JSR 352 - The Future of Java Batch and WebSphere Compute Grid

David Follis
IBM
<table>
<thead>
<tr>
<th>Session</th>
<th>Title</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>17363</td>
<td>Debug 101-Using ISA Tools for Apps in WebSphere Application Server z/OS</td>
<td>Monday 11:15</td>
<td>Europe 11</td>
</tr>
<tr>
<td>17367</td>
<td>WebSphere Liberty on Windows and z/OS (Among Other Things) Hands-On Lab</td>
<td>Tuesday 10:00</td>
<td>Asia 5</td>
</tr>
<tr>
<td>17361</td>
<td>ABCs of WAS</td>
<td>Tuesday 1:45</td>
<td>Oceanic 7</td>
</tr>
<tr>
<td>17368</td>
<td>z/OS Connect: Opening up z/OS Assets to the Cloud and Mobile Worlds</td>
<td>Tuesday 3:15</td>
<td>Oceanic 7</td>
</tr>
<tr>
<td>17362</td>
<td>Configuring Timeouts for WebSphere Application Server on z/OS</td>
<td>Wednesday 8:30</td>
<td>Oceanic 7</td>
</tr>
<tr>
<td>17366</td>
<td>WebSphere Liberty and WebSphere Application Server Classic - What's New?</td>
<td>Wednesday 11:15</td>
<td>Oceanic 7</td>
</tr>
<tr>
<td>17364</td>
<td>IBM Installation Manager for z/OS System Programmers: Web-based Installs, Fix Packs, and How iFixes Really Work</td>
<td>Thursday 4:30</td>
<td>Oceanic 7</td>
</tr>
<tr>
<td>17365</td>
<td>JSR 352 - The Future of Java Batch and WebSphere Compute Grid</td>
<td>Friday 10:00</td>
<td>Oceanic 6</td>
</tr>
</tbody>
</table>
Topics to be Discussed

• Brief Overview of Batch Processing
  Including background on Java Batch evolution

• Overview of JSR 352
  A review of the key elements of the standard

• IBM Implementation and Extensions
  A review of how JSR 352 is implemented by IBM, including extensions to the standard that provide additional operational features and benefits
Batch Processing ...

... and what led up to Java Batch
Batch Processing Has Been Around a Very Long Time

There has long been a need to process large amounts of data to arrive at results from the data.

There continues to be the same need today.

It is unlikely the need to do processing in batch will go away any time soon.

The need persists, the approach has evolved ...

A picture from the 1960s, and batch processing pre-dated this by several decades, or even centuries, depending on what is considered a “computer”.

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
Change is driven by need. So what is driving the trend towards Java for batch processing?
Things Creating Push to Java for Batch

**Desire to Modernize Batch Processes**

Motivation behind this takes many forms – new business needs; some update to an existing batch program is needed and it’s seen as a good opportunity to re-write in Java; separate business logic into rules engine for more agile processing.

**Availability of Java Skills**

Particularly relative to other skills such as COBOL.

**z/OS: Ability to Offload to Specialty Engines**

Workload that runs on z/OS specialty engines (zAAP, zIIP) is not counted towards CPU-based software charges.

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
Can Java Run as Fast as Compiled Code?

Comparably … and *sometimes* faster*:

• Batch processing is by its nature iterative, which means Java classes prone to being Just-in-Time (JIT) compiled at runtime
• Java JIT compilers are getting very good at optimizing JIT’d code
• z/OS: System z processor chips have instructions specifically designed to aid JIT-compiled code
• COBOL that has not been compiled in a long time is operating with less-optimal compiled code that does not take specific advantage of chip instructions

*Results vary, depending on many factors. This is not a promise of performance results.*
The Evolution of Java Batch …

**Roll Your Own**
- Typically built around the JVM launcher concept
- Processing logic all custom-code, often specific to the batch program, with only some re-use

**Vendor Batch Frameworks**
- Examples: IBM Compute Grid, Spring Batch
- Framework removed a lot of coding effort from developer and allowed focus to be on business logic

**Open Standard**
- JSR 352
- Open standards allow for code portability and the development of larger libraries of re-usable code

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
Overview of JSR 352
The Process of Creating an Open Standard

**Formation of Working Group**

Individuals working on the challenges independent of one another

**Initial Release of Standard Specification**

The group works to create a vision and a document of the proposed specification.

After review and acceptance, it becomes a published specification.

IBM led this group, with involvement from people from several other companies.


The specification details the requirements and interfaces.

The JSR 352 specification was released in May 2013, and has been accepted as a component of the Java EE 7 specification as well.

