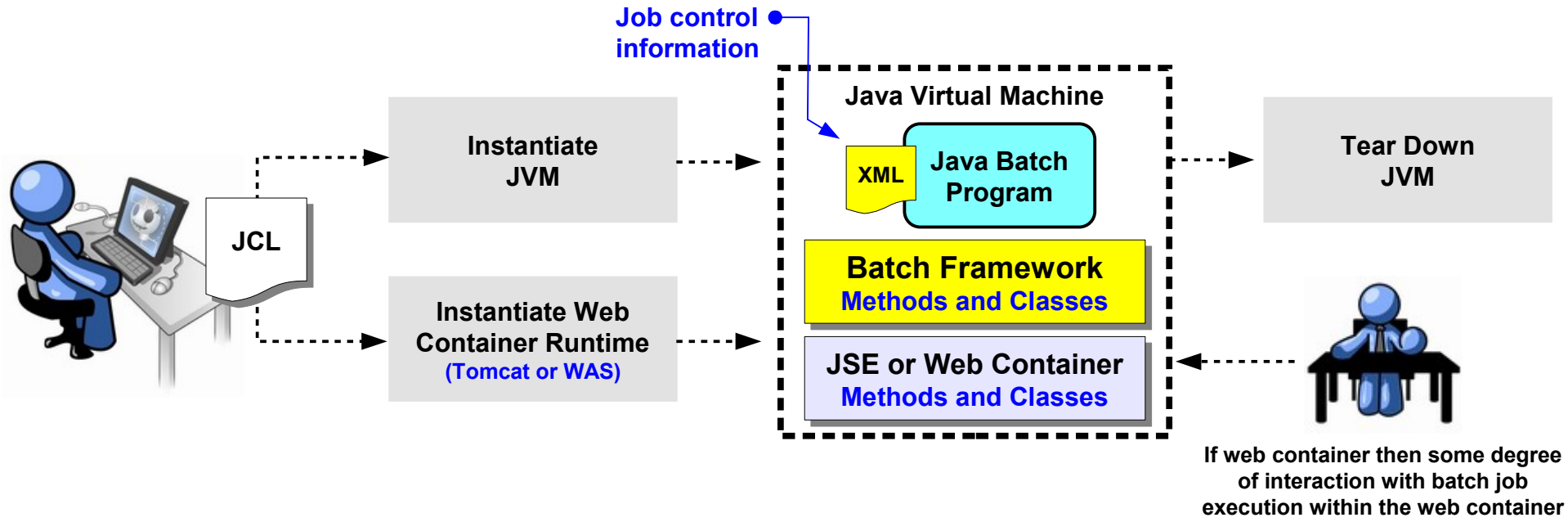
- **Simple to use**
- **Included with z/OS**

Disadvantages:

- **Overhead of repeated instantiation**
- **Provides limited batch support functionality**
  - **JZOS represents a substantial improvement over BPXBATCH, but provides a relatively small set of batch functional support**

Java Batch Development Framework

Provide a set of class libraries that provide batch functionality to a program:



Advantages:

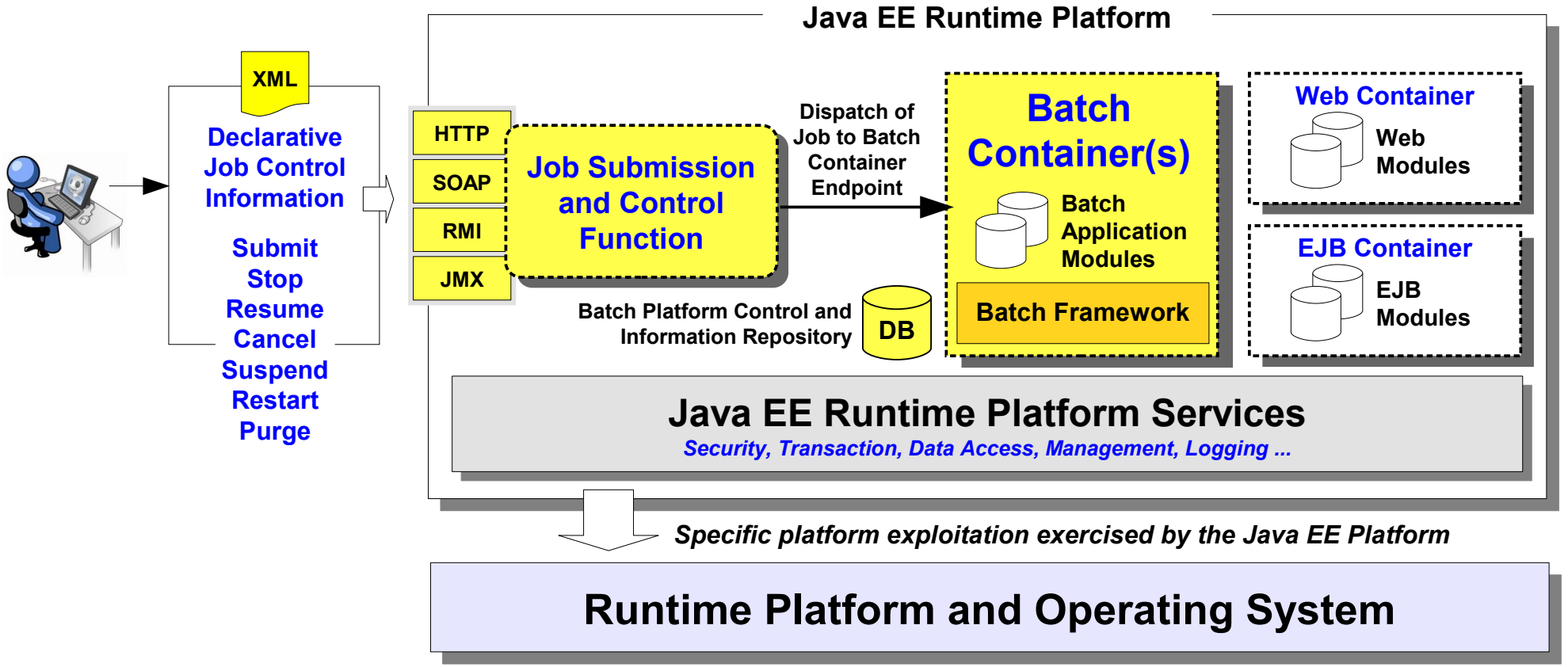
- More functionality than JVM launchers
- Allows greater focus on batch processing and less on building your own custom framework

Disadvantages:

- Limited to JSE or web container
- Uses available JSE or web container function but offers little integration beyond that
- Most common are open source software
  Whether or not that is a concern to your organization is something you must consider when commiting to a strategic investment in the technology.

Java Batch Platform

Provide the framework and integrates with the underlying middleware and platform:



Advantages:

- More functionality
- Greater integration with platform services
- Creates managed container environment for Java batch

