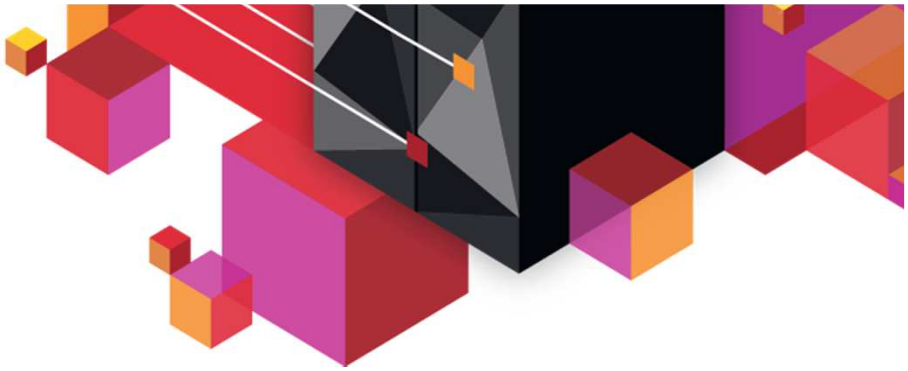


Java Batch solutions on z/OS

Carl Farkas
Europe IOT zWebSphere Consultant
IBM France D/2708
Paris, France
Internet : farkas@fr.ibm.com
Notes : Carl Farkas/France/IBM @ IBMFR

Université du Mainframe 2013

4-5 avril

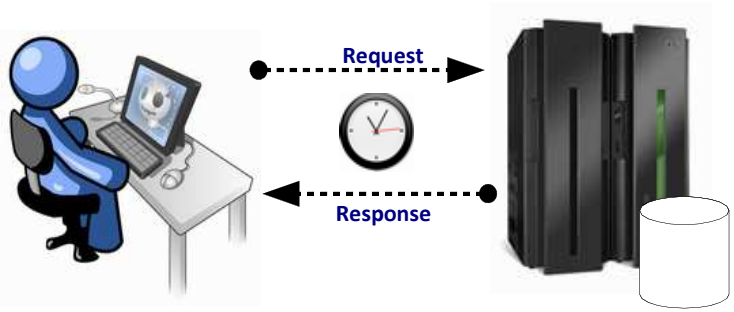
Agenda

- Why do Batch on z/OS?
- Why do Batch with Java on z/OS?
- Solutions for Java Batch on z/OS
  - Java Batch Launchers
  - Java Execution Platforms

What Do We Mean by "Batch"?

This is best defined by contrasting against traditional online processing:

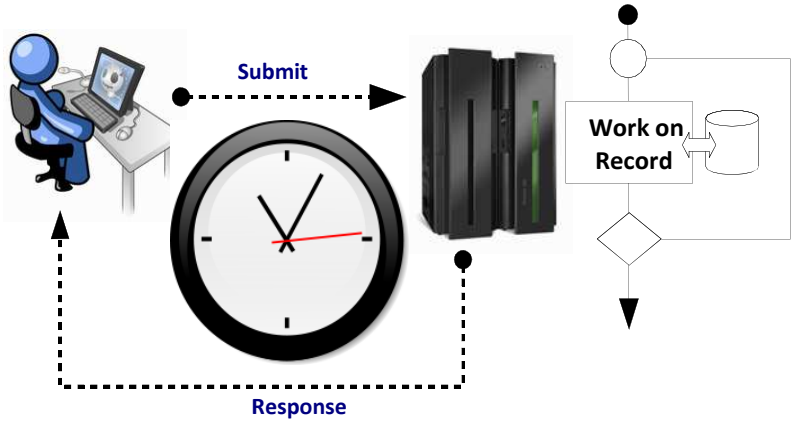
OLTP



- **A request / receive model**
  (not necessarily synchronous, but often is)
- **Duration of processing relatively short**
- **Often transactional in nature**
- **Many runtime environments (eg. WAS) enforce timeouts for this workload**
- **Presentation (IHM) is a key element**

"Batch"

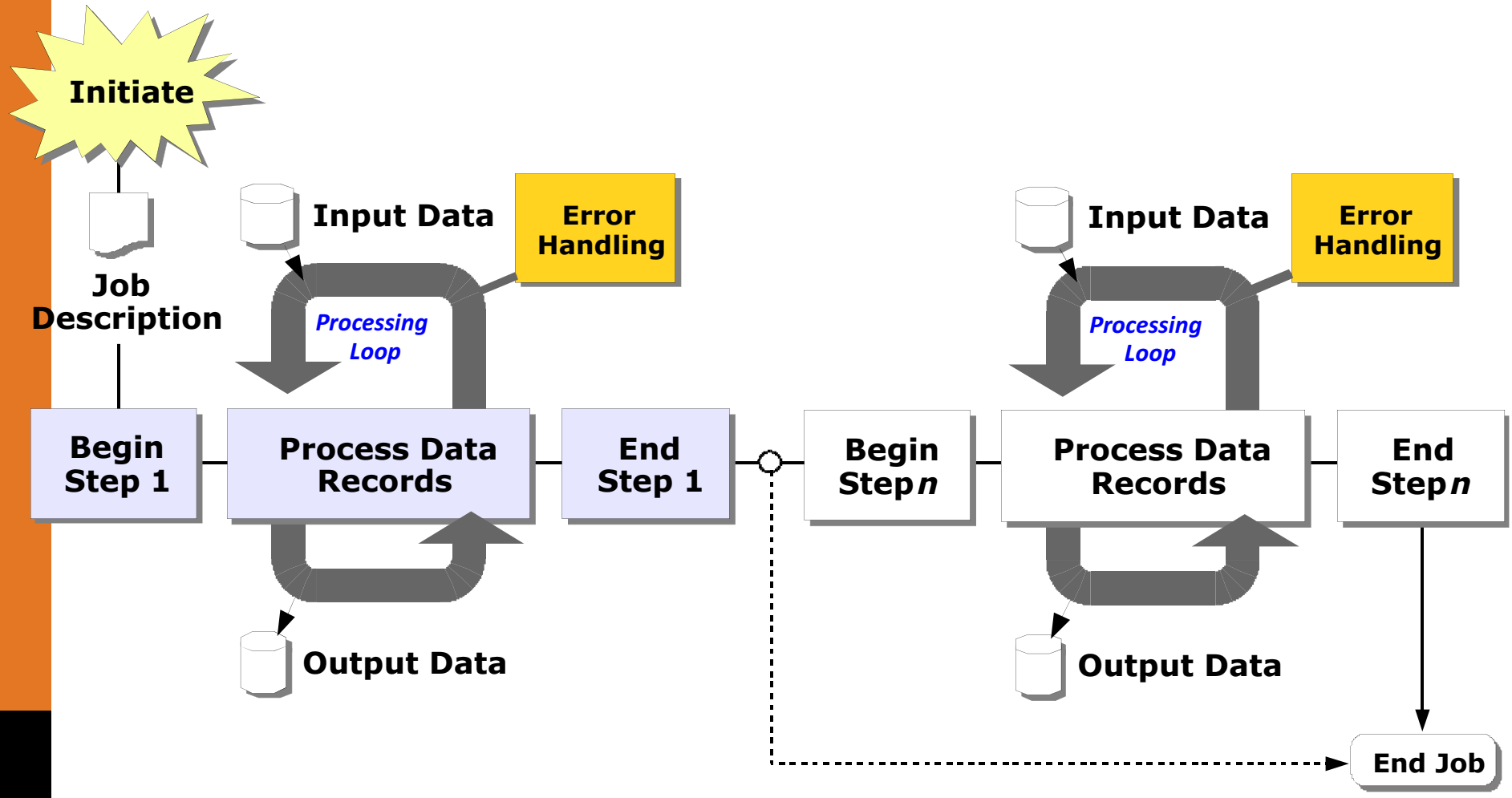
(or "Long Running")



- **A submit / work / result set model**
  (asynchronous to the submitter)
- **Duration of processing a function of work to be done; could be hours/days**
- **Often transactional in nature**
- **Often multi-step processes**
- **Typically data-intensive**

Anatomy of a Batch Job

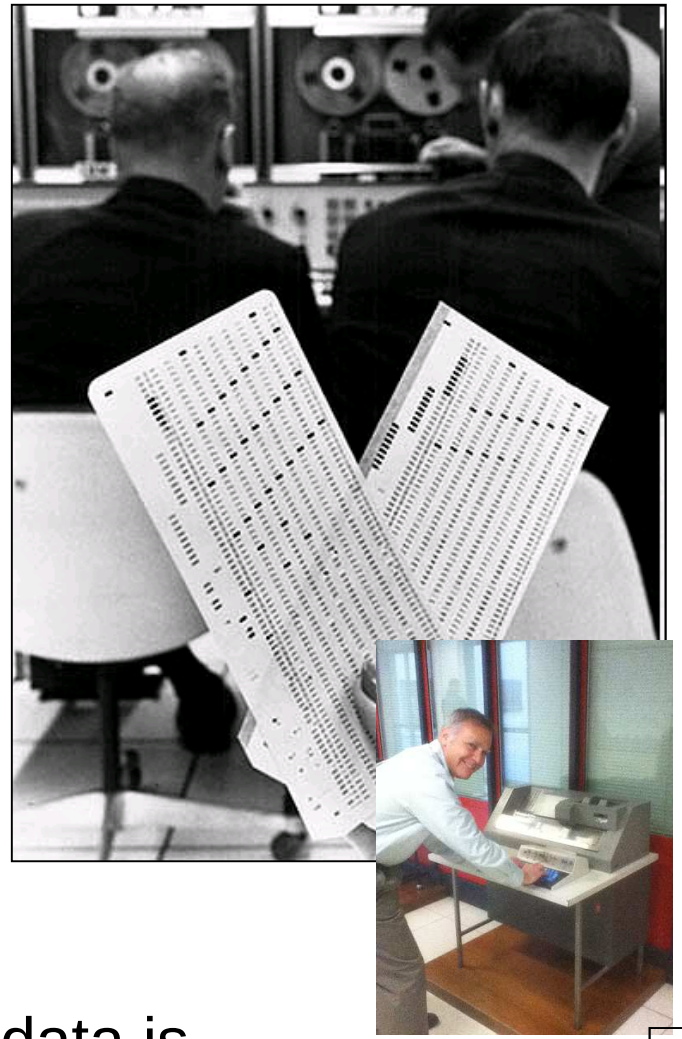
This is a very schematic representation of what takes place….



z/OS 45 years of Batch – From punch cards to Java Batch

45 years of evolution in Batch processing on the IBM Mainframe have provided the foundation for heavy-duty, reliable and efficient Batch for most large companies in the world:

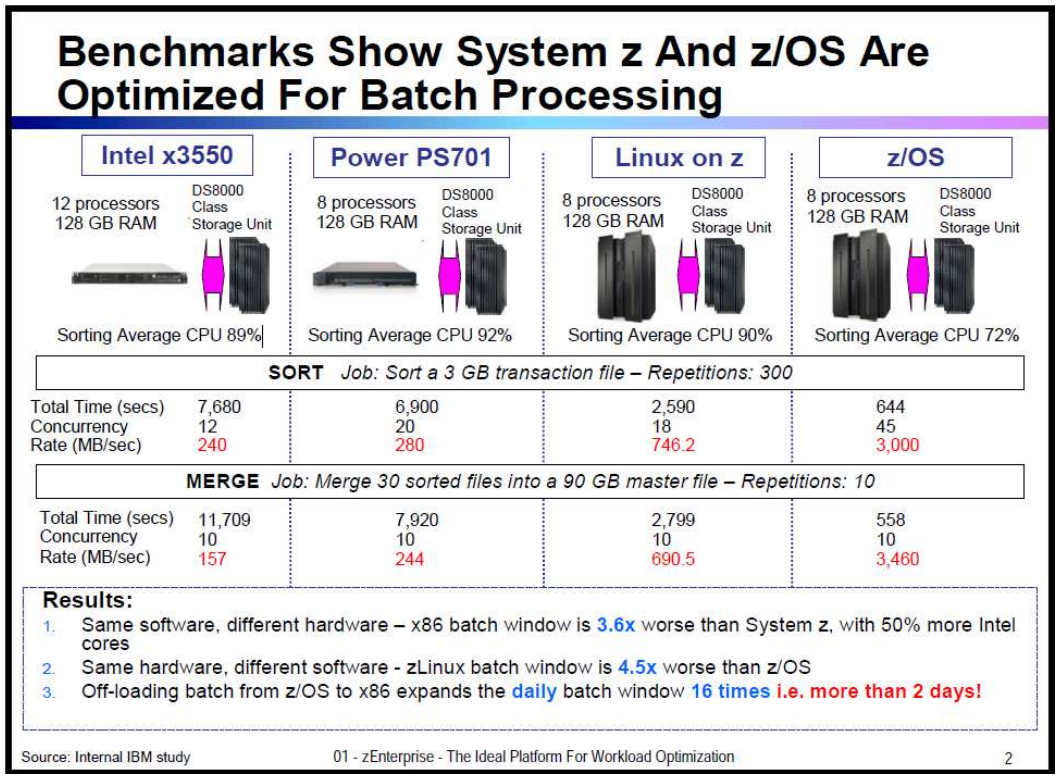
- WLM, WLM Batch initiators
- Batch & Print Subsystem JES2, JES3, PSF
- Job Control Language (JCL)
- Batch Management Interfaces (for example: SDSF)
- Step and Job dependencies by means of Condition Codes and Job Networks
- Online and Batch in parallel
- Time-driven Job execution
- Job / Step Restart functions, Start, Submit, Remote submit, Syntax Scanner
- Accounting based on Job/USER, Job statistics and RMF reports
- Pre-loaded address spaces (initiators)
- All Mainframe programming languages can be used in Batch



And if all that doesn't convince you….
it's about the data! z/OS is where the data is.