**Release of Vendor Implementations**

Vendors release products and provide extensions for additional value

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
Very Abstract Representation of a “Batch Job”

A way to initiate and control job

A way to describe the details of the job to whatever does the job submission and control

A way to keep track of the state of the execution

We offer this as a way to set the stage for the discussion of the JSR 352 specification

Job

Job Step

• Read data
• Process data
• Write data

Job Step

• Read data
• Process data
• Write data

Job Step

• Read data
• Process data
• Write data

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
The JSR 352 Diagram to Describe the Architecture

A file that declares the specifics of the job and the steps contained in the job

A job is a logical representation of the batch processing

A job may consist of one to many steps

You’ll see how this is implemented in an upcoming section of this presentation

www.SHARE.org/Orlando-Eval
How Much of that Picture Do / Have to Code?

We explain why this is “optional” in the section on IBM extensions.

It turns out … relatively little

Much of the processing is handled by the vendor implementation of the JSR 352 standard. Your code sits behind standard interfaces and is called by the JSR 352 runtime.

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
Job Step Types – Chunk and Batchlet

**Chunk Step**

- What we typically think of as a “batch job” – an iterative loop through data, with periodic commits of data written out during processing.
- This involves the ItemReader, ItemProcessor and ItemWriter interfaces shown earlier.

**Batchlet Step**

- A job step with much less structure … it is called, it runs and does whatever is in the code, and ends.
- This job step type is useful for operations that are not iterative in nature, but may take some time … a large file FTP, for example.
- This is also useful for encapsulating existing Java main() programs into the JSR 352 model.

**A multi-step job may consist of either … or both**
**High-Level Example … to Illustrate the Key Concepts**

Not real-world, but useful to illustrate essential JSR 352 concepts. What does packaging look like?

---

**SimpleBonus**

- **generate**
  - First step is a Batchlet that writes account data to a file. This data will serve as input to the second step.

- **addBonus**
  - Second step is a Chunk step that reads the records from the file, adds a fixed integer value to each account, and inserts a row in a table.

---

Job Specification Language (JSL) file

CSV Format

ACCOUNT table

---

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
A Peek Inside the Sample Application WAR file

Application Developer

BonusPayout-1.0.war

WEB-INF

\classes\com\ibm\websphere\samples\batch

\artifacts

GenerateDataBatchlet.class
GeneratedCSVReader.class
BonusCreditProcessor.class
AccountJDBCWriter.class
(other class files)

\beans
(data bean class files)

\util
(utility class files)

\classes\META-INF\batch-jobs
SimpleBonusPayoutJob.xml

This deploys into the Liberty Profile server’s /dropins directory, or pointed to with <application> tag like any other application.

The “How to write JSR 352 applications” topic is important, but outside the scope of this overview discussion.

Step 1 Batchlet

Step 2 Chunk ItemReader ItemProcessor ItemWriter

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
Properties are a way to get values into your batch job. They can be specified in the JSL as shown, and overridden at submission time using IBM’s REST interface (shown later).

The first step is defined as a Batchlet. The Java class file that implements the Batchlet is indicated. The property to tell the Batchlet how many records to create is specified.

The job specification is taking shape. What about the second step? That’s shown next …
A property on the processor provides the integer bonus to add to each account. Properties on the writer indicate how to reach the database and what table to use.

Summary: the JSR 352 runtime provides the infrastructure to run batch jobs; this JSL tells it what Java classes to use and other details related to the operation of the job.
Parallel Job Processing

- Splits and Flows provide a mechanism for executing job steps concurrently at the orchestration layer
- A flow is a sequence of one or more steps which execute sequentially, but as a single unit.
- A Split is a collection of flows that may execute concurrently
  - A split may only contain “flows”; a step is not implicitly a flow
- This is done entirely in the JSL descriptor
  - Imposed on the batch application with no code changes!
Step-level parallelism

- Step-level parallelism can be achieved programmatically using step partitioning
- A partitioned step runs as multiple instances with distinct property sets
- PartitionMapper defines the number of partitions, and property values for each partition
  - Can be a fixed set of partitions in JSL
  - Can be dynamic using a PartitionMapper implementation
Step-level parallelism

- No new Java artifacts
  - Not necessarily the general case
  - Might have to
    - Coalesce Exit Status (PartitionAnalyzer)
    - Process Intermediate results on parent thread (PartitionCollector->PartitionAnalyzer)
    - Perform other tasks on end of partition (PartitionReducer)
- Also might want to programmatically partition (PartitionMapper) rather than via JSL
Parallel Job Processing

Design Source

Partition plan property substitutions
Partition mapper reference and properties

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
IBM Implementation
And Extensions
Liberty 8.5.5.6 and above
- IBM’s fast, lightweight, composable server runtime
- Dynamic configuration and application updates

JVM Stays Active Between Jobs
- Avoids the overhead of JVM initialize and tear down for each job

IBM Extensions to JSR 352
- JSR 352 is largely a *programming* standard
- IBM extensions augment this with valuable *operational* functions
  - Includes:
    - Job logs separated by job execution
    - REST interface to JobOperator
    - Command line client for job submission
    - Integration with enterprise scheduler functions
    - Multi-JVM support: dispatcher and endpoint servers provide a distributed topology for batch job execution
JobRepository Implementation

The JSR 352 standard calls for a JobRepository to hold job state information, but it does not spell out implementation details.