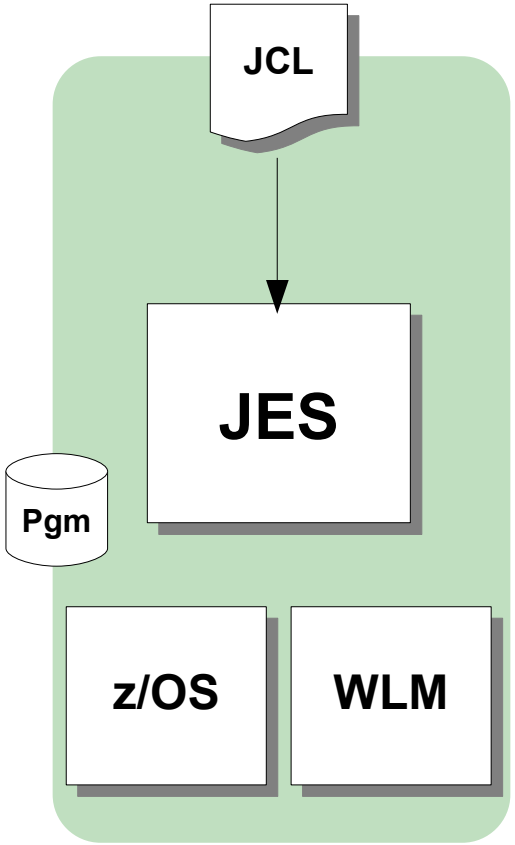
Disadvantages

- Requires a Java EE platform on which to operate and integrate

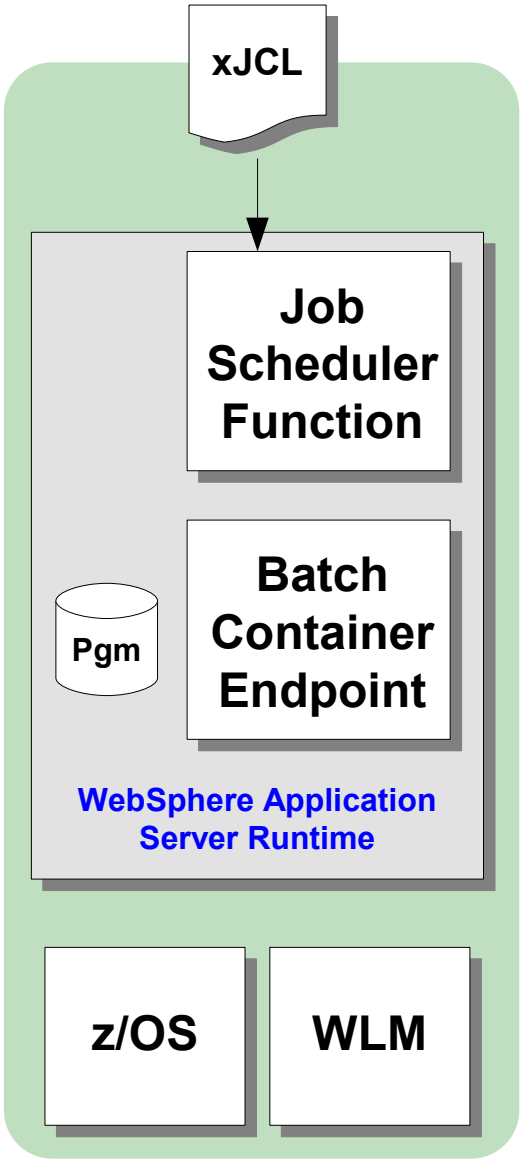
IBM Java Batch Solutions

The IBM Java Batch Structure, *Very High-Level*

It's all about leveraging the proven foundation of WebSphere Application Server while maintaining the proven methodologies of tradtional batch:

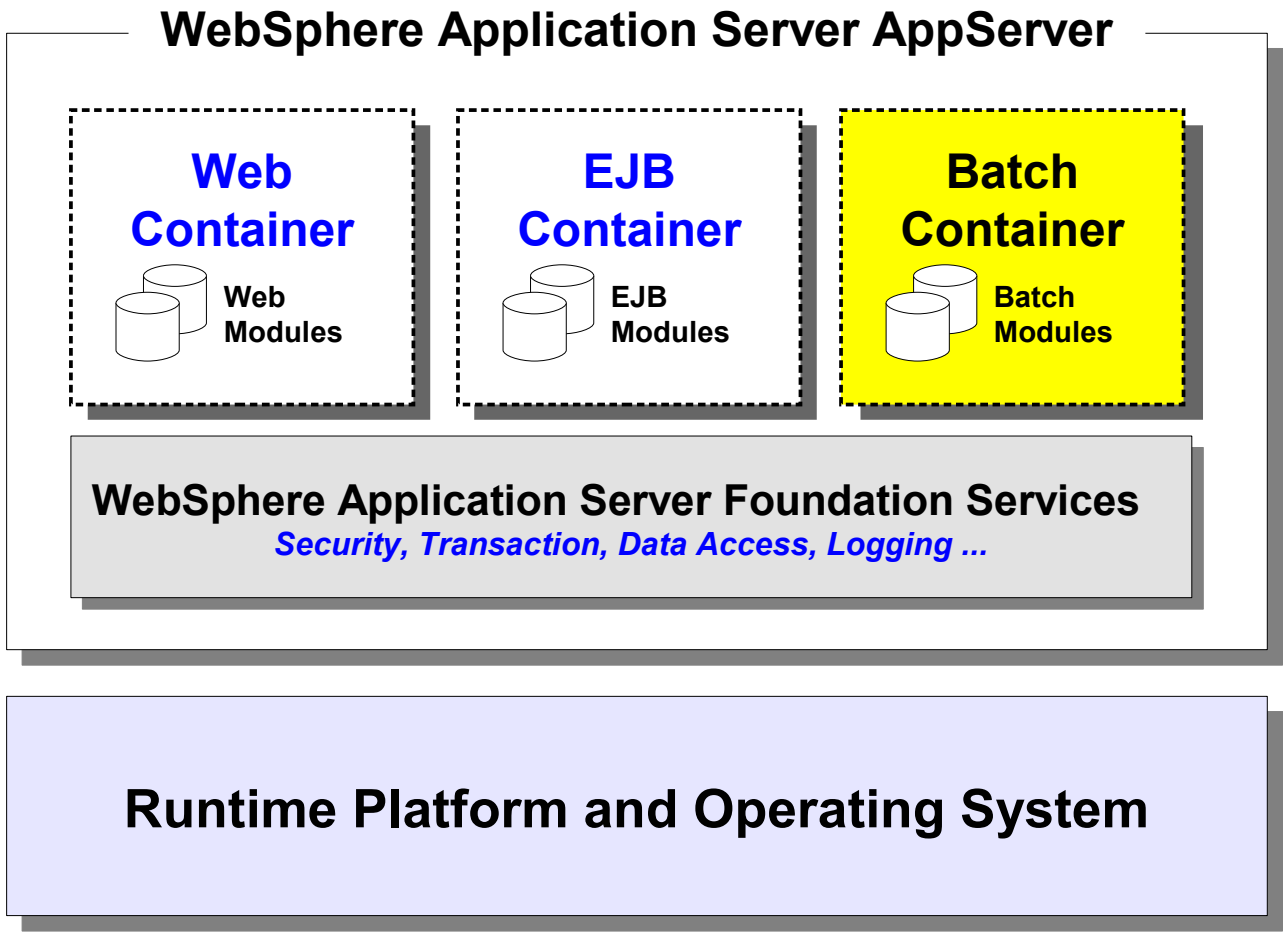


Means of describing the structure and details of the job to be submitted
Means of separating application code from the job declaration and submission
Means of interpreting the job declaration and preparing the job for execution
Means of separating submission from execution with some awareness of system
Means of intelligently dispatching to one of several execution endpoints
Means of viewing and controlling processing during execution
Means of collecting and maintaining job output and results



The Concept of a "Batch Container"

WebSphere Application Server already has two pre-packaged "containers." The IBM Java batch solutions add to that by providing a batch container.



The benefit of this is it provides a structured, managed batch environment *within the broader WAS environment*

Container management is a key element of the WAS design. By extending it with a batch container IBM is staying consistent with the overall platform strategy.

All the benefits of WAS itself accrue upwards to the batch runtime.

What's the Objective?

In a picture and a few words:

Allows you to maintain a focus on your core business objectives while leveraging Java as a batch execution language

Helps you avoid writing expensive and hard-to-maintain custom batch middleware functionality

Makes you a hero in the eyes of management, who's focused on the business imperatives

Funny pictures ... *serious point*. Countless hours and dollars have been spent by companies pursuing custom middleware solutions. It's costly, time consuming and may lead to disparate islands of Java batch processes

A Multi-Platform, Multi-Tiered Strategy

From our earlier review ...

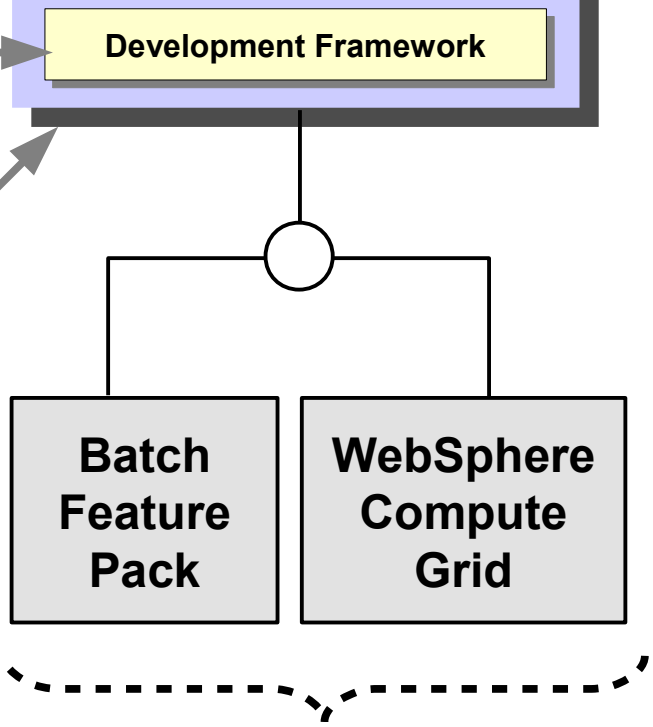
JVM Launcher

- Command line Java invocation or shell script automation
- BPXBATCH
- JZOS

Development Framework

- IBM does not provide a separately packaged development framework as such
- The "Execution Platform" includes a development framework as part of that offering
- The JZOS function can be made accessible to applications in the batch container

Execution Platform



Spans all supported WAS platforms, with differing degrees of capabilities.