Where do you want to run your batch?

- System z, and z/OS have unique advantages for running batch
- Asynchronous I/O is built into the System z architecture
- System z I/O processing is delegated to System Assist Processors (SAP), a specialized hardware engines delivered with every machine
- A highly parallelized architecture is key
- Business objectives are set and utilized to ensure equitable processing



Benchmarks Show System z And z/OS Are Optimized For Batch Processing

	Intel x3550	Power PS701	Linux on z	z/OS
	12 processors 128 GB RAM, DS8000 Class Storage Unit	8 processors 128 GB RAM, DS8000 Class Storage Unit	8 processors 128 GB RAM, DS8000 Class Storage Unit	8 processors 128 GB RAM, DS8000 Class Storage Unit
	Sorting Average CPU 89%	Sorting Average CPU 92%	Sorting Average CPU 90%	Sorting Average CPU 72%
SORT *Job: Sort a 3 GB transaction file – Repetitions: 300*				
Total Time (secs)	7,680	6,900	2,590	644
Concurrency	12	20	18	45
Rate (MB/sec)	240	280	746.2	3,000
MERGE *Job: Merge 30 sorted files into a 90 GB master file – Repetitions: 10*				
Total Time (secs)	11,709	7,920	2,799	558
Concurrency	10	10	10	10
Rate (MB/sec)	157	244	690.5	3,460

Results:

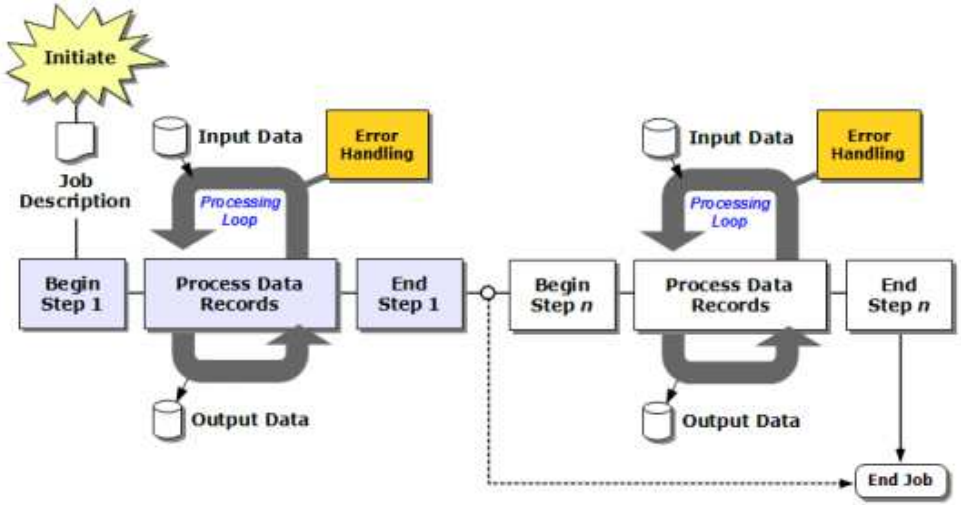
1. Same software, different hardware – x86 batch window is **3.6x** worse than System z, with 50% more Intel cores
2. Same hardware, different software - zLinux batch window is **4.5x** worse than z/OS
3. Off-loading batch from z/OS to x86 expands the **daily** batch window **16 times i.e. more than 2 days!**

Source: Internal IBM study 01 - zEnterprise - The Ideal Platform For Workload Optimization 2

Whitepaper available at http://thebrainhouse.ch/gse/silvio/76.GSE/Silvio%27s%20Corner%20Doc%20Jukebox/

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 enterprise schedulers?**
What will be mechanism for overall job description?	**What platform is the data found on?**	**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?**

Agenda

- Why do Batch on z/OS?
- Why do Batch with Java on z/OS?
- Solutions for Java Batch on z/OS
  - Java Batch Launchers
  - Java Execution Platforms

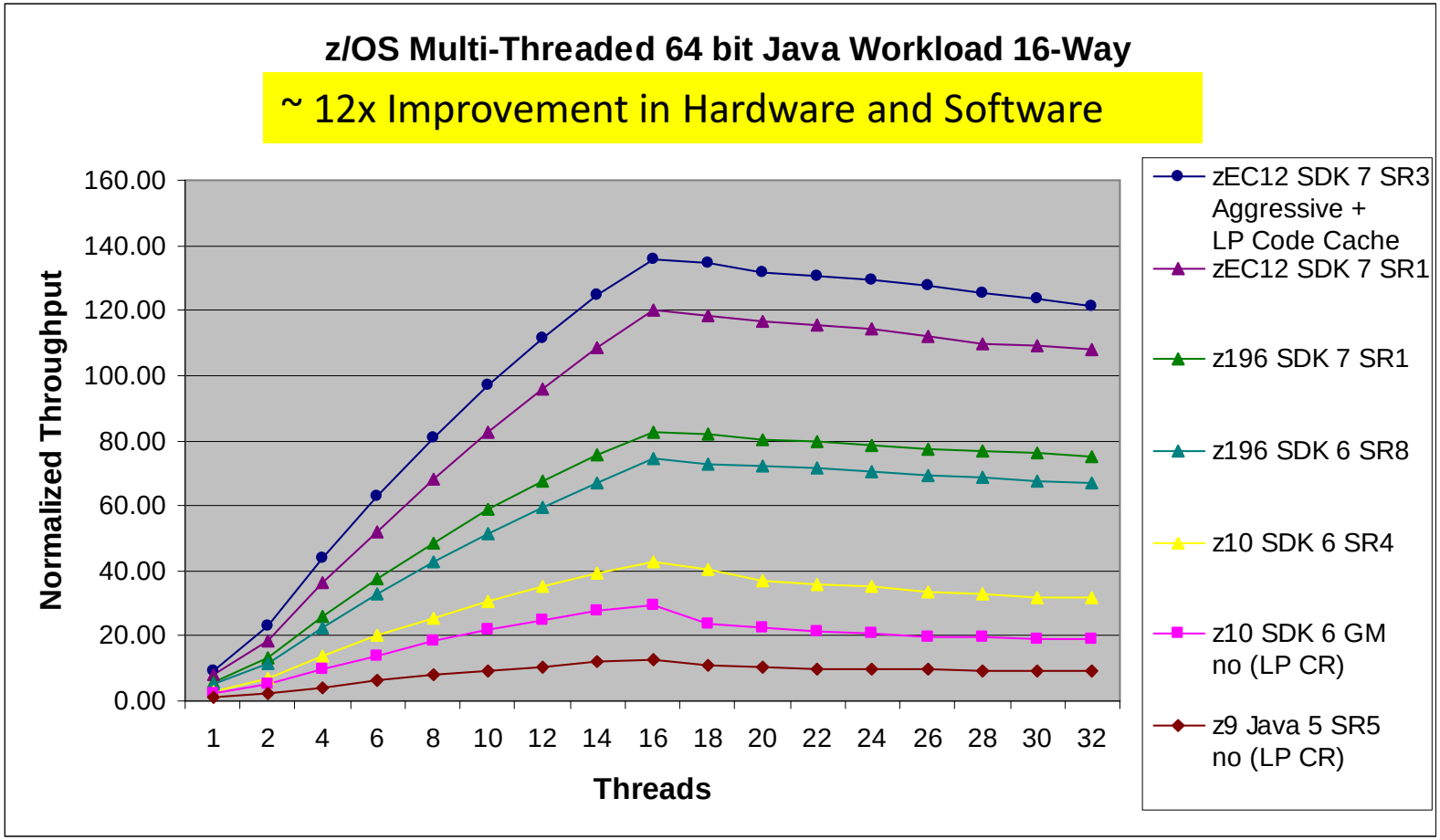
Why do Java Batch on z/OS?

- 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

Important! This does *not* mean wholesale replacement of existing batch function that serves the business needs. Business imperatives drive technical solutions.

z/OS Java SDK 7: 16-Way Performance

Aggregate HW and SDK Improvement z9 Java 5 SR5 to zEC12 Java 7SR3



~12x aggregate hardware and software improvement comparing Java5SR5 on z9 to Java7SR3 on zEC12

LP=Large Pages for Java heap CR= Java compressed references

Java7SR3 using -Xaggressive + Flash Express pageable 1Meg large pages

Tests performed and reported by IBM in October 2012; your results may vary

Java for batch? Common myths …

- "COBOL is a *very* mature language" – people commonly believe it runs faster than Java.
  NOT ALWAYS TRUE: performance tests demonstrate Java yields high performance…. Sometimes surpassing COBOL!

- "Java is used widely for OLTP application" – people commonly believe it is ill-suited for batch.
  NOT ALWAYS TRUE: Java is a general purpose language that easily handles varied tasks

"The performance of compute grid is better or equal to COBOL batch"
Marcel Schmidt, *Global IT Infrastructure*

Swiss Re

Agenda

- Why do Batch on z/OS?
- Why do Batch with Java on z/OS?
- Solutions for Java Batch on z/OS
  - Java Batch Launchers
  - Java Execution Platforms

IBM proposes two Java Batch Models on z/OS

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

Execution Platform

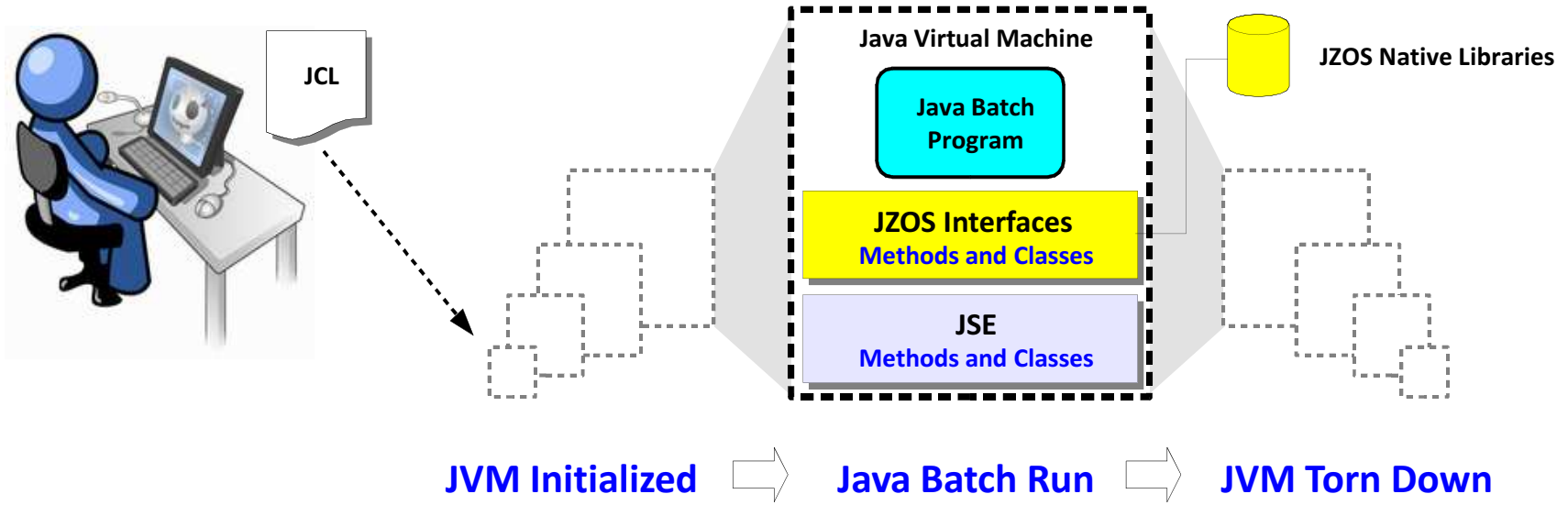
- Provides a set of batch functionality in the form of supporting class libraries and development libraries
- Specific integration with underlying middleware or platform
- Examples: WAS Feature Pack for Modern Batch, WebSphere Compute Grid

Comparison of IBM Java Batch options

	BPXBATCH JZOS	WAS with Modern Batch Feature Pack	WebSphere Compute Grid
Basic Java batch job submission with data access	√	√	√
Conditional multi-step, z/OS SYSOUT, MVS datasets, logical names	√	√	√
Job classes and workload classification	√	√	√
Multi-site disaster recovery for batch platform	√	√	√
Long-running Java batch environment	√	√	√
Portable batch container between environments and platforms		√	√
Development tools to develop batch applications		√	√
Container managed check point, restart capabilities		√	√
Job management console, Operational commands		√	√
Application Execution Platform		√	√
Basic Scheduler/Job dispatcher		√	√
System managed job logs		√	√
High availability and clustering of Batch Job Scheduler/Job Dispatcher		√	√
Non-disruptive batch application update/endpoint quiesce			√
Job usage accounting, including SMF integration on z/OS			√
Parallel Job Manager to reduce Batch window			√
Enterprise Scheduler connectors			√
Direct COBOL program call (COBOL container) in WCG v8 z/OS			√

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**

Using “raw” Java on z/OS - BPXBATCH

- ”Simple” Java Batch means that we run plain old Java in a JVM without WebSphere Application Server, CICS, IMS, or any other transaction monitor.
- Java Batch runs as a normal step in a JCL-procedure very similar to COBOL or PL1
- Java Batch can be started with BPXBATCH (or BPXBATSL for a bit better environment)
- BPXBATCH is an MVS utility that you can use to run shell commands or shell scripts
  - BPXBATCH runs in an OMVS initiator (BXPAS)
- Java Batch programs are located in HFS (or zFS)
- Java Batch programs can access HFS/zFS files as well as ”classic” QSAM & VSAM (KSDS) dataset through special classes called RECORDIO classes (JRIO)
- Data proximity to DB2 z/OS through JDBC Type II driver, JMS with MQ bindings, etc.
- Typically takes 1-2 seconds of CPU per startup

```
//HELLO     EXEC PGM=BPXBATCH,REGION=0M,
// PARM='sh /u/lec/lfajee/runHelloWorld.sh'
//STDOUT     DD   PATH='/tmp/runstdout',
//               PATHOPTS=(OCREAT,OTRUNC,OWRONLY),PATHMODE=SIRWXU
//STDERR     DD   PATH='/tmp/runstderr',
//               PATHOPTS=(OCREAT,OTRUNC,OWRONLY),PATHMODE=SIRWXU
```