IBM JSR 352 provides three options for this:

1. An in-memory JobRepository
   For development and test environments where job state does not need to persist between server starts

2. File-based Derby JobRepository
   For runtime environments where a degree of persistence is desired, but a full database product is not needed

3. Relational database product JobRepository
   For production and near-production environments where a robust database product is called for

Table creation is automatic. Relatively easy to drop one set of tables and re-configure to use a different data store.

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
REST Interface to JobOperator

The JSR 352 standard calls for a JobOperator interface, but leaves to vendors to implement function to handle external requests for job submission, control and monitoring.

The IBM JSR 352 REST interface provides:

1. A RESTful interface for job submission, control and monitoring
   Job submission requests may come from outside the Liberty Profile runtime

2. Security model for authentication and authorization
   Authorization is role-based: administrator, submitter, monitor

3. JSON payload carries the specifics of the job to be submitted
   With information such as the application name, the JSL file name, and any parameters to pass in

   This permits the remote submission and control of jobs; it provides a way to integrate with external systems such as schedulers.

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
The `batchManager` command line interface client provides:

1. A way to submit, monitor and control jobs remotely using a command line interface
   On the same system, or a different system ... different OS ... doesn’t matter: TCP/IP and REST/JSON

2. Uses the REST interface on the IBM Java Batch server
   Which means the same security model is in effect: SSL, authentication, role-based access

3. External schedulers can use this to submit and monitor job completion
   `batchManager` parameters allow the script to “wait” for Java to complete. Parameters allow for discovery of job log information, and a mechanism to retrieve the job log for archival if desired.
Same LPAR, cross-memory

batchManagerZos

> command

Person

Script

**Same batchManager command line function, but …**

1. **Not a Java client, so do not need to spin up a JVM for each invocation**
   - Saves the CPU associated with initiating the JVM, and when there’s a lot of jobs this can be significant

2. **Cross-memory**
   - Very low latency, and since no network then no SSL and management of certificates

3. **Same access security model**
   - Once the WOLA connection is established, the same “admin,” “submitter” and “monitor” roles apply

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
Multi-JVM Support: Job Dispatchers, End-Points

Separation of duties …

1. Server designated as dispatchers handle job requests, and places them on JMS queue
   The endpoints listen on the JMS queues and pick up the job submission request based on criteria you set to indicate which jobs to pick up (more on that next chart)

2. Endpoint servers run the batch jobs
   Deploy the batch jobs where most appropriate; co-locate some batch jobs and others have their own server

3. JMS queues (either Service Integration Bus or MQ) serve as integration between two
   This provides a mechanism for queuing up jobs prior to execution

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
Multi-JVM Support: Get Jobs Based on Endpoint Criteria

A property in the `server.xml` defines the “message selector” criteria to use to pick up messages. You can designate – by server – what criteria to use.

```xml
messageSelector="com_ibm_ws_jbatch_applicationName = 'BatchJobA'"
```

```xml
messageSelector="com_ibm_ws_jbatch_applicationName = 'BatchJobA'
AND com_ibm_ws_jbatch_myProperty = 'myValue'"
```

Submit jobs and have them run only when intended server starts and picks up the submission request

Have jobs run in intended servers based on selection criteria of your choice

Not limited to system, not limited to platform … may span systems and platforms

Complete your session evaluations online at www.SHARE.org/Orlando-Eval
Job logs separate from the server log, separate from each other

1. Each job’s logs are kept separate by application name, date, instance and execution

2. The IBM JSR 352 REST interface has a method for discovery and retrieval of job logs
   This is accessible through the batchManager command line interface as well. This is how job log retrieval and archival can be achieved if needed.
Liberty server.xml

:  
<featureManager>
   <feature>servlet-3.1</feature>
   <feature>batch-1.0</feature>
   <feature>batchManagement-1.0</feature>
</featureManager>

:  
<batchPersistence databaseStoreRef="BatchDatabaseStore" />
<databaseStore id="BatchDatabaseStore"
   dataSourceRef="batchDB" schema="JBATCH" tablePrefix="" />

Relatively simple updates to server.xml ...

1. The batch-1.0 feature enables the JSR 352 core functionality
2. The batchManagement-1.0 feature enables the REST interface, job logging, and the ability to configure the multi-JVM support.
3. The <batchPersistence> element provides information about where the JobRepository is located

Some details left out of this chart, of course ... but the key point is that configuring the support is based on updates to server.xml
Overall Summary

IBM Extensions

JSR 352 Standard

Liberty

IBM JSR 352 Java Batch

Multiple JobRepository
Job logging
REST interface
Command line client
z/OS: native client
Multi-JVM capability

Early Days of Batch Processing

Over time …

Modernization
Java

JSR 352

Windows, AIX, Linux, Linux for z Systems, z/OS …

Complete your session evaluations online at www.SHARE.org/Orlando-Eval