We explore this as our focus

Feature Pack vs. Compute Grid

It's a subset / superset relationship:

IBM WebSphere Compute Grid

We'll explore some of these in upcoming charts

Feature Pack for Modern Batch

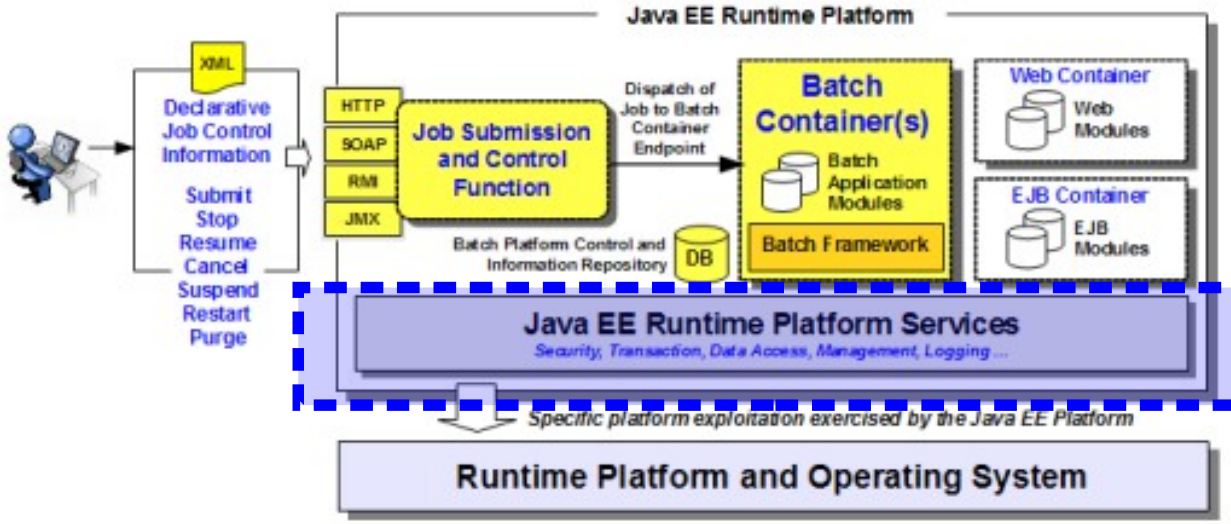
- No charge feature pack
- The essential batch container framework
- Same essential job submission and job monitoring functinality
- A *subset* of the more robust Compute Grid

- *Additional enterprise functionality*
- Separate FMID, separate charge
- Migration path from the Feature Pack

The underlying strategy is to provide an onramp to batch container functionality in a low-risk, low-impact manner. The expectation is that once customers see the benefit of the model, they'll migrate up to Compute Grid.

Both Are Built on WAS Foundation

The Feature Pack for Modern Batch and WebSphere Compute Grid are both extensions to the WAS foundation:



This is WebSphere Application Server

The Feature Pack and Compute Grid were designed to leverage the *existing functionality* of the WAS platform

The Feature Pack and Compute Grid bring *additional functionality*, but do not duplicate existing function

Installation is a relatively simple matter of "augmenting" the WAS environment with the additional code

IBM Batch Container Model ... Demystified

Mapping against our picture from earlier:

Job Submission and Control Mechanism

Supplied system application that offers several user interfaces, interacts with the database, and understands what batch applications are deployed in which endpoints

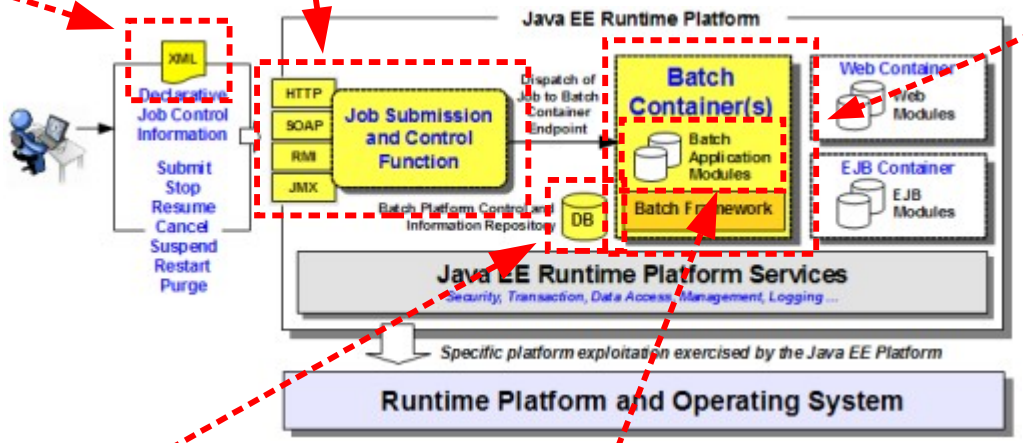
Batch Container Environment

Another supplied system application, this provides the interface to the scheduler function, interacts with the database to maintain job status, and provides the interface point for job control

xJCL

Relatively simple XML structure that describes the job characteristics -- programs, steps, conditional processing, checkpoint algorithms, data streams and variable substitution.

Exactly the same in concept with JCL, just a different syntax.



Database

A set of relational tables (several formats) that provide a central point of organizational control for the submission, monitoring and completion statistics for submitted jobs

Batch Applications

Separate from the "job," which is a given invocation of the application. The batch application is written as a POJO, using the Batch Data Stream (BDS) programming framework, and under the control of a supplied asynchronous bean facility

What Does it Provide?

Here's a summary of the key features:

Feature Pack for Modern Batch

- Batch container environment
- Job scheduler and dispatcher function
- Declarative job control file (xJCL)
- Development class libraries
- Batch Data Stream (BDS)
- Conditional multi-step job support
- Checkpoint processing leveraging WAS transaction manager

Go to details page

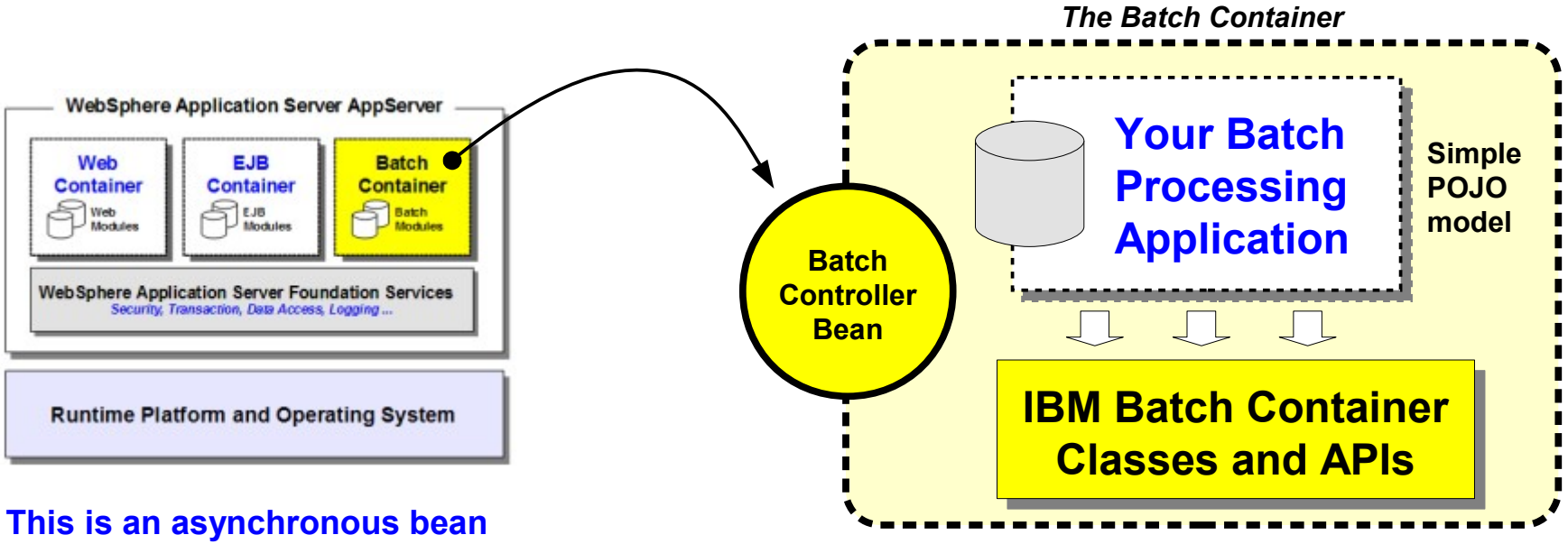
WebSphere Compute Grid

Everything you see under "Feature Pack for Modern Batch" plus ...