/u/lec/lfajee/runHelloWorld.sh:

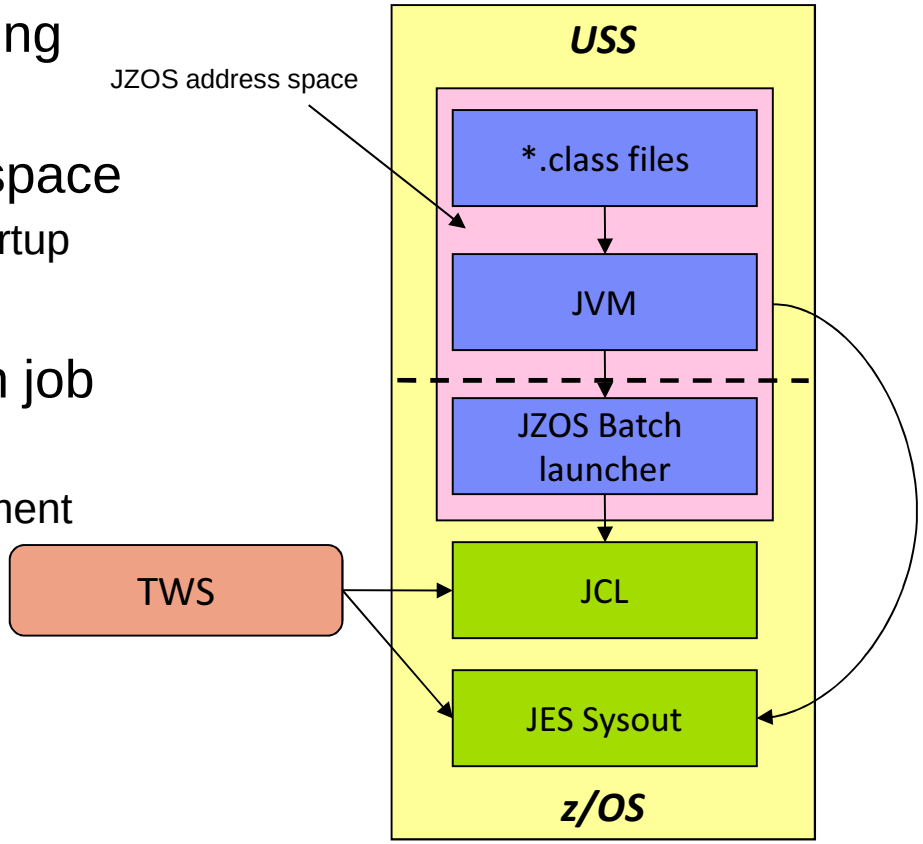
```
cd /u/farkas/mydir
set MYENVSWITCH=”TRACEON”
java -cp ./ HelloWorld
```

Using “raw” Java on z/OS - JZOS

- Using ”classic” Java on z/OS still has (had!) some limitations...
- Difficult to set up sophisticated JCL scripts for controlling the Java programs
- Control must be done primarily within the “shell”
- Access to many z/OS features is limited….
- IBM JZOS Batch Toolkit for z/OS SDKs
  - A batch launcher (EXJZOSVM) for an enhanced runtime environment
  - A toolkit to leverage many z/OS-specific services from Java, eg.
    - com.ibm.jzos.ZFile for native z/OS file access (QSAM, PDS, VSAM..) in record or stream mode.
    - MvsConsole.wto for z/OS console interaction (writing, reading), MvsJobSubmitter
    - Etc.
- Originally developped by Dovetail, but bought by IBM and distributed with z/OS
- For details and downloading, see http://www.ibm.com/servers/eserver/zseries/software/java/products/jzos/overview.html
- Delivered with z/OS today, and exploited by many other IBM z/OS products, eg. WAS

JZOS advantages over BPXBATCH

- Easy integration of Java into existing z/OS batch environment
- Java runs in the original address space
  - ✓ Less overhead for address space startup
  - ✓ CPU associated with batch id
- Return codes are passed between job steps
  - ✓ Better integration with batch environment and schedulers
- DD cards exploited
  - ✓ Data can be passed between steps



Example of JCL with JZOS

```
//PROCLIB JCLLIB ORDER=D2200.PROCLIB
//JAVA EXEC PROC=EXJZOSVM,VERSION='14',
// JAVACLS='HelloWorld'
 :
//STDENV DD *
. /etc/profile
export JZOS_HOME=/u/lec/jzos/
export JAVA_HOME=/usr/lpp/java/J1.4/
......
......
CLASSPATH=/u/lec/lfajee/
......
......
/*
//SYSPRINT DD SYSOUT=*          < System stdout
//SYSOUT   DD DSN=CARL.HOLD.SYSERR,DISP=SHR   < Sy
//STDOUT   DD SYSOUT=*          < Java System.out
//STDERR   DD SYSOUT=*          < Java System.
//CEEDUMP  DD SYSOUT=*
//MAINARGS DD *
arg.number.3 'argument4 with embedded spaces and newline'
/*
//DELETE EXEC PGM=IDCAMS,COND=(7,LT)
//SYSIN    DD *
  DELETE MY.BIGFILE.BACKUP
   :
```

Simple class invocation

Environment set within the JCL

Redirection of the output

Support for Java-style parameters

Use the Java system.exit(rc)

A richer Java batch environment

- But how about some of those other requirements?
  - A “hot” environment?
  - A scheduling environment?
  - Administration & monitoring?
  - Transactionality?
  - Checkpoints?
  - Leveraging JEE assets?
  - Parallel execution?
  - Etc?

Running Java in JEE

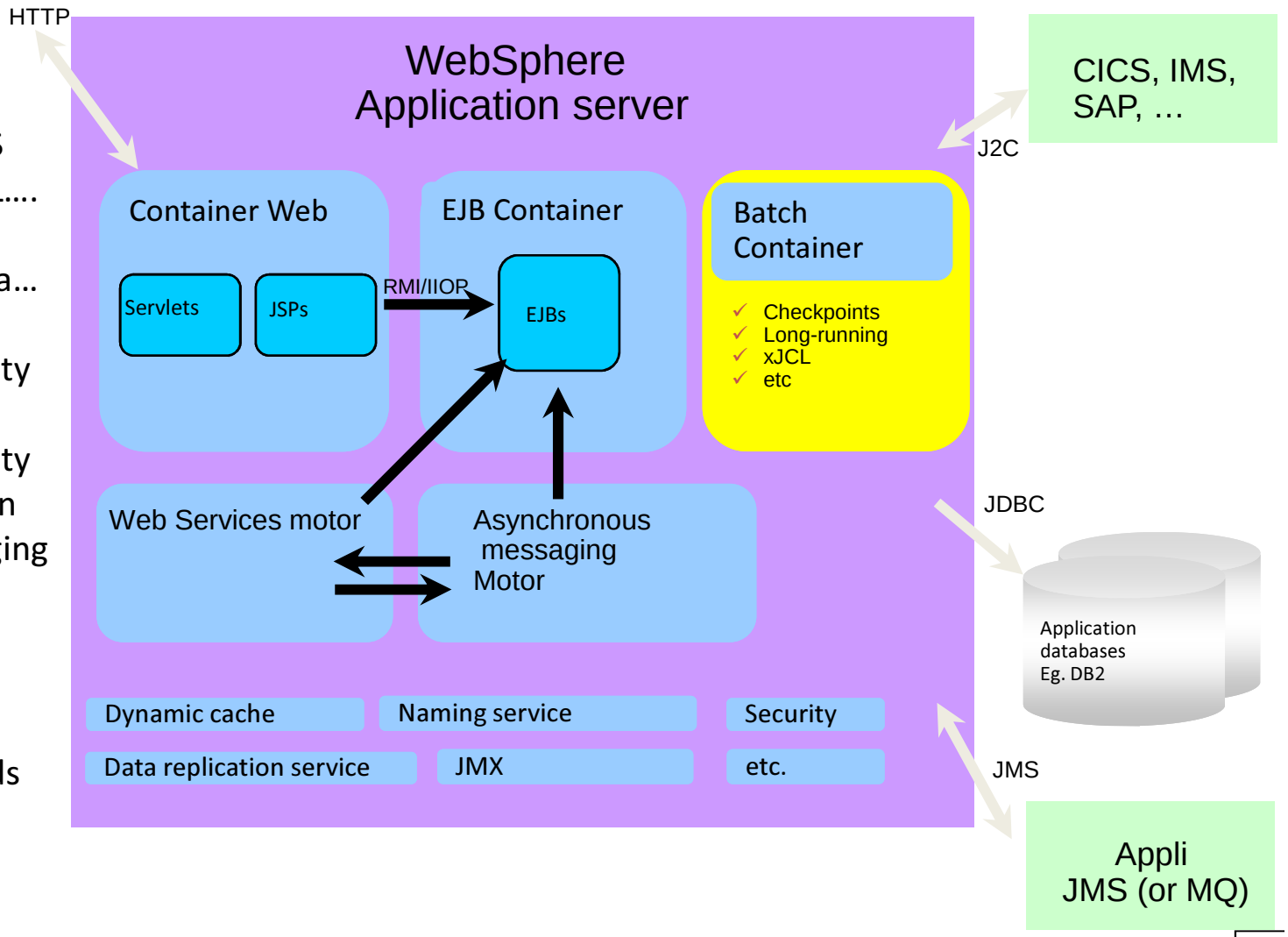
Just as IMS & CICS brought to COBOL….

WAS brings to Java…

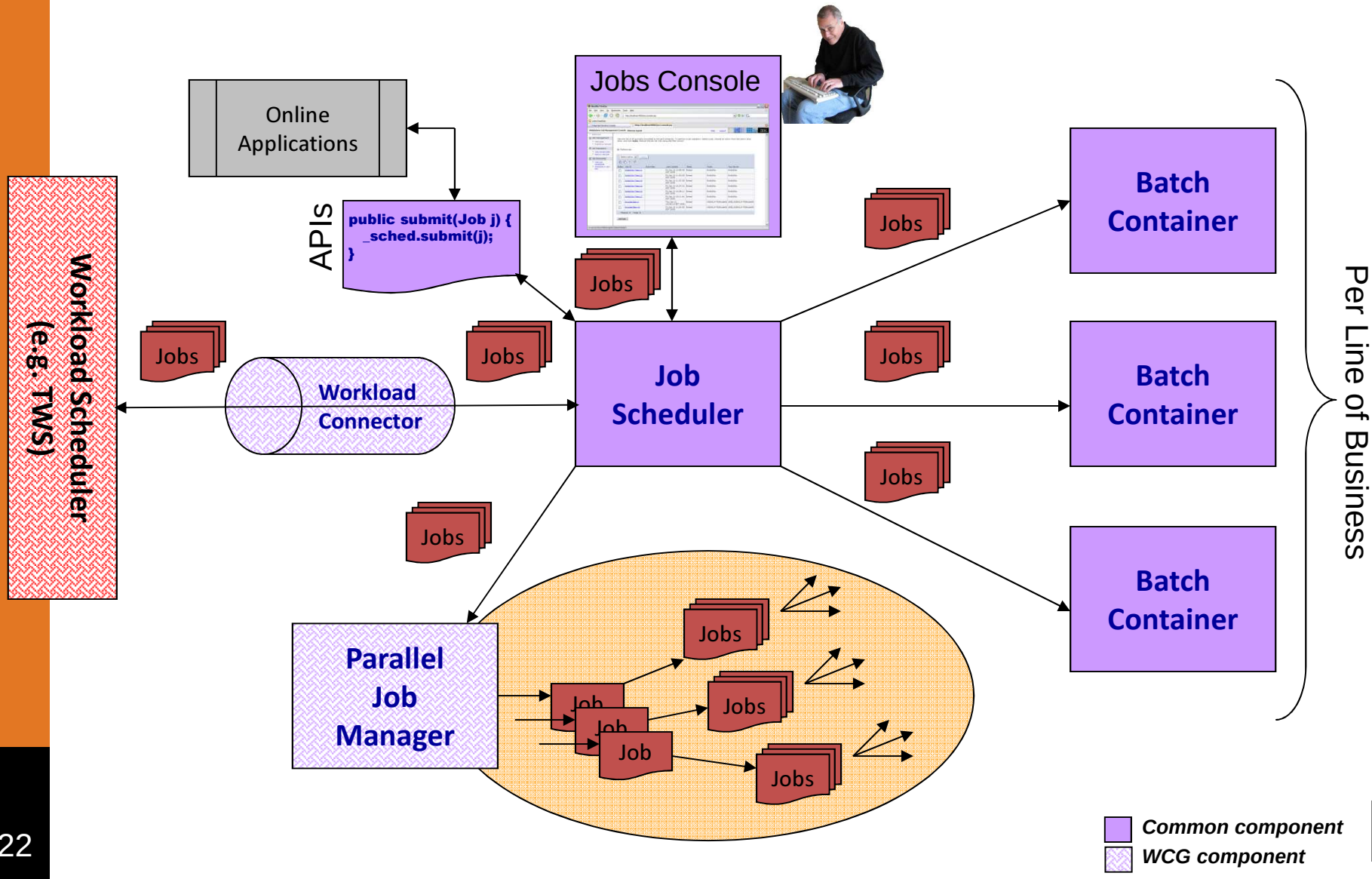
- Dispatching
- High availability
- Security
- Transactionality
- Administration
- Common logging
- Etc.

And…

- Heterogeneity
- Open standards



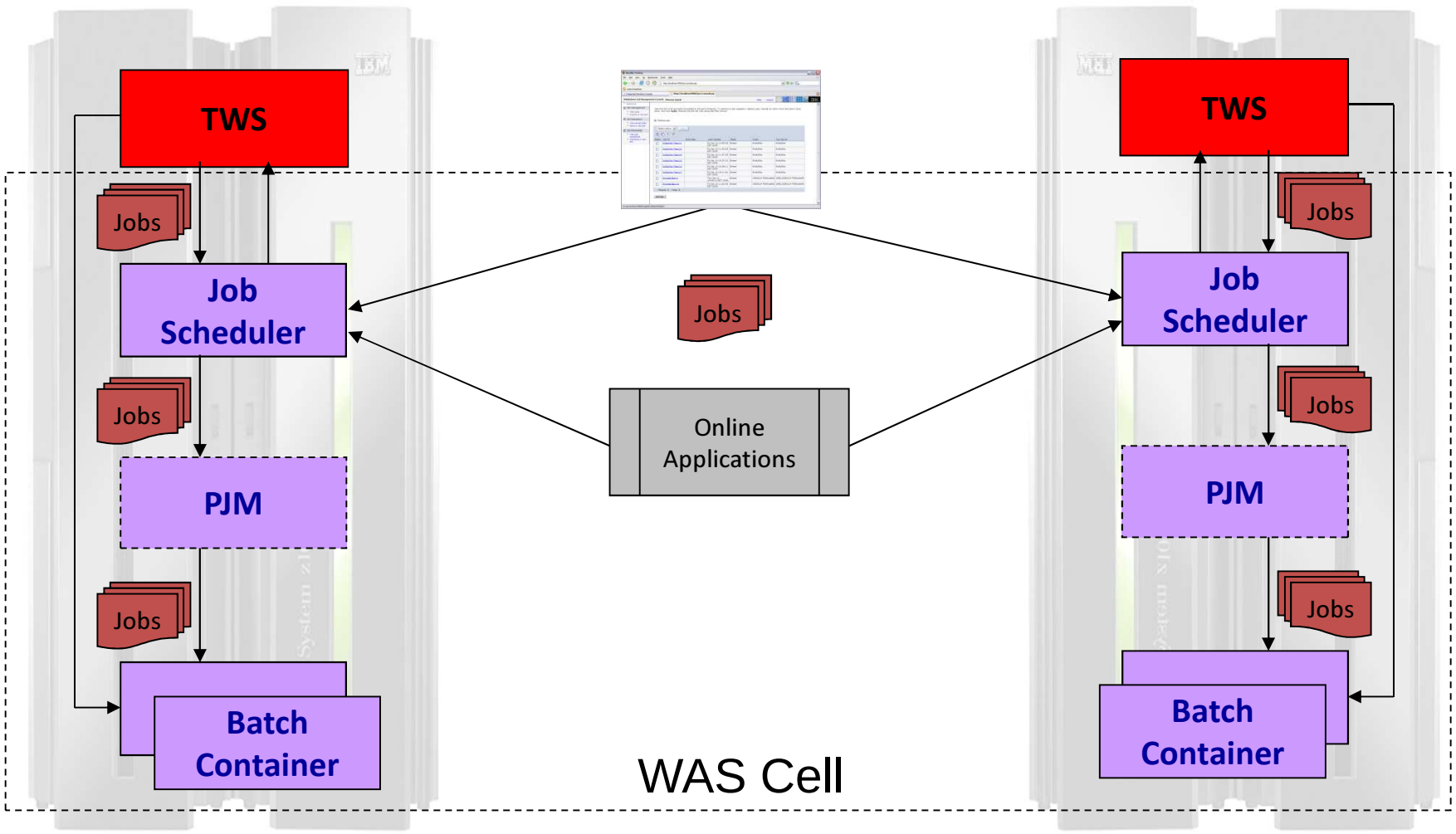
WebSphere Modern Batch Feature Pack and Compute Grid



Major Components

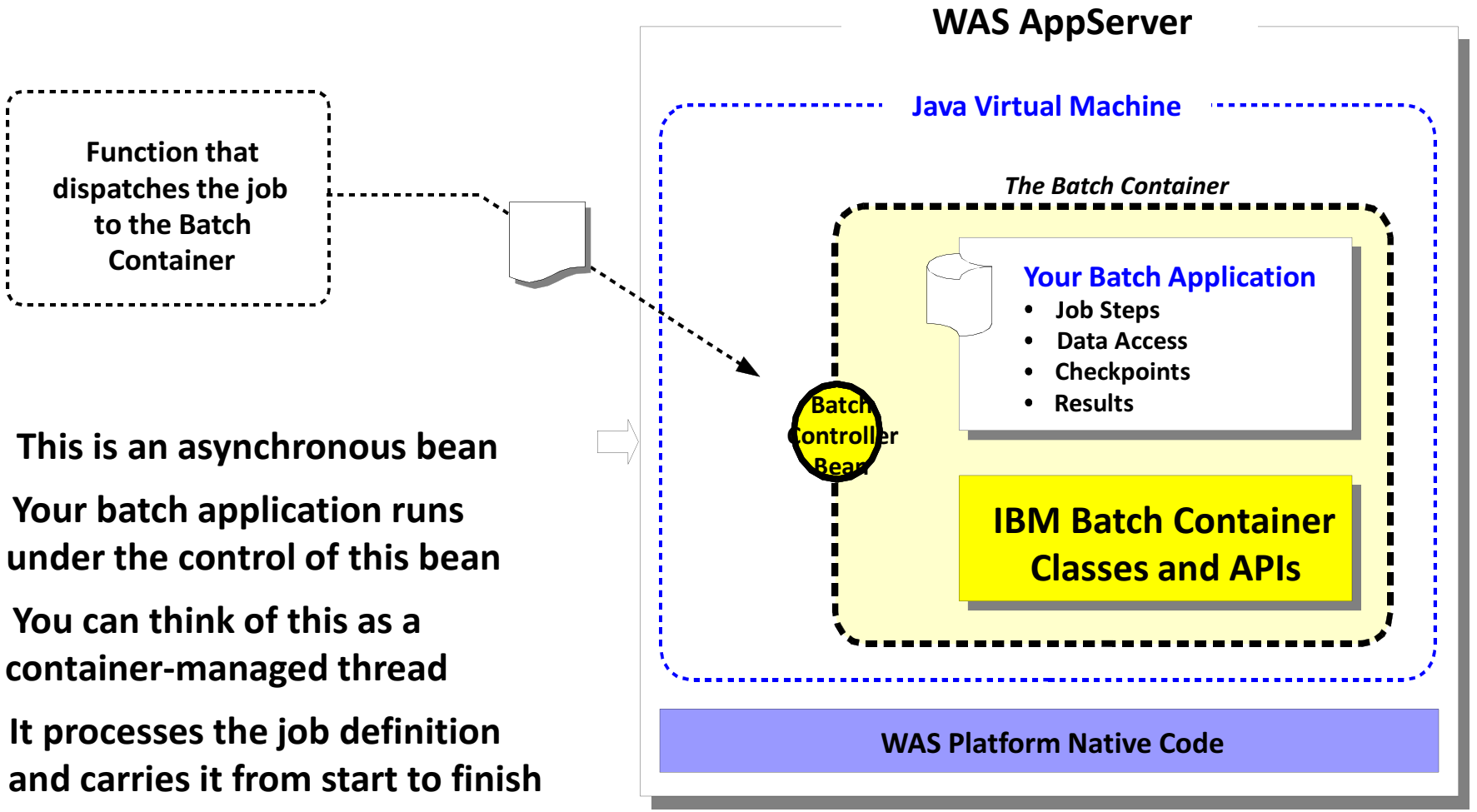
Name	Description
1. Batch Container (alias End Point, “GEE”, “LREE”..)	Provides execution environment for running a batch job. This is where the customer’s application runs.
2. Job Scheduler (alias “Dispatcher”)	Provides job management control point: – receives “run job” requests from users – decides where/when to run jobs – supports operational commands (e.g. restart job) – provides visual job console
3. Job Console (and APIs)	Human (web) interface for submitting jobs.
4. Parallel Job Manager (WebSphere Compute Grid)	Provides control point for job parallelization/control – splits/merges parallel jobs – exposes “single logical job” operational interface
5. Workload Scheduler Connector (WebSphere Compute Grid)	Enables Workload Scheduler products (e.g. Tivoli Workload Scheduler) to control WCG batch jobs.
6. Development Tools	1.Batch Framework 2.Lightweight Batch Container (“BatchSimulator”) 3.Packaging Tool 4.xJCL Generator 5.Unit Test Server

Physical Deployment – Typical z/OS



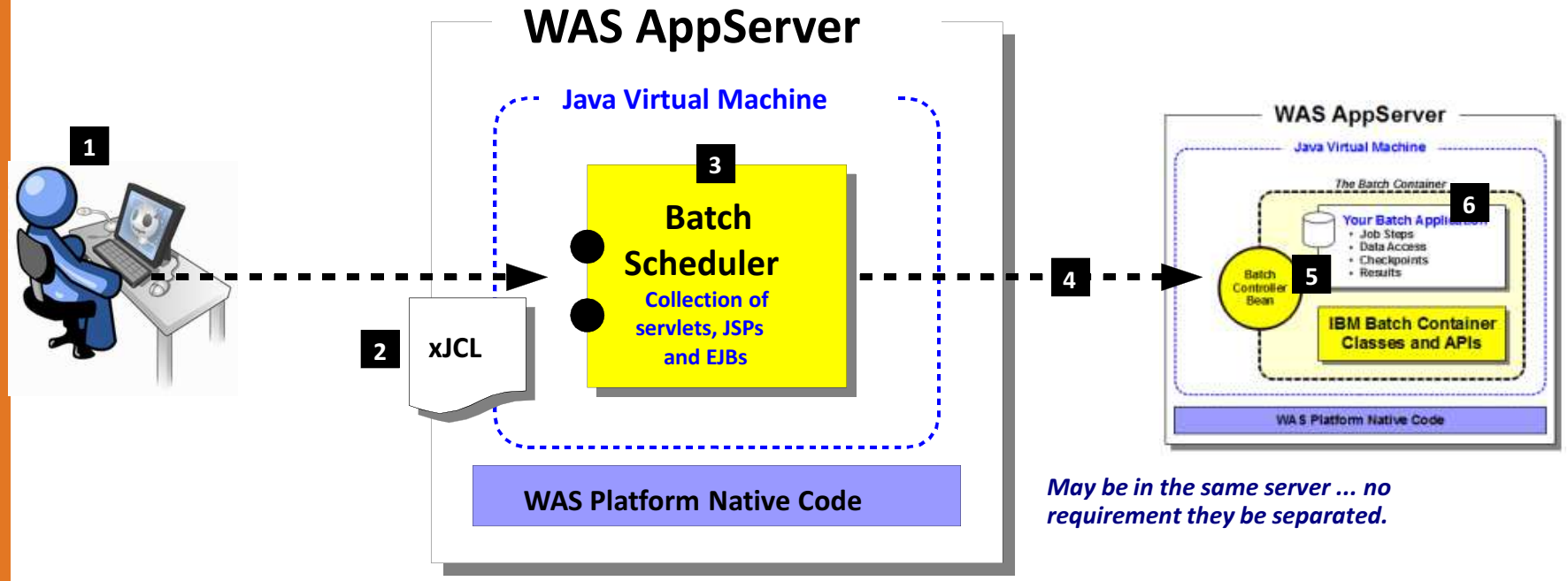
At the Heart -- The "Batch Container"

This is inside both the Feature Pack and the fuller function Compute Grid product.
The Batch Container is what truly separates this from JVM launchers



- 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

Jobs, Job Control, Scheduler, Dispatching and Batch Apps



May be in the same server ... no requirement they be separated.