- Calendar and clock scheduling of jobs from xJCL repository
- Mechanism for integration with enterprise scheduler products *Go To Page*
- Usage reporting with SMF 120.20 records (z/OS only) *Go To Page*
- WLM transaction classification *by job* (z/OS only) *Go To Page*
- Application quiesce and update
- Job submission pacing and job execution throttling
- Parallel job management and dispatching *Go To Page*

How Does it Provide This Function?

Batch by its nature has an long running execution profile. Therefore it can't be run under the traditional request / response model. So it uses asynchronous beans:

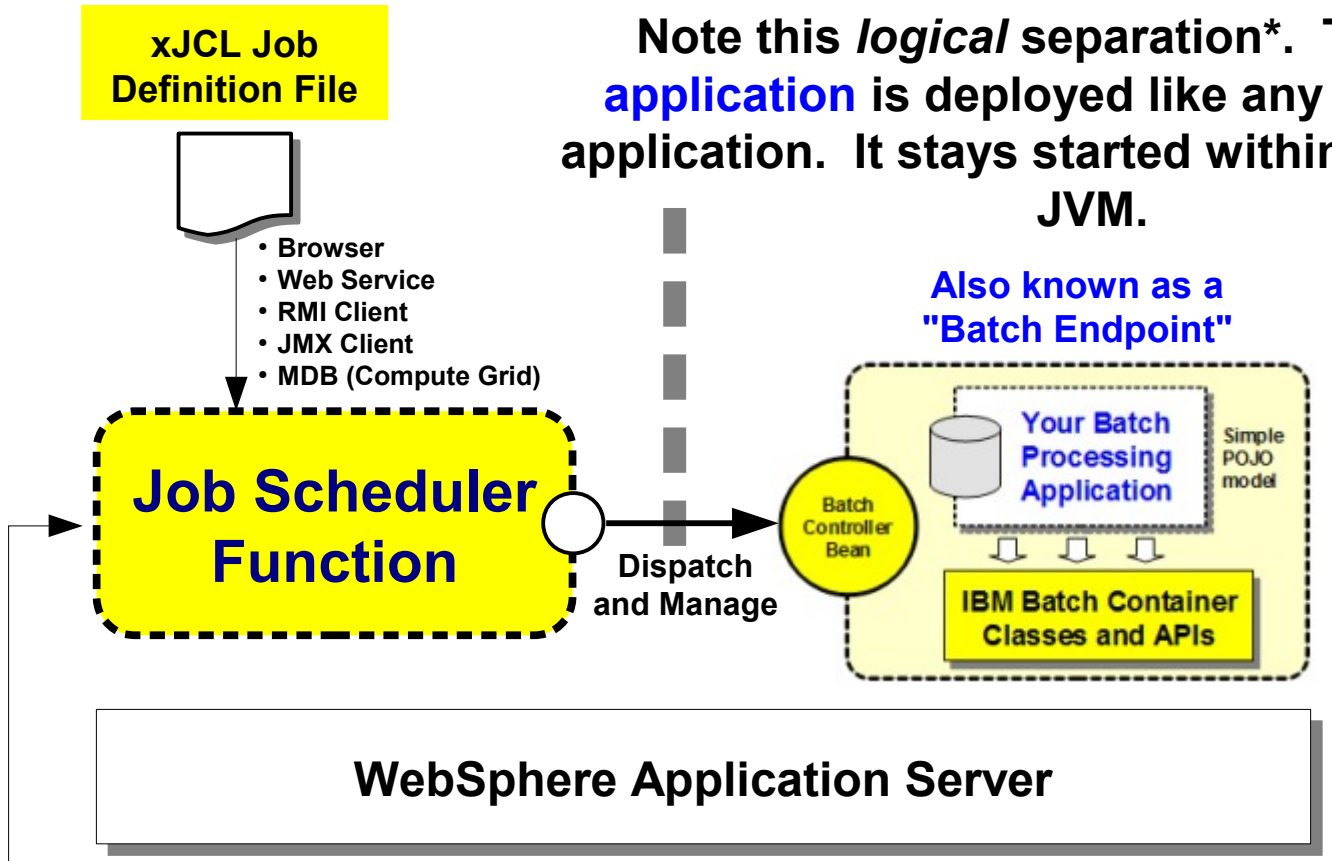


- This is an asynchronous bean
- Your batch application runs under the control of this bean
- You can think of this as a container-managed thread
- It processes the job definition and carries it from start to finish

If you tried to roll your own batch environment in WAS you'd end up using an asynch bean structure and writing a lot of custom middleware code to support it. Here your work is limited to your application logic. IBM provides the middeware structure.

Job Scheduler, and Separation of Job / App

A key concept to appreciate the differentiation of this over other Java batch models:



Note this *logical* separation*. The batch application is deployed like any other WAS application. It stays started within the running JVM.

Also known as a "Batch Endpoint"

The job is an instance of program invocation based on the description offered in the xJCL file

That may involve one deployed batch application *or several*

Do you see what's going on here? Your batch applications become a set of resident re-usable batch class objects

The xJCL describes the job, which may contain multiple steps employing multiple batch classes

The asynchronous batch controller bean provides the managed container environment and the interface to that environment

The Job Scheduler provides the job managment function and interface

The job scheduler provides you a view into the batch container environment and gives you control to submit and manage jobs

*** The two may be in the same server, or separate servers. Or clustered. Your choice. ☺**

The Job Scheduler Interfaces

The previous chart tended to focus on the web interface, which is certainly the easiest to use. But others are present and offer great value:

Browser

A web interface allows very simple access. Use this when first starting out and for ad hoc management

Command Line

Automation through shell script programming to the CLI

Web Service

Useful to expose the Java batch environment without requiring access to the Job Management Console

RMI

Useful to expose the Java batch environment to EJB clients

JMX

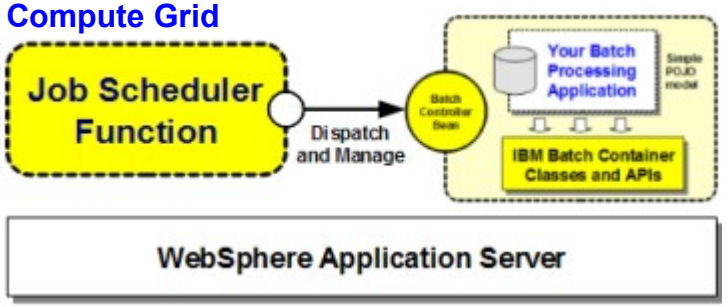
Useful to expose the Java batch environment to Java JMX client

Tivoli. software

Or others

MDB (Compute Grid only)

The method used to integrate with enterprise schedulers



A wide variety of access methods

Blend to meet your business needs

The Job Management Console

Provides a browser-based view into the batch environment deployed on WAS. From there you can see status and take actions:

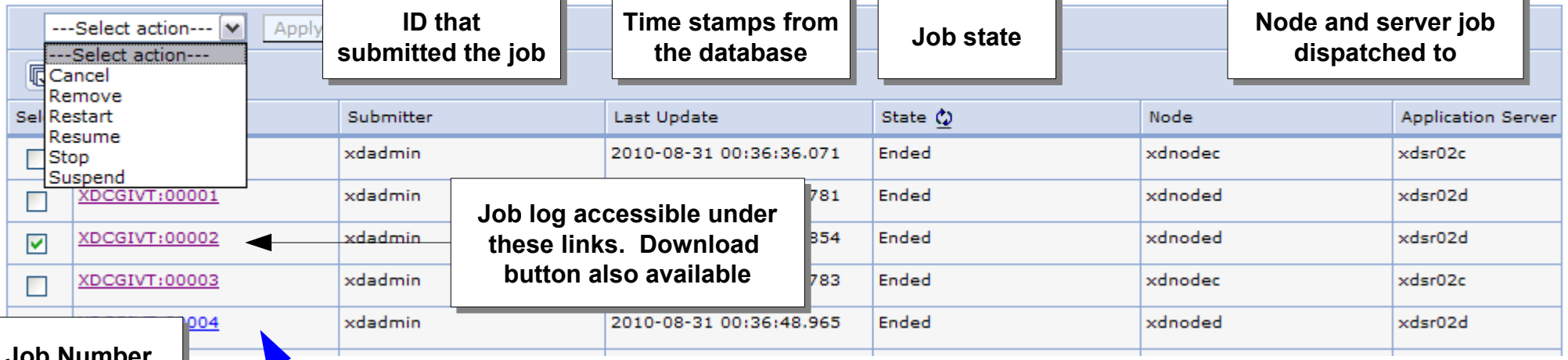
Actions against select jobs

ID that submitted the job

Time stamps from the database

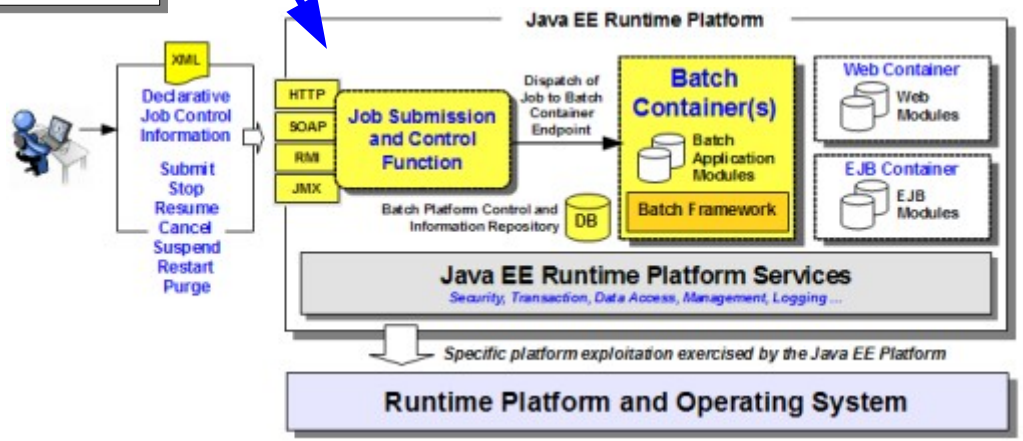
Job state

Node and server job dispatched to



Job log accessible under these links. Download button also available

Job Number



The separation of "job" from "application," and the execution under a managed container environment is what enables this.

This offers considerable operational flexibility.

Command Line, Web Services, IIOP and JMX interfaces as well

The Job Control Definition File -- "xJCL"

Syntax different than traditional JCL, but the concepts are very similar:

```
<?xml version="1.0" encoding="UTF-8" ?>
<job name="name" ... >
  <jndi-name>batch_controller_bean_jndi</jndi-name>
    <substitution-props>
      <prop name="property_name" value="value" />
    </substitution-props>
    <job-step name="name">
    <classname>package.class</classname>
      <checkpoint-algorithm-ref name="chkpt"/>
      <results-ref name="jobsum"/>
        <batch-data-streams>
          <bds>
            <logical-name>input_stream</logical-name>
              <props>
                <prop name="name" value="value"/>
              </props>
          </bds>
      </batch-data-streams>
    </job-step>
</job>
```

Roughly analogous to the JOB card

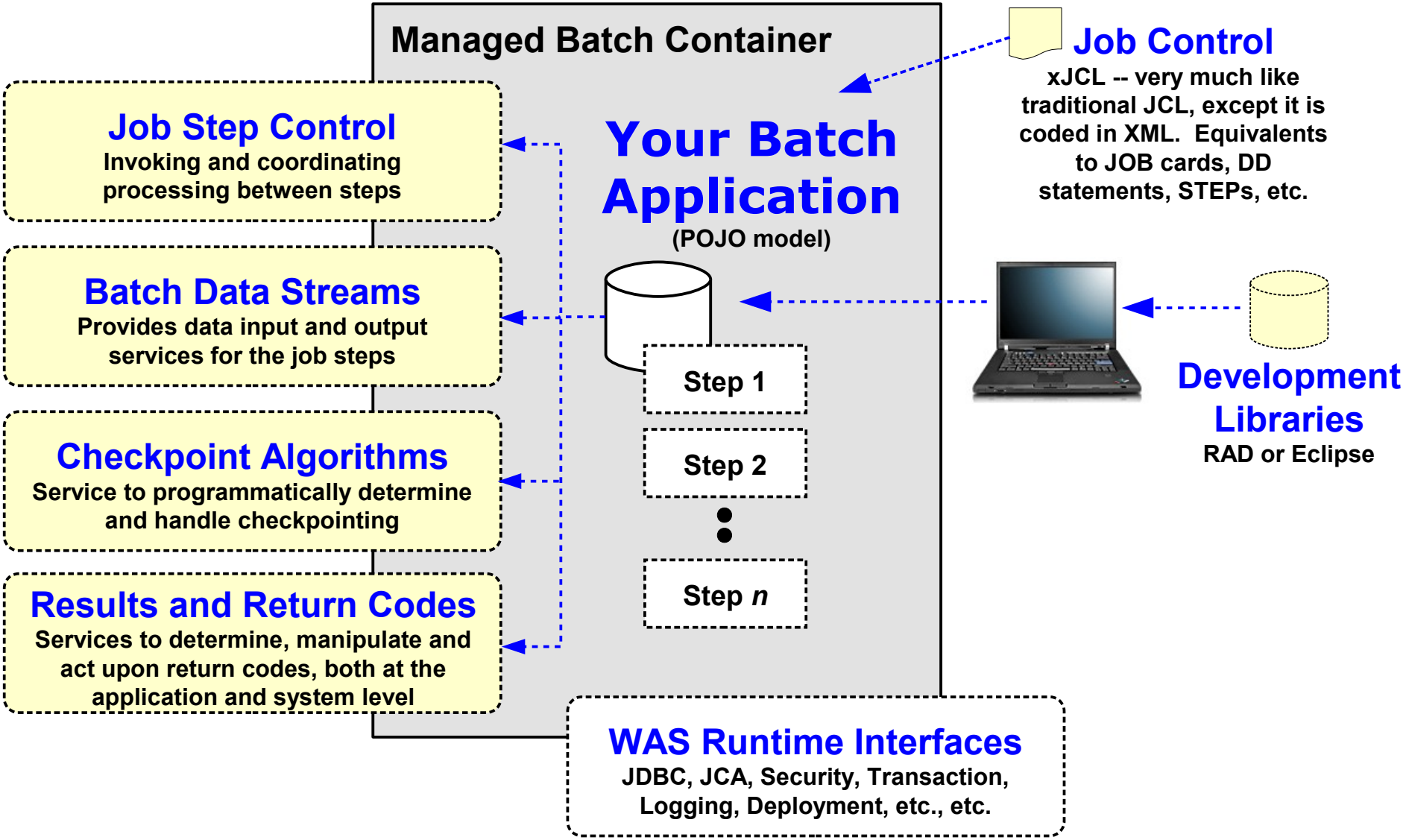
A job step

Like the EXEC PGM= statement in JCL

Similar to DD statements

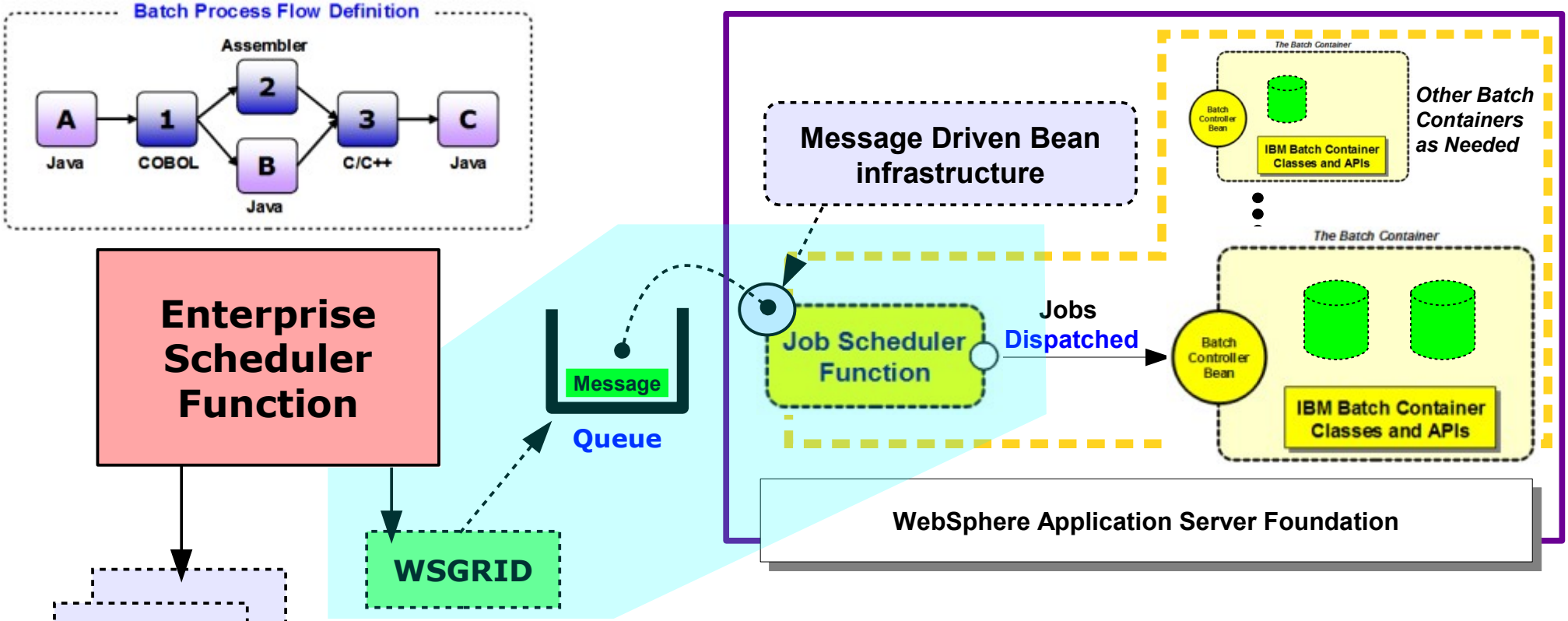
Batch Programming Framework

Provides key functional structures for use by your batch application:



Integration with Enterprise Schedulers

Compute Grid adds another interface to the Job Scheduler function ... an MDB. It supplies a utility that allows schedulers to integrate with Compute Grid:



WSGRID is a program supplied with Compute Grid
- Java client or native MQ client (z/OS only)

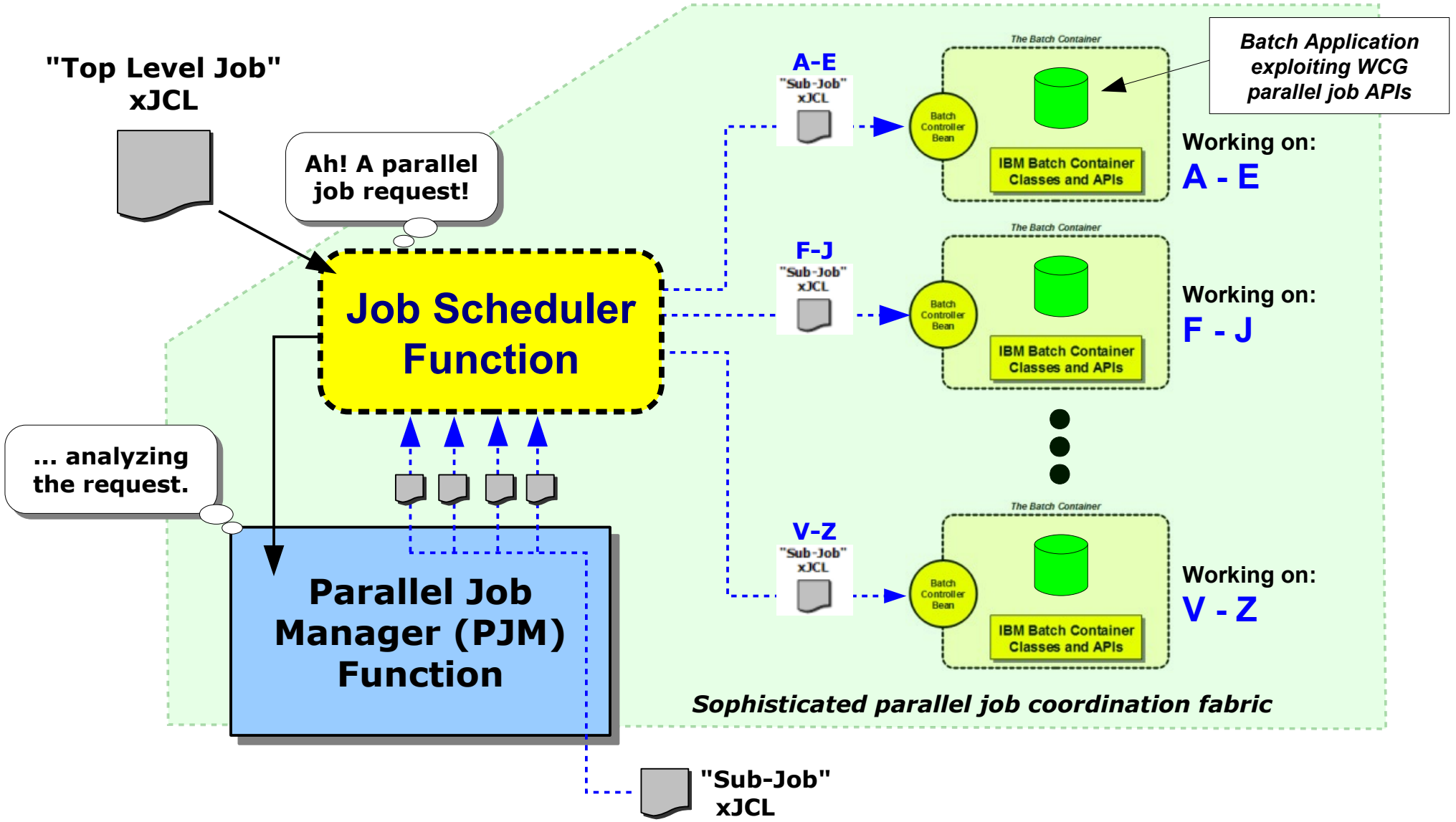
It places a job submission message on the named queue
- Contents of message based on inline properties or properties in named file
- Either MQ or WebSphere messaging implemented with SIBus

The MDB picks it up and submits the job

WSGRID *stays active*, feeding back to Enterprise Scheduler information about the submitted job. Ends only when Compute Grid job ends.

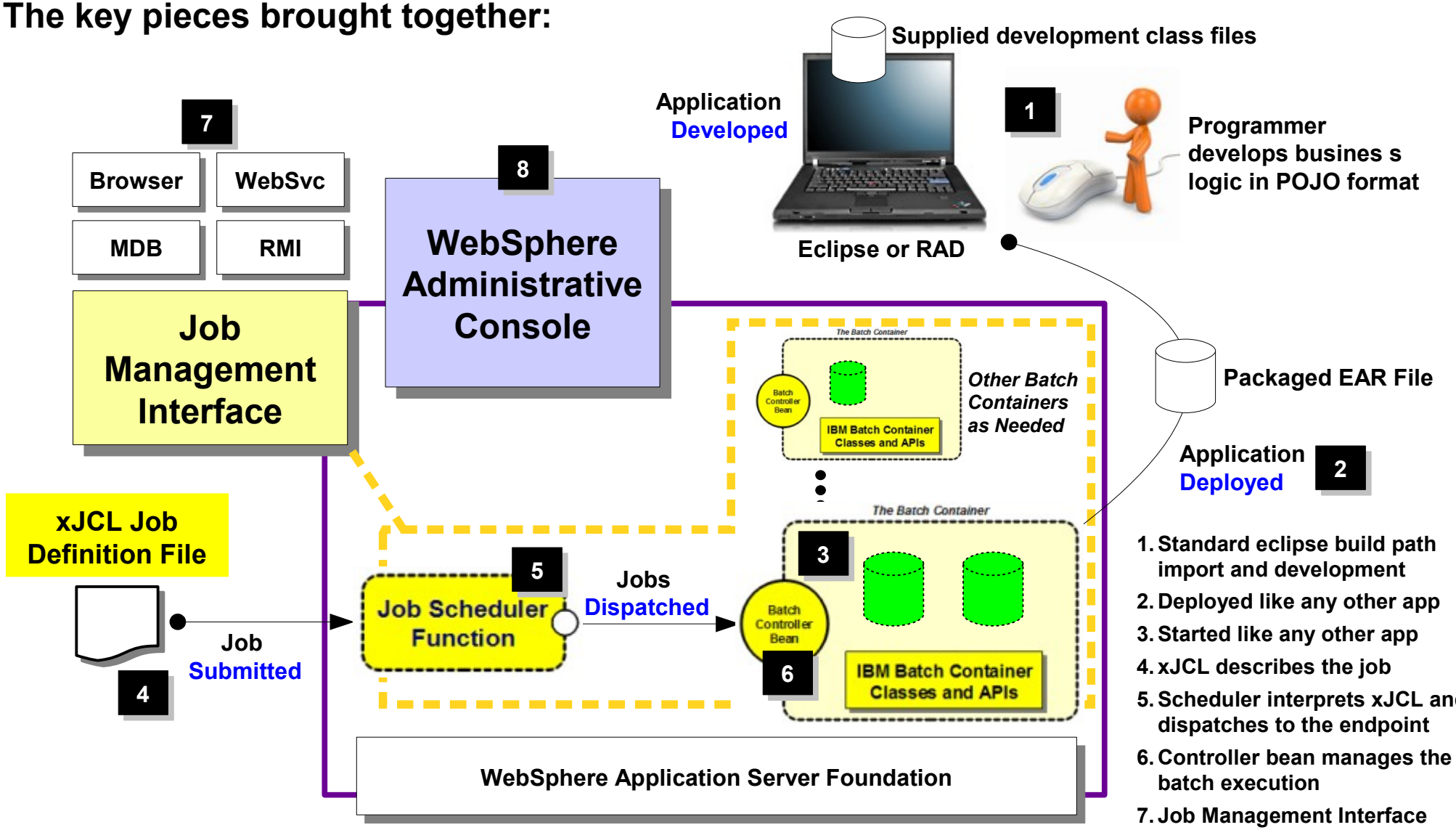
Parallel Job Manager (PJM)