1. User requests a job be submitted
Using a browser, a command line client, a Web service Client, IIOP client, MDB (WCG)

2. The Job Control Definition is specified
This is XML and is in many ways just like JCL

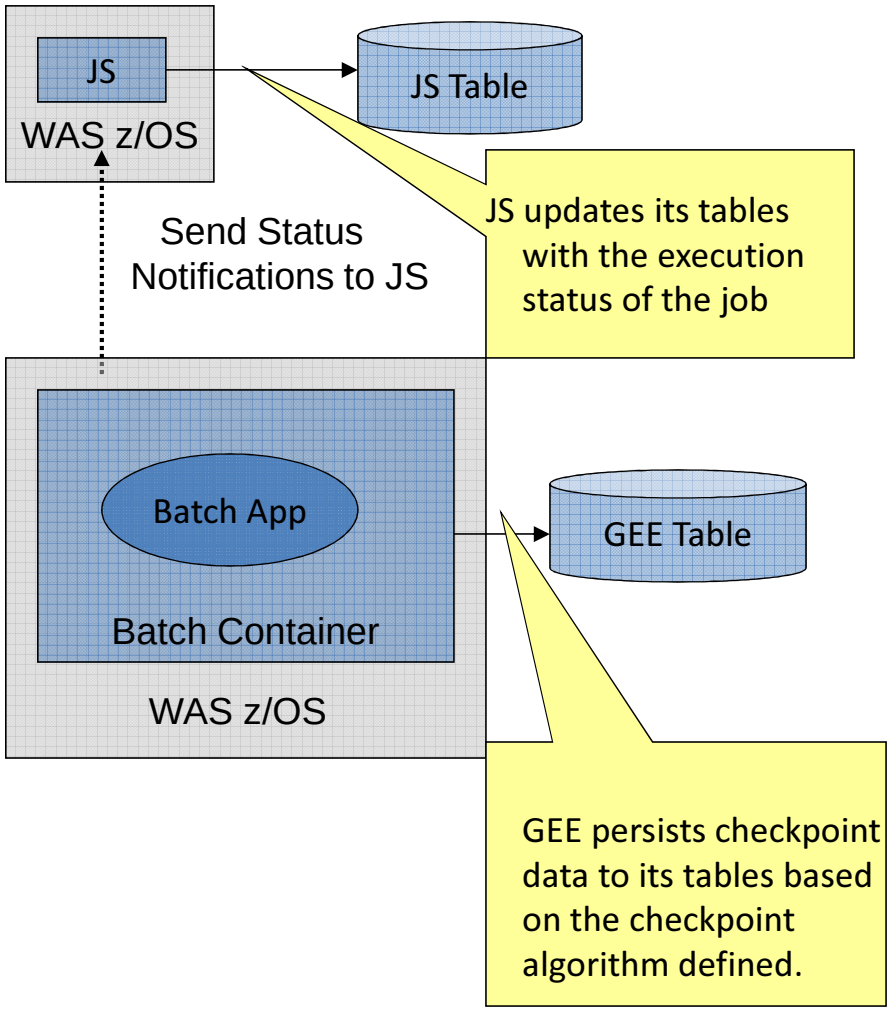
3. The Scheduler analyzes the request
It reads the xJCL, determines what's being asked, performs any substitutions and determines where that batch program runs

4. The job is dispatched to a batch endpoint
If clustered then the scheduler determines which best at that time

5. The batch endpoint begins execution
An instance of the batch controller asynch bean is invoked and it carries out the batch job lifecycle based on the definitions in the xJCL

6. Your batch application is invoked
Your *application* is deployed separately and sits waiting to be invoked by the asynch bean

Executing Batch Jobs

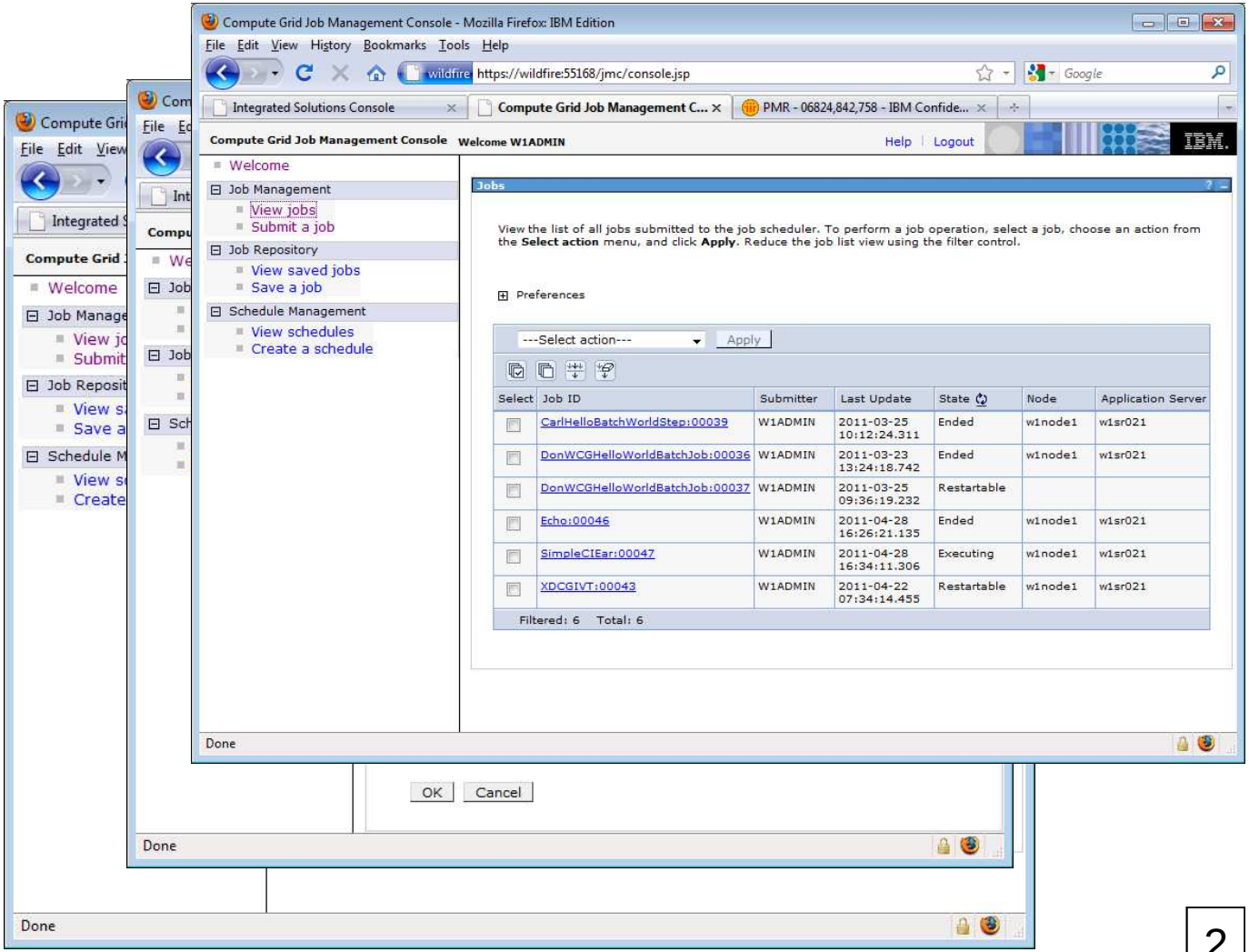


- The JS listens for execution updates- Job Failed, Job Executed Successfully, etc and updates the JS table accordingly.

- The batch application executes in the Batch Container with the properties specified in the xJCL that was submitted. During execution checkpoint data, such as current location within the batch data streams, is persisted to the GEE table for restartability.

WebSphere Compute Grid: job management console

- Through the job management console, you can:
  - Submit jobs
  - Monitor job execution
  - View job logs
  - Manage job schedules
  - Role-based Access control (EJBROLE with SAF control on z/OS)



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>
```

Roughly analogous to the JOB card

```
    <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>
```

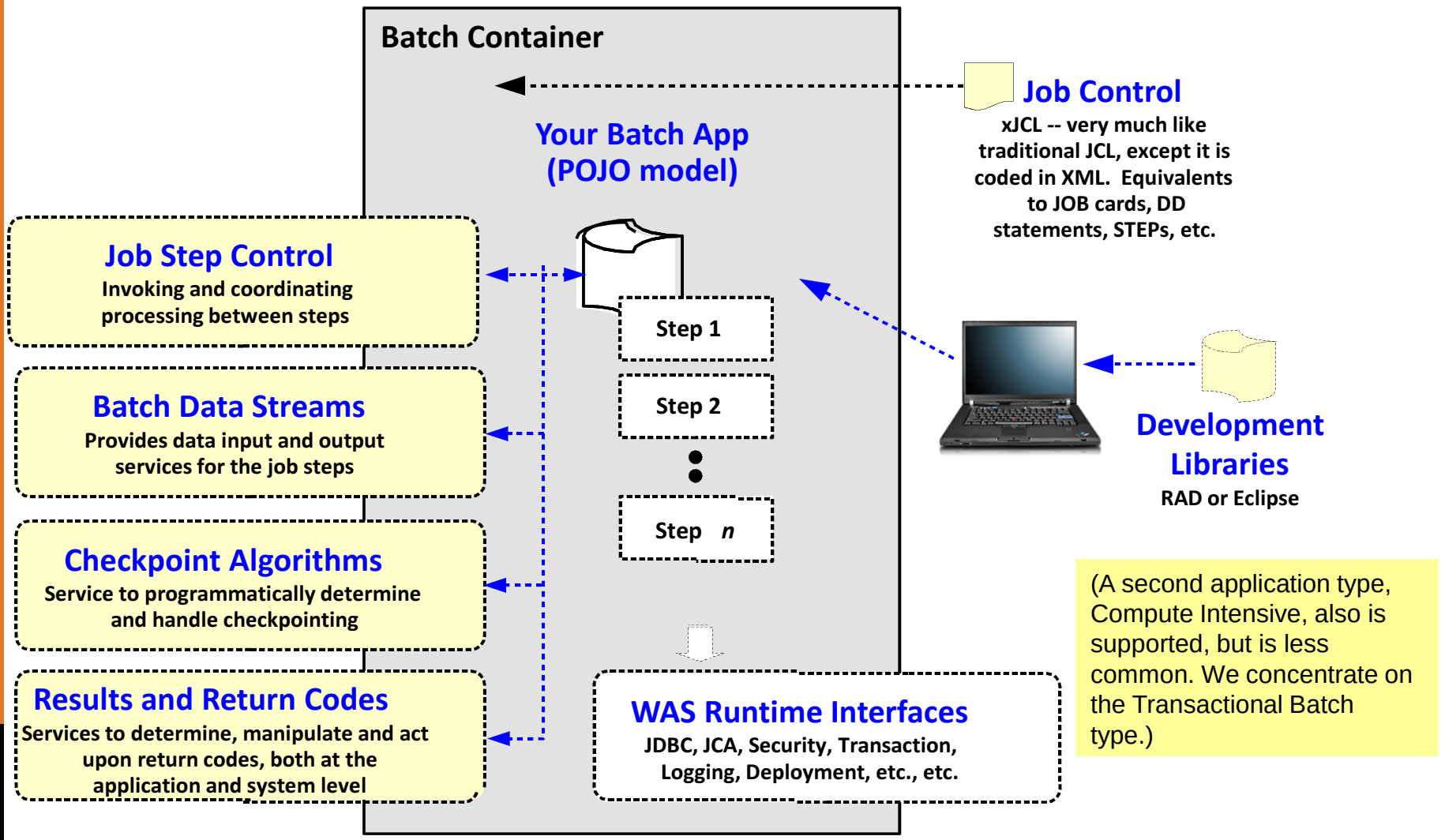
A job step

Like the EXEC PGM= statement in JCL

Similar to DD statements

The Batch Programming Model

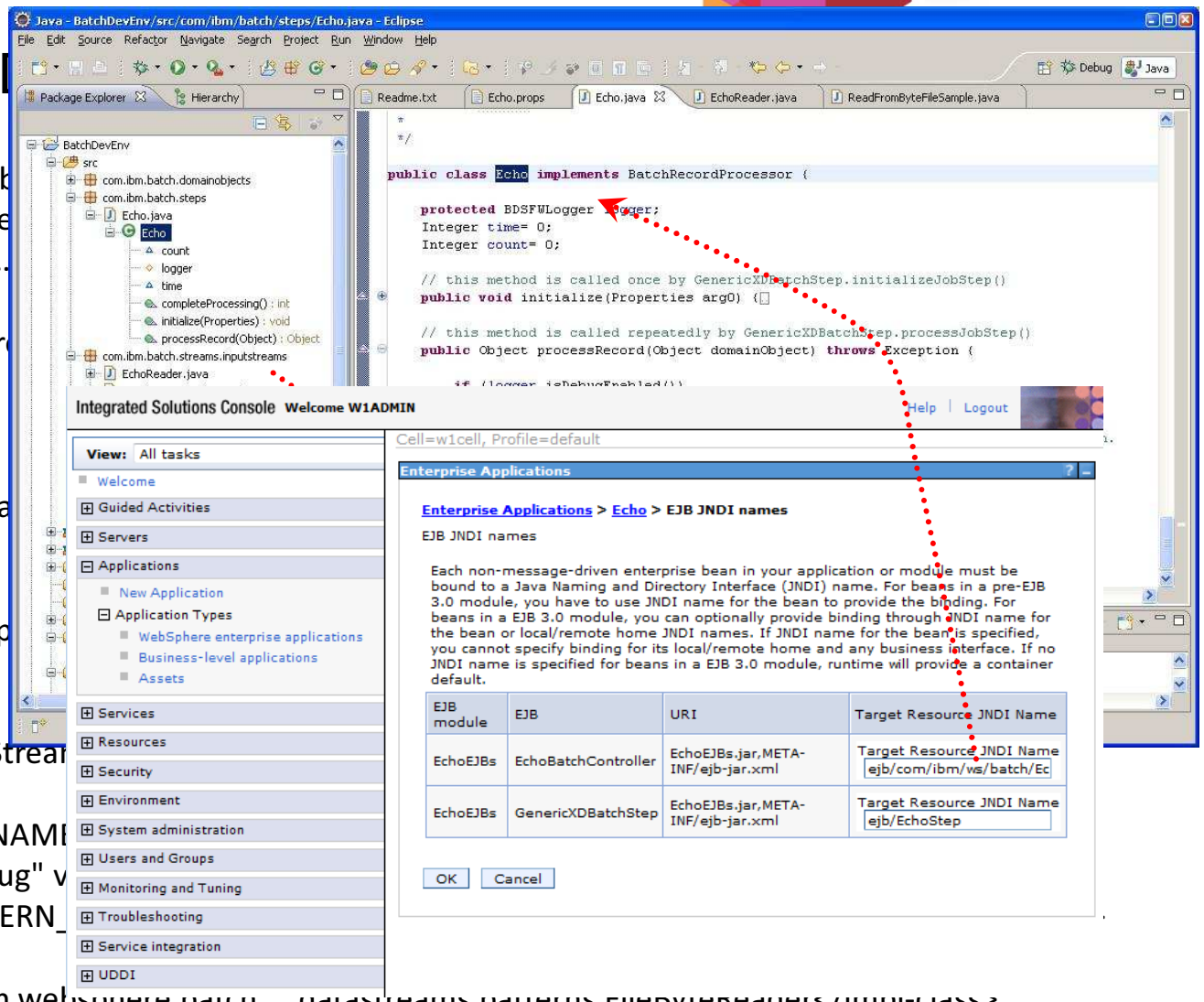
Functions and class libraries supplied with the Feature Pack and Compute Grid



(A second application type, Compute Intensive, also is supported, but is less common. We concentrate on the Transactional Batch type.)

The Job Control I

```
<job name="Echo" ...>
      <jndi-name>ejb/com/ib
      <checkpoint-algorith-re
      <classname>com.ibm...
       <props>
            <prop name="r
      </props>
 </checkpoint-algorithm>
<substitution-props>
     <prop name="echo.data
 </substitution-props>
 <job-step name="Step1">
   <jndi-name>ejb/EchoStep
    <batch-data-streams>
     <bds>
      <logical-name>inputStrea
      <props>
        <prop name="FILENAM
         <prop name="debug" v
        <prop name="PATTERN_
       </props>
        <impl-class>com.ibm.websphere.batch.....datastreams.patterns.FileByteReader</impl-class>
     </bds>
    </batch-data-streams>
 </job-step>
 </job>
```

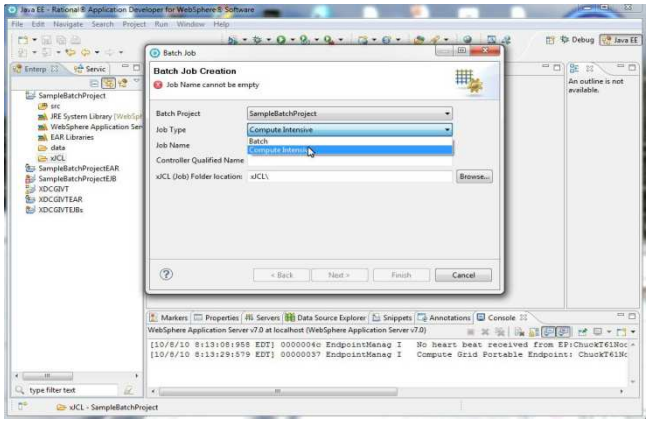


Rational Tool Support

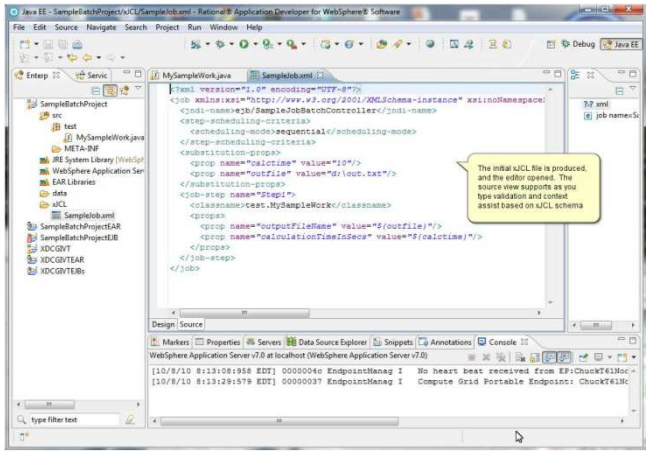
Available now as of Rational v8.0.1

- New Project Type for WebSphere Batch application
- Project and Job Creation Wizards help guide user through batch programming model semantics
- Editors for xJCL Job Control
- Source code wizards for Java code skeleton
- Menu choices for
  - generating required additional files
  - deploying to unit test or remote Compute Grid runtime environment
  - submitting the batch job Job output log viewer
- Samples / Help

Wizard-driven



xJCL Editor



Unit Test Support

WebSphere Compute Grid installation & configuration

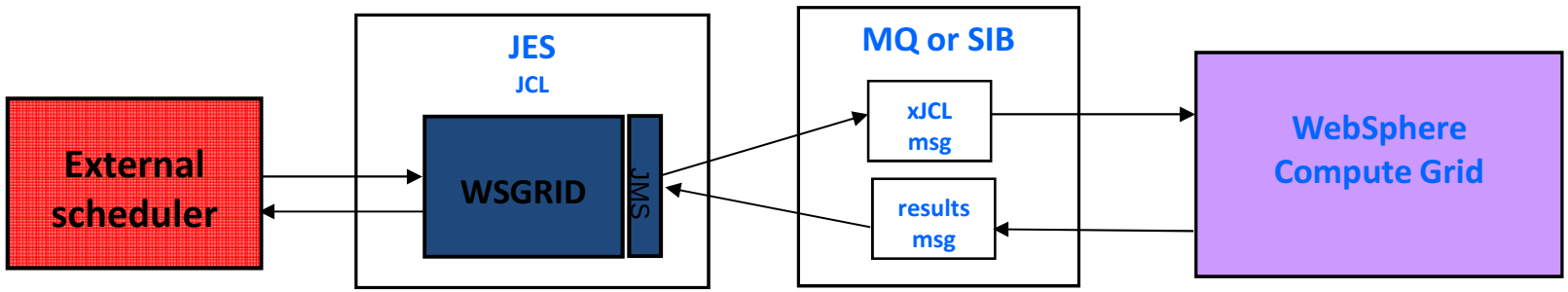
1. Install the product files
   - ✓ Classic SMP/E installation on z/OS to create binary HFS for WCG
   - ✓ Download and install WCG plugin into WCT
2. Augment the WAS Profiles
   - ✓ Begin with pre-existing Cell (DMgr + Nodes, or Standalone)
   - ✓ Standard WebSphere profile augmentation; WCT builds jobs for z/OS. Run tw jobs per node
3. Create JDBC Provider, DataSources and initialize database
   - ✓ Standard WAS configuration task – nothing special; sample scripts provided for DB2 z/OS & LUW, Oracle, Derby
4. Configure the Job Scheduler and the Grid End Points
   - ✓ Simple AdminConsole menus were added with the augmentation
   - ✓ Need to deploy a WCG EAR to each node to “enable” it
5. Validation
   - ✓ Several sample applications are supplied

Not necessary with WAS v8.5

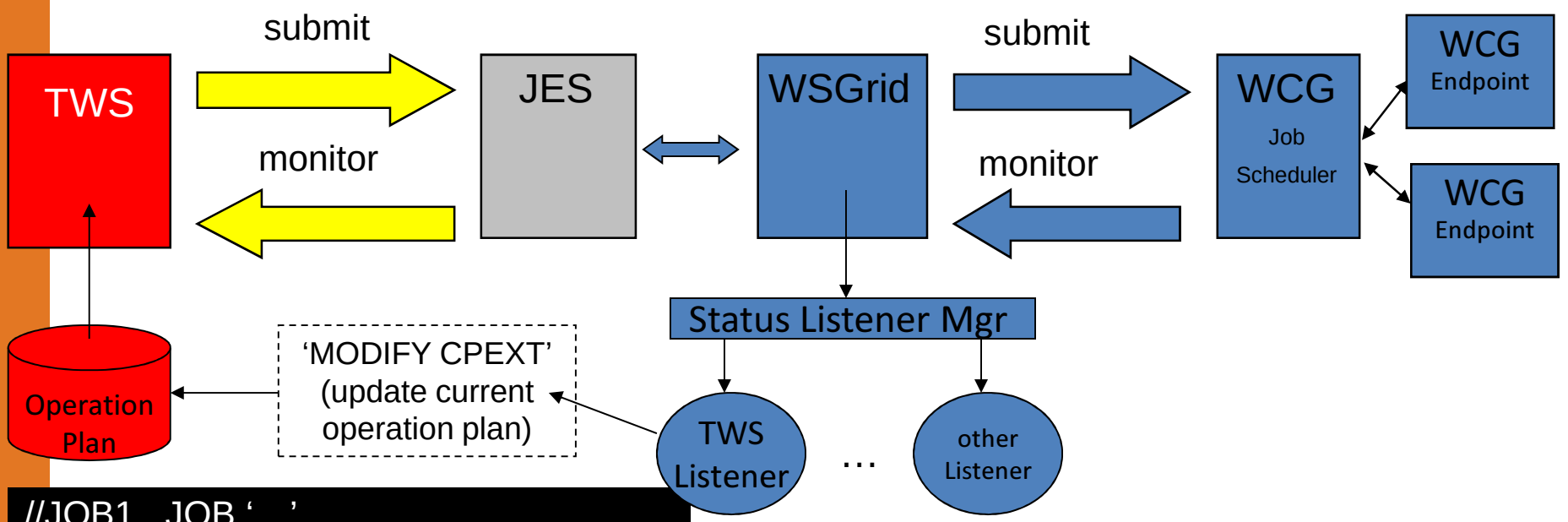
- *The installation is fully documented in the InfoCenter at* http://publib.boulder.ibm.com/infocenter/wxdinfo/v6r1m1/index.jsp
- *Also, there is an excellent additional guide from the WSC at* http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101783

WebSphere Compute Grid: External Scheduler integration

- An External Scheduler (eg. TWS, Control-M, Zeke…) invokes WCG via the WSGRID "proxy"
- WSGRID is a utility supplied by WCG, it is invoked by the external scheduler using JCL
  - WSGRID utility is a client application of the Job Scheduler message-driven interface; it is used to submit to submit jobs into a WebSphere XD Compute Grid topology.
  - The job log from the WebSphere batch job is written to the standard output stream of the environment in which the WSGRID utility was invoked
- WSGrid offers two implementations:
  - Native code with MQ bindings (most efficient, but only available on z/OS)
  - JEE client code using SIBus (can be used if MQ is not available)



External Scheduler Integration … TWS example




```
//JOB1   JOB '…'
//STEP1 PGM=IDCAMS
//STEP2 PGM=WSGRID
//SYSIN  DD *
// jndi-name="ejb/DOIT"
// …
```

```
<job name="JOB1" …
<job-step name="STEP2">
<jndi-name>"ejb/DOIT"</jndi-
name>
…
```

Example WSGRID native bindings

Invoke IBM-supplied WSGRID WCG interface

QMGR and queues monitored by Job Scheduler

WCG xJCL

Substitution variables

```
//CFGRID    JOB NOTIFY=FARKAS,CLASS=A,REGION=0M,
// MSGCLASS=H,USER=DCADMIN,PASSWORD=GRIDMAN
/*JOBPARM    SYSAFF=ZT01
//*****************************************************************
//* Submit a WebSphere Compute Grid (WCG) job
//*****************************************************************
//RUN EXEC PGM=WSGRID
//STEPLIB DD DSN=FARKAS.CG.LOAD,DISP=SHR
//        DD DSN=WMQ.V7R0M1.SCSQLOAD,DISP=SHR
//        DD DSN=WMQ.V7R0M1.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//*****************************************************************
//* WCCNTL specifies MQ input/output queues for WCG
//*****************************************************************
//WGCNTL DD *
queue-manager-name=CBF2
scheduler-input-queue=WCG_INPUT
scheduler-output-queue=WCG_OUTPUT
/*
//*****************************************************************
//* WGJOB specifies the xJCL job to submit in EBCDIC
//*****************************************************************
//WGJOB DD PATH='/u/farkas/SimpleCIxJCL_FP.ebcdic.xml'
//*****************************************************************
//* WGSUBS specifies any substitution properties
//*****************************************************************
//WGSUBS DD *
substitution-prop.calcTime=30
substitution-prop.outFile=/tmp/SimpleCIout.txt
/*
```

Example WSGRID sysprint