Batch processing often lends itself to parallelization of work. WebSphere Compute Grid facilitates this with function to farm out and collect back parallelized work:



Tie it All Together

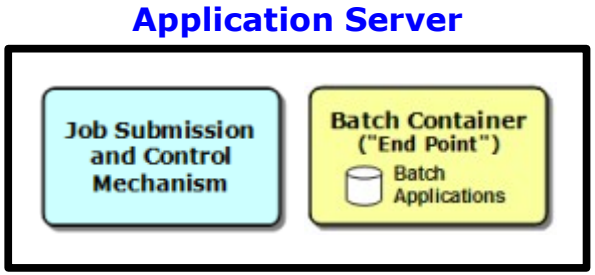
The key pieces brought together:



1. Standard eclipse build path import and development
2. Deployed like any other app
3. Started like any other app
4. xJCL describes the job
5. Scheduler interprets xJCL and dispatches to the endpoint
6. Controller bean manages the batch execution
7. Job Management Interface gives you visibility / control
8. WAS admin console used just as it is for WAS in general

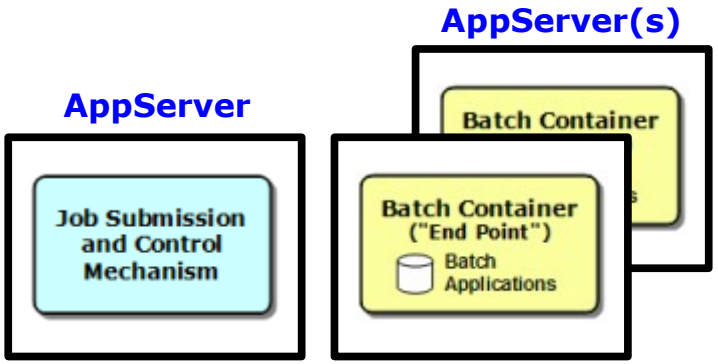
Configuration Topologies -- *All Platforms*

The key message here is really one of flexibility based on your needs:



Both in same application server

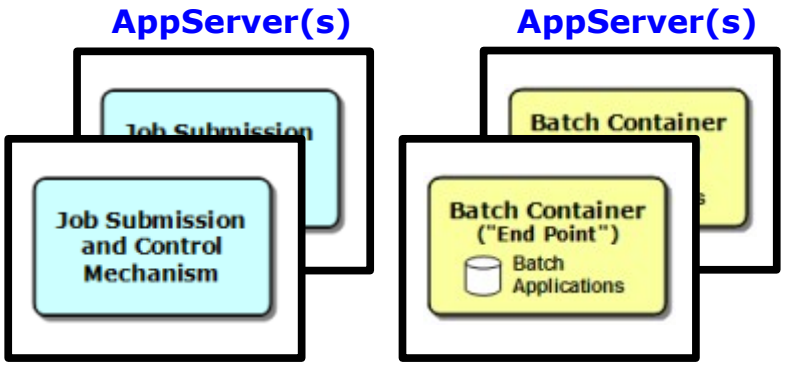
Both functions may reside in the same application server. This is good for development and unit test scenarios



Separate application servers

You may separate the Job Submission from the Batch Containers.

You may have multiple batch container servers, and they may be clustered if desired.



Highly available

You may cluster up the Job Submission mechanism and load balance input between them

You may have multiple batch container servers, and they may be clustered if desired.

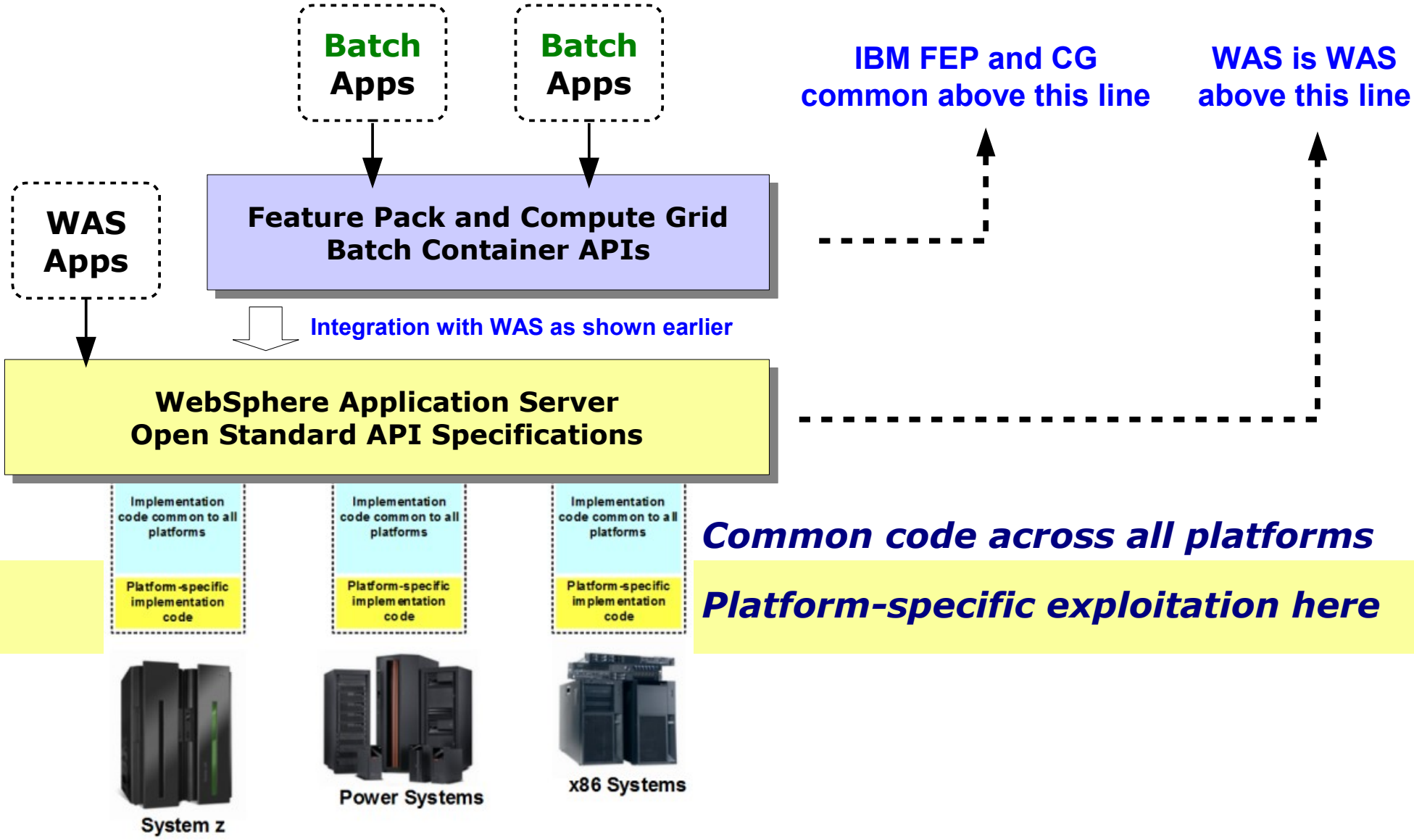
Note: Only one Job Scheduler (submission mechanism) per *cell*

That's because it's integrated into the Administrative Console for the cell

IBM Java Batch and z/OS

Key Point #1 - Common Program Interfaces

IBM has a strategy of providing common and consistent programming interfaces across platforms, with *platform exploitation* taking place below:



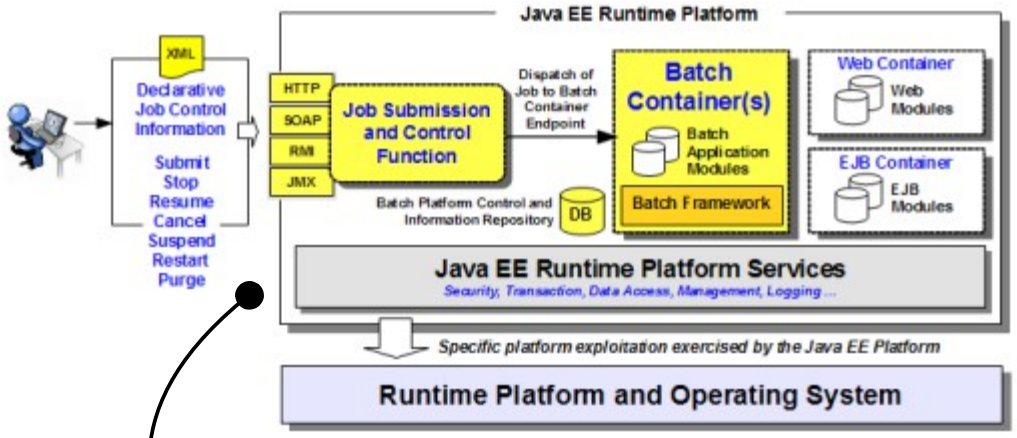
Summary of z/OS Platform Exploitation

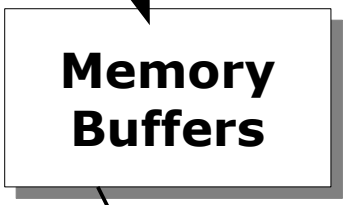
We'll summarize this into four categories:

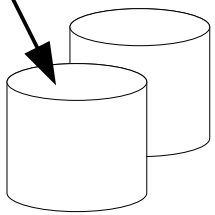
"Why z/OS" Topic	**Covered in considerable detail in the WP101532 Techdoc at** `ibm.com/support/techdocs`
Exploitation of the WAS Services	**WebSphere Application Server services:** • **Transaction management ⇨ z/OS RRS** • **Security management ⇨ z/OS SAF** • **Data access (JDBC, JCA, JMS, WOLA)**
Exploitation of SMF	**Compute Grid for z/OS has its own SMF 120.20 record, details on next chart**
Exploitation of WLM	**The ability to use WLM classification to allocate system resources based on region or in Compute Grid, *by job …* details on upcoming chart**

IBM Compute Grid and SMF 120.20

SMF is a powerful (and fast) activity recording subsystem on z/OS. Compute Grid z/OS exploits this with its own SMF record:







Dump Data Sets

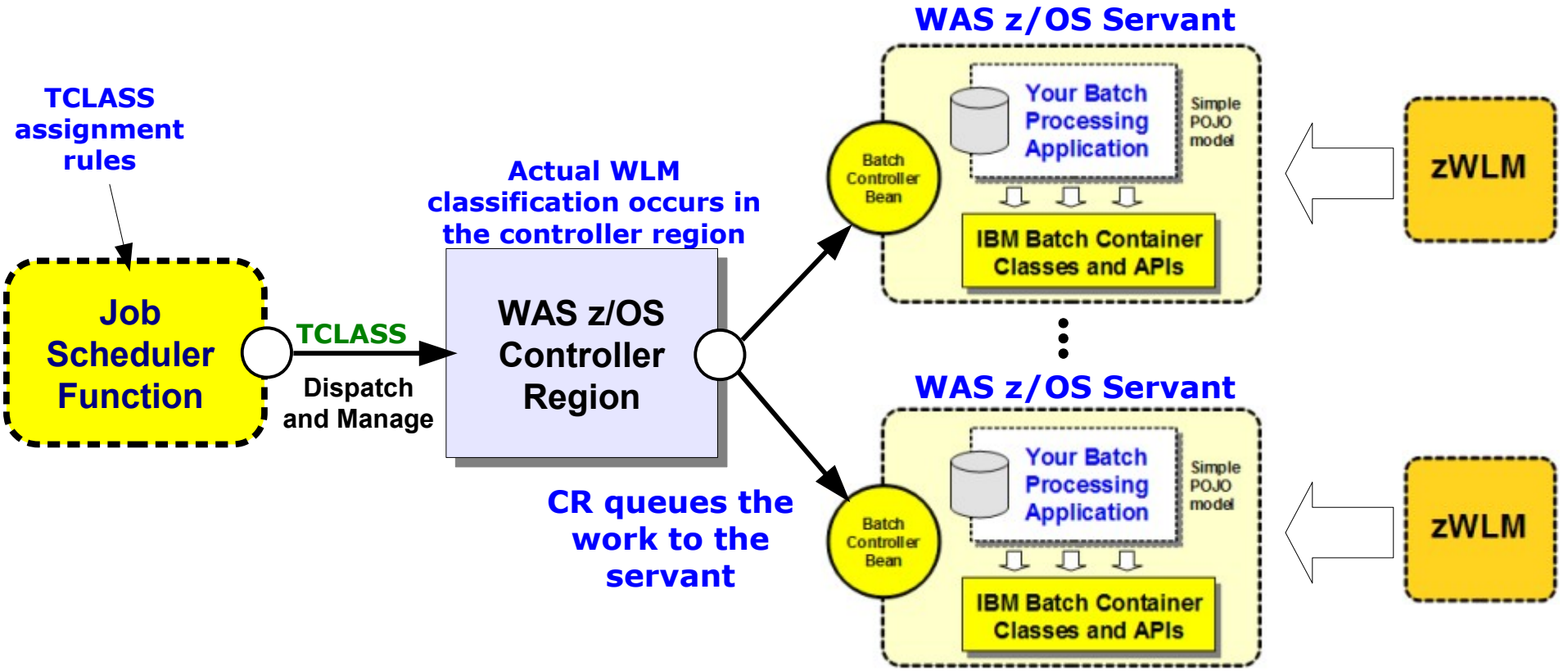
Information in SMF 120.20 record:

- **Job identifier**
- **Job submitter**
- **Final Job state**
- **Server**
- **Node**
- **Accounting information**
- **Job start time**
- **Last update time**
- **General CPU**
- **zAAP CPU**

Then powerful industry analysis tools can be used to generate reports and determine usage for reasons such as capacity planning and chargeback

WLM and Batch Container Job Classification

The function provided depends on Feature Pack or full WebSphere Compute Grid



Feature Pack — Classify batch work separately from other work and allow WLM to manage all according to goals

Compute Grid — Classify batch jobs separately, queue them to separate servants, and allow WLM to manage each according to goals

Summary

Summary

- Java is a viable programming language for batch processing
- There are different Java batch solutions with varying degrees of sophistication and functionality
- Beware the temptation to start coding custom middleware functionality to add features to the less robust Java batch solutions
- IBM's Java batch solutions are based on WebSphere Application Server
- Two offerings: Feature Pack for Modern Batch and Compute Grid
- Both provide a managed batch container environment
- Both provide separation of "application" from "job"
- Both provide a rich set of interfaces for submission and control
- There is a migration path from the Feature Pack to Compute Grid
- Compute Grid provides full enterprise features such as exploitation of SMF, exploitation of WLM classification, and integration with enterprise scheduler systems.

Document Change History

October 25, 2010	Original document published with WP101783 Techdoc number
November 19, 2010	Updated with PJM information and navigation assists
December 14, 2010	Small corrections to spelling and grammar

Details Page

Batch container environment

The batch container is the mechanism inside the application server that provides the interfaces and control entry point for running batch jobs in a Java EE environment. It's implemented as an asynchronous bean stucture with your batch application running under the control of that asynch bean.

Job scheduler and dispatcher function

A mechanism that takes as input the job control file, interprets the contents of that file and determines where the batch application resides. It then dispatches the job to the chosen batch endpoint. It then provides a management control point for viewing the status of the jobs, canceling or restarting the jobs, and seeing the held job output.

Declarative job control file (xJCL)

An XML file that provides the description of the batch job and the components that comprise it. In concept it is very much like the traditional JCL. xJCL provides a way to organize a batch job into steps, provide conditional step-wise execution, designate the checkpoint algorithms and intervals, and designate substitution properties that may be passed to steps during execution.

Development class libraries

A set of Java class librariesused during Java batch application development so the batch application may interface easily with the managed batch container environment offered by IBM.

Batch Data Stream (BDS)

A functional service of the batch container that abstracts data input and output in such a way that a considerable amount of that work is offloaded from the developer. This aids batch application design and development.

Conditional multi-step job support

The ability to designate step processing based on prior step completion codes. The managed container provides a means of carrying step results forward and rolling forward and maintain an overall view of results.

Checkpoint processing leveraging WAS transaction manager

A built-in facility of the managed batch container, this allows you to specify the type of checkpoing (record or time) and the interval. You may do this by step within the job. And the managed batch container interacts with the underlying WAS transaction manager (and RRS on z/OS) to perform transaction rollbacks as needed.