```
 SDSF OUTPUT DISPLAY CFGRID   JOB08589  DSID   104 LINE 0       COLUMNS 02- 133
 COMMAND INPUT ===>                                          SCROLL ===> CSR
******************************** TOP OF DATA ***********************************
WSGrid Version WASX.XDNAT [cf61045.07] 2010-11-08 15:35:48
WGCNTL   DD: queue-manager-name=CBF2
WGCNTL   DD: scheduler-input-queue=WCG_INPUT
WGCNTL   DD: scheduler-output-queue=WCG_OUTPUT
WGJOB    DD: <job name="SimpleCIEar" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
WGJOB    DD: <jndi-name>ejb/com/ibm/ws/ci/SimpleCIEJB</jndi-name>
WGJOB    DD: <substitution-props>
WGJOB    DD: <prop name="calcTime" value="30" />
WGJOB    DD: <prop name="outFile" value="/tmp/SimpleCIout.txt" />
WGJOB    DD: </substitution-props>
  :
INFO: Using output queue name= WCG_OUTPUT
INFO: Putting job on MQ queue WCG_INPUT
INFO: Job put on queue - RC=0
INFO: Waiting for jobid assignment.
CWLRB5020I: Wed Apr 03 05:54:27 GMT 2013 : Job [55] has been submitted for execution.
CWLRB5020I: Wed Apr 03 05:54:27 GMT 2013 : Job [SimpleCIEar:00055] has been submitted for execution.
CWLRB5671I: [04/03/13 05:54:27:496 GMT] Processing for job SimpleCIEar:00055 started.
Original XJCL
   1 : <job name="SimpleCIEar" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   2 :     <jndi-name>ejb/com/ibm/ws/ci/SimpleCIEJB</jndi-name>
   3 :     <substitution-props>
   4 :         <prop name="calcTime" value="30"></prop>
   5 :         <prop name="outFile" value="/tmp/SimpleCIout.txt"></prop>
    :
  17 : </job>
CWLRB5684I: [04/03/13 05:54:27:509 GMT] Job SimpleCIEar:00055 is queued for execution
CWLRB5586I: [04/03/13 05:54:27:521 GMT] CWLRS6006I: Job class Default, Importance 8, Service Class null, Service Goal Type 0, Applic
ation Type j2ee, Submitter DCADMIN.
CWLRB5586I: [04/03/13 05:54:27:522 GMT] CWLRS6007I: Job Arrival Time 4/3/13 5:54 AM, Goal Max Completion Time 0, Goal Max Queue Time
 0, Breach Time 4/4/13 5:54 AM.
CWLRB5586I: [04/03/13 05:54:27:524 GMT] CWLRS6021I: List of eligible endpoints to execute the job: dcnode1/dcsr021.
CWLRB5586I: [04/03/13 05:54:27:527 GMT] CWLRS6011I: APC is not active. GAP will make the endpoint selection.
CWLRB5586I: [04/03/13 05:54:27:535 GMT] CWLRS6013I: GAP is dispatching job SimpleCIEar:00055. Job queue time 0.023 seconds.
CWLRB3090I: [04/03/13 05:54:27:556 GMT] Job SimpleCIEar:00055 is dispatched to endpoint dcnode1/dcsr021: result: 0
CWLRB5588I: [04/03/13 05:54:27:578 GMT] Setting up j2ee job SimpleCIEar:00055 for execution in Grid Execution Environment dccell/dcn
ode1/dcsr021: [jobClass Default] [jobName SimpleCIEar] [module ] [user DCADMIN] [applicationName SimpleCIEar] [applicationType j2ee]
 [transactionClass $édefault_iiop_transaction_classè]
CWLRB5784I: [04/03/13 05:54:27:587 GMT] Setting step SLSB property: outputFileName=/tmp/SimpleCIout.txt
CWLRB5784I: [04/03/13 05:54:27:587 GMT] Setting step SLSB property: calculationTimeInSecs=30
System.out: [04/03/13 05:54:27:591 GMT] Wed Apr 03 05:54:27 GMT 2013: SimpleCI application starting...
   :
CWLRB3800I: [04/03/13 05:54:57:601 GMT] Job [SimpleCIEar:00055] ended normally.
CWLRB3880I: Job [SimpleCIEar:00055] ending status: RC=0
****************************** BOTTOM OF DATA ********************************
```

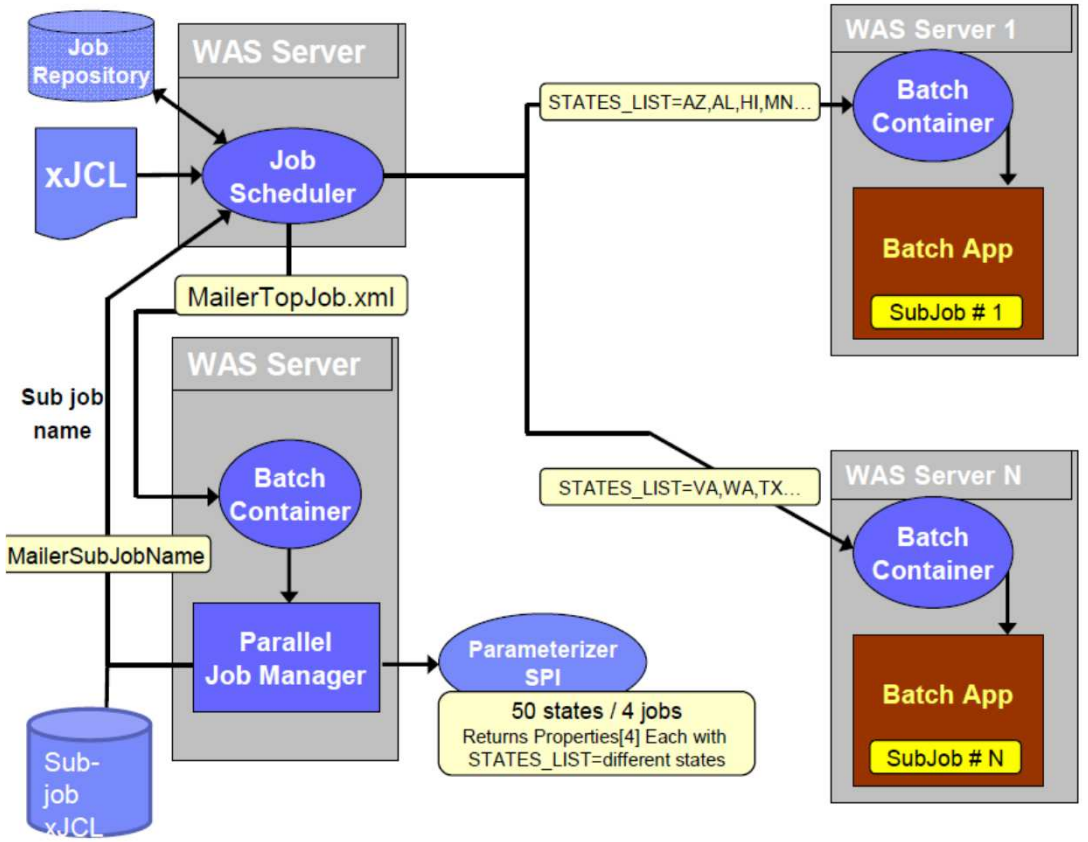
WSGRID parameters

Original xJCL

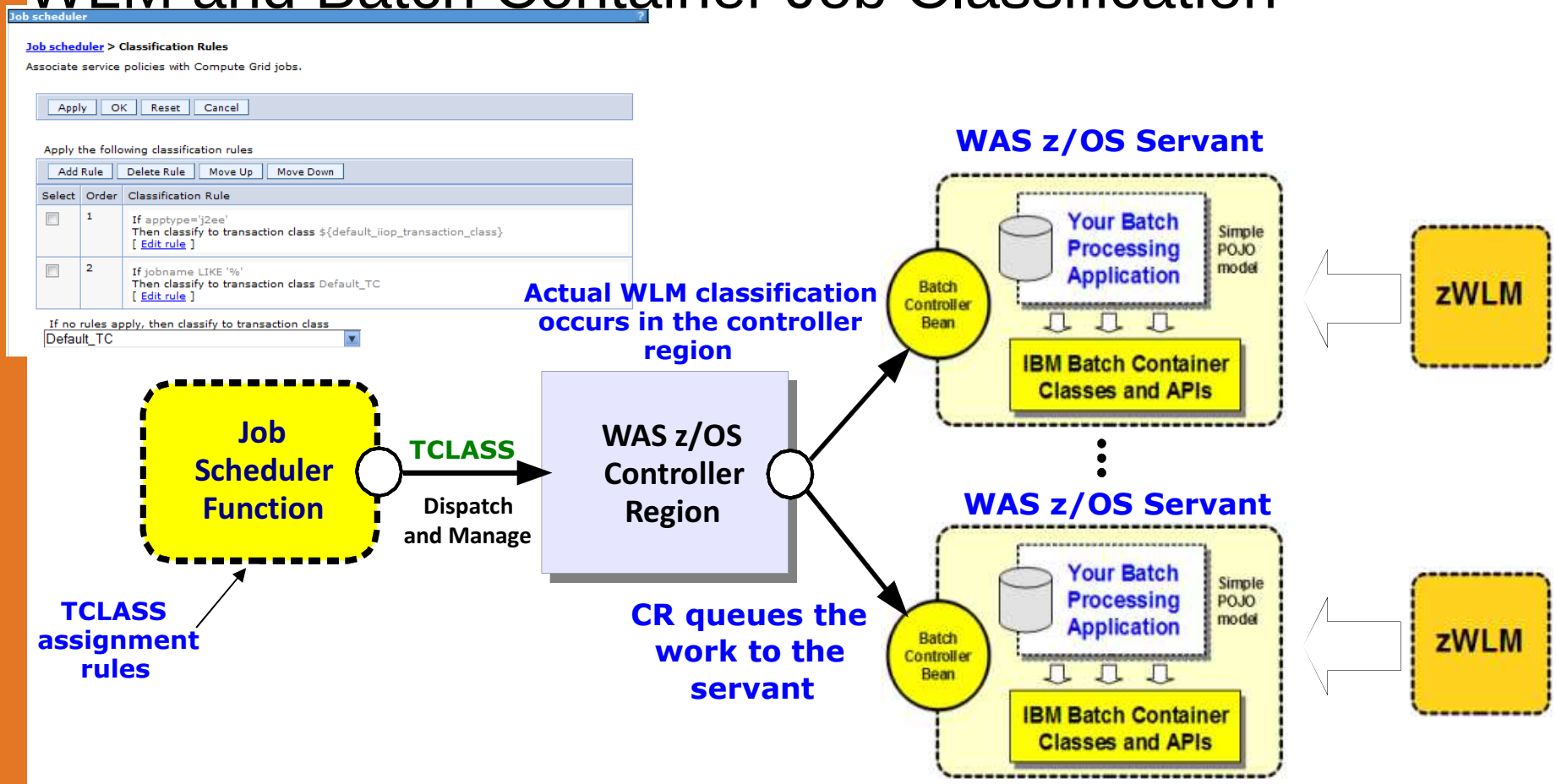
Job run

WebSphere Compute Grid: Parallel Job Manager

- PJM is a system-provided EJB application to control parallel jobs
- Installed using wsadmin/jython script
- Target of a parallel job
- PJM does not process any batch data streams
- Monitors and manages sub-jobs (which are then submitted to the JS)
- Sub-jobs are aggregated by the PJM into one logical top level job
- Optional component of Compute Grid



WLM and Batch Container Job Classification

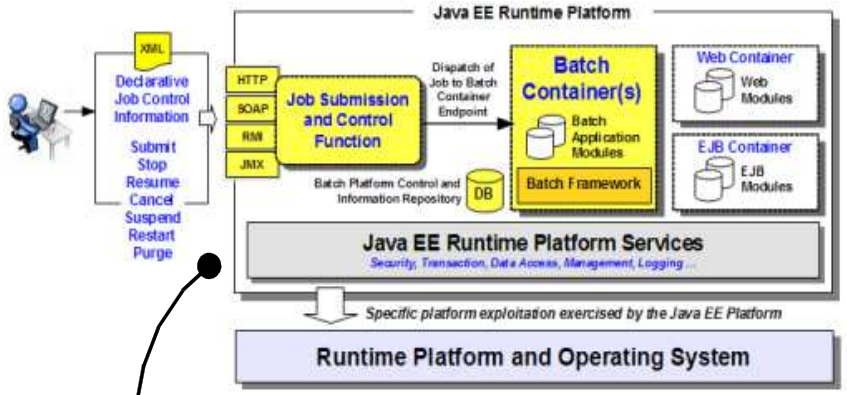


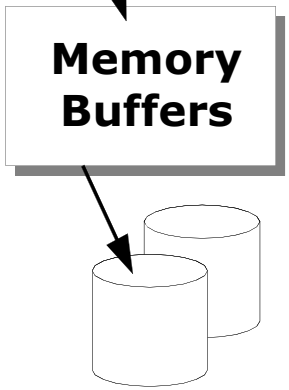
Feature Pack — Classify batch work separately from other work and allow WLM to manage all according to goals

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

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

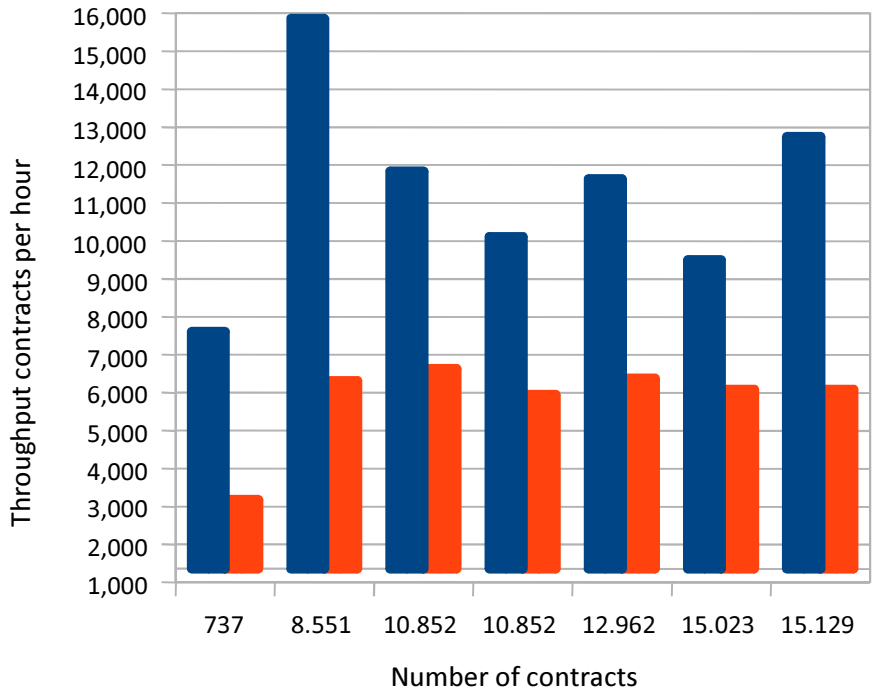
Information in SMF 120.20 record:

- **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

Co-location advantage with WCG , example DB2 data access

Performance comparison WAS AIX vs. WAS z/OS - dynamic sampling jobs

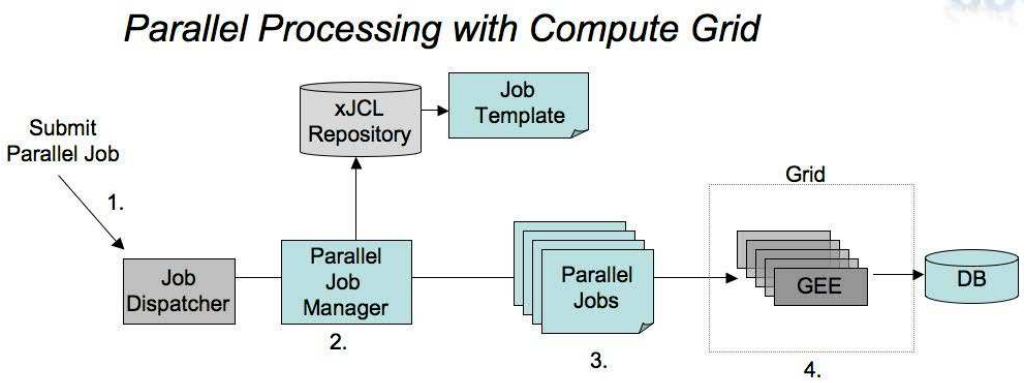


LVM Versicherungen

- “Performance can be doubled; expectations were exceeded by far”
- “Applications can run unchanged on the mainframe as well”
- “Economic efficiency can be reached with high zIIP [zAAP] rates”

WAS v6.1 with Compute Grid; AIX on p4; z/OS on z9

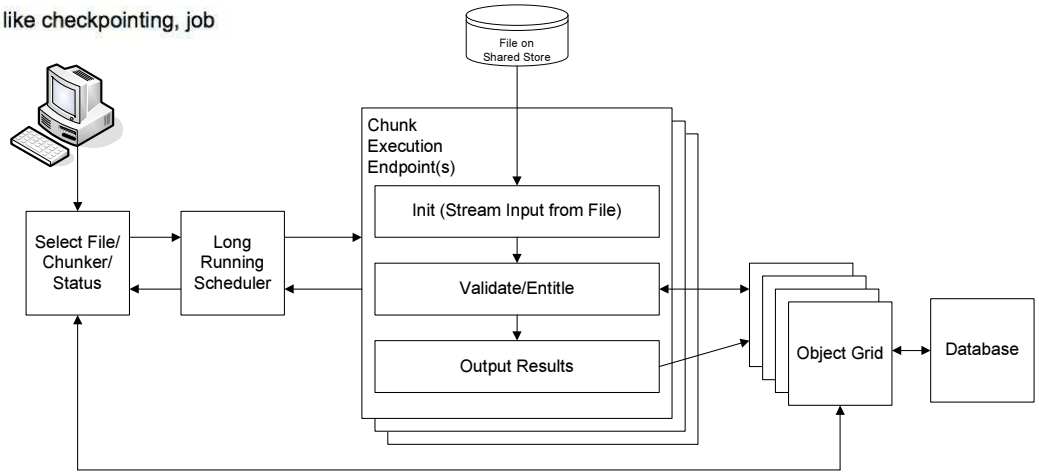
Wall St. Bank: High Performance, Highly-Parallel Batch Jobs with XD Compute Grid



1. Large, single job is submitted to the Job Dispatcher of XD Compute Grid
2. The Parallel Job Manager (PJM), with the option of using job partition templates stored in a repository, breaks the single batch job into many smaller partitions.
3. The PJM dispatches those chunks across the cluster of Grid Execution Environments (GEE)
4. The cluster of GEE's execute the parallel jobs, applying qualities of service like checkpointing, job restart, transactional integrity, etc.

Major Wall St. Bank uses the Parallel Job Manager for highly parallel XD Compute Grid jobs with eXtreme Scale for high-performance data access to achieve a cutting edge grid platform

Applying the Pattern at the bank



For additional details, see http://www.ibm.com/developerworks/websphere/techjournal/0804_antani/0804_antani.html

WebSphere Compute Grid v8

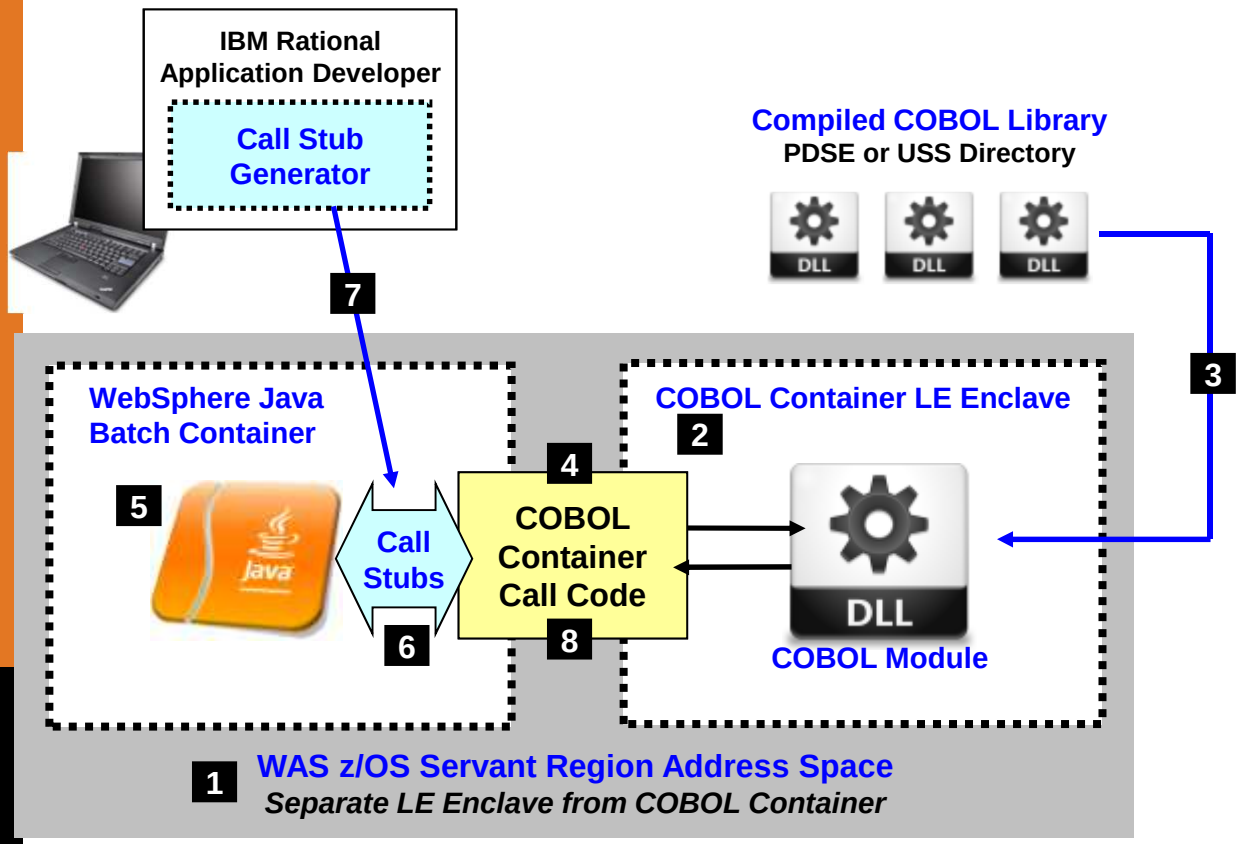
- Announced 5 April, 2011, GA for 17 June 2011
- Enable mixed language batch applications in WebSphere Compute Grid z/OS environment.
  - Increase portability of COBOL code to WebSphere Compute Grid environment so more code can be reused with less modification.
  - Increase developer productivity by providing code generator for inter-language call stubs and parameter marshalling logic.
  - Support mixed transactions (Java and COBOL in one UOW)
- Implementation: COBOL Connector + Container + Marshalling Stubs
  - Java and COBOL can execute SQL on a shared DB2 connection
  - COBOL conditions exposed as Java exceptions
  - COBOL must be recompiled with specific options
    - A few restrictions (eg. No explicit commits, rollbacks..)
- See TechDocs white paper WP101909 for additional details
  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101909

```
// get reference to COBOL container
ILContainer ilc =
ILContainerFactory.getFactory().create();
// Create the COBOL PROCEDURE stub
PRIMITVE primitve = new PRIMITVE();
// now invoke the COBOL procedure
int rc = ilc.invokeProcedure(primitve);
```

```
program-id. PRIMITVE.
...
LINKAGE SECTION.
```

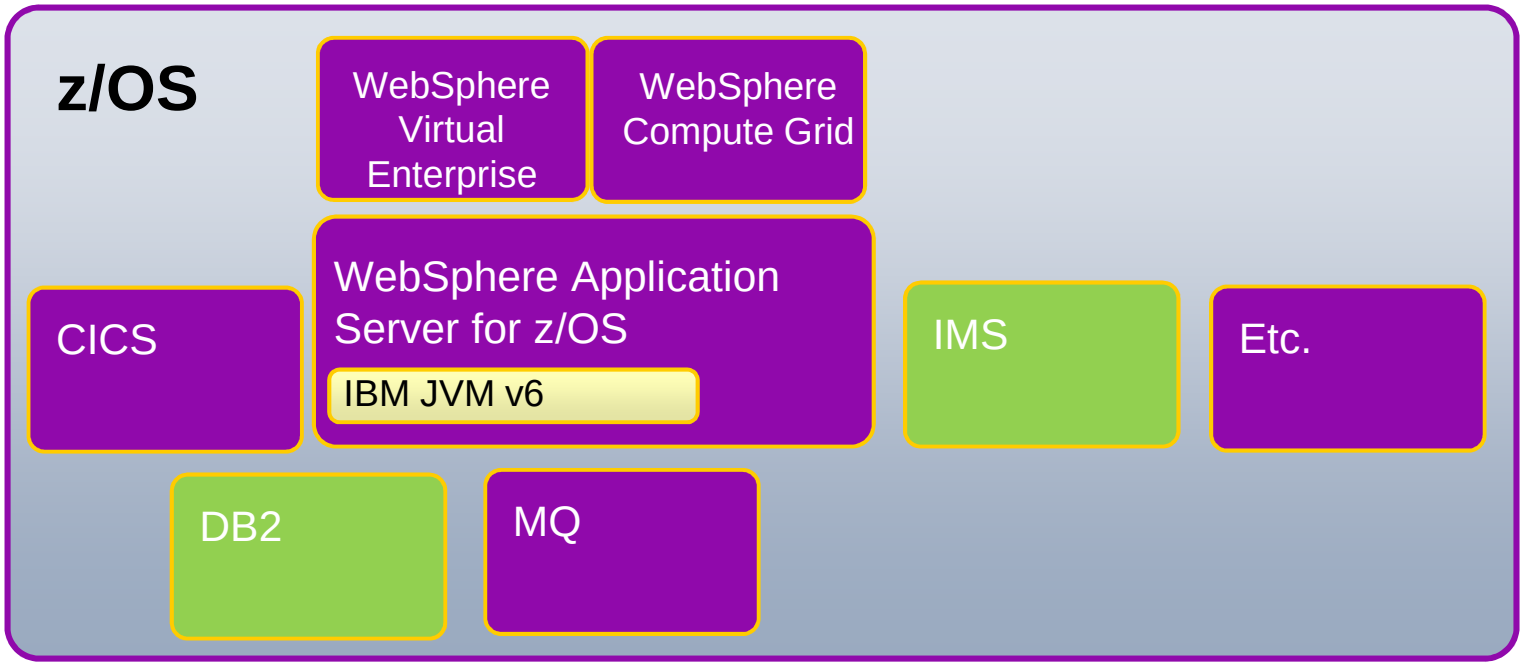
COBOL Container

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

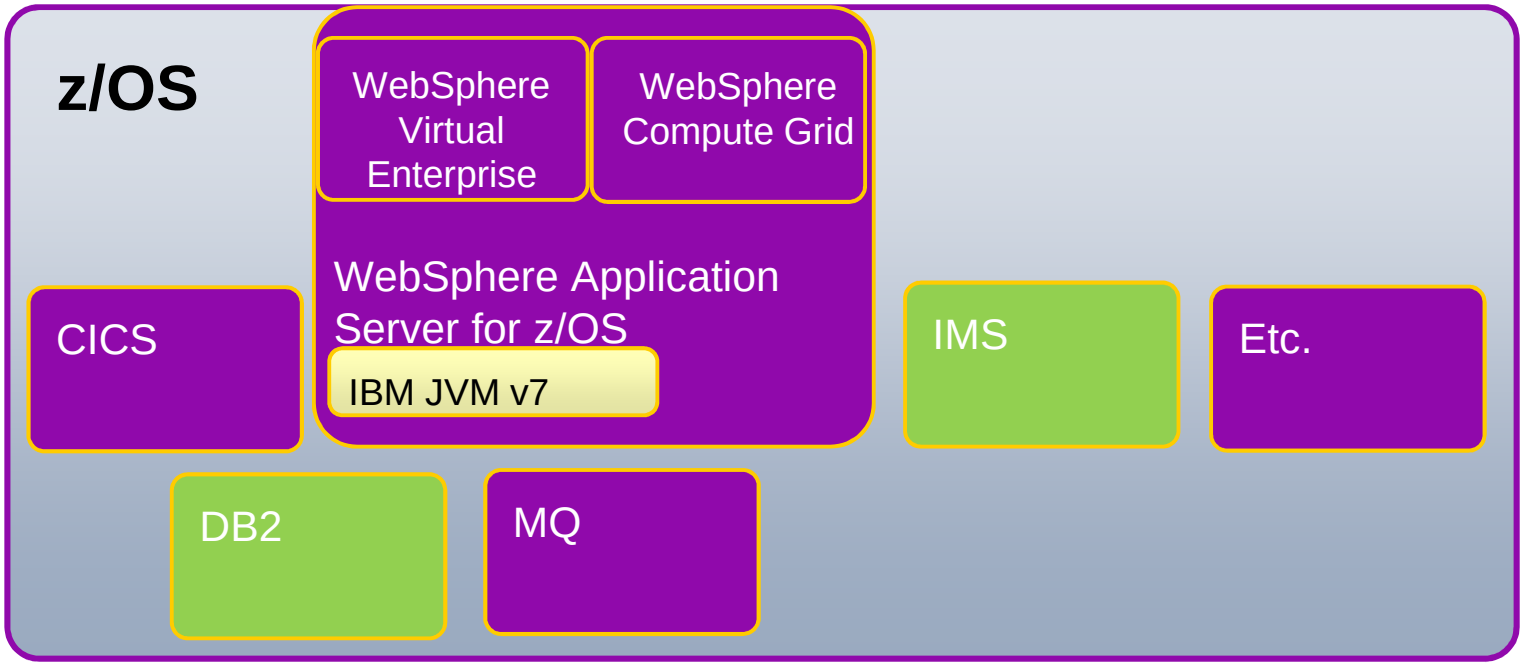


1. Batch application runs in the WAS z/OS servant region address space
2. The COBOL container is created as a separate LE enclave in the address space
3. COBOL DLLs are accessed using STEPLIB or LIBPATH
4. COBOL Container code provides the "glue" between the Java environment and the native COBOL
5. Java 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

WAS v8 in z/OS in 2011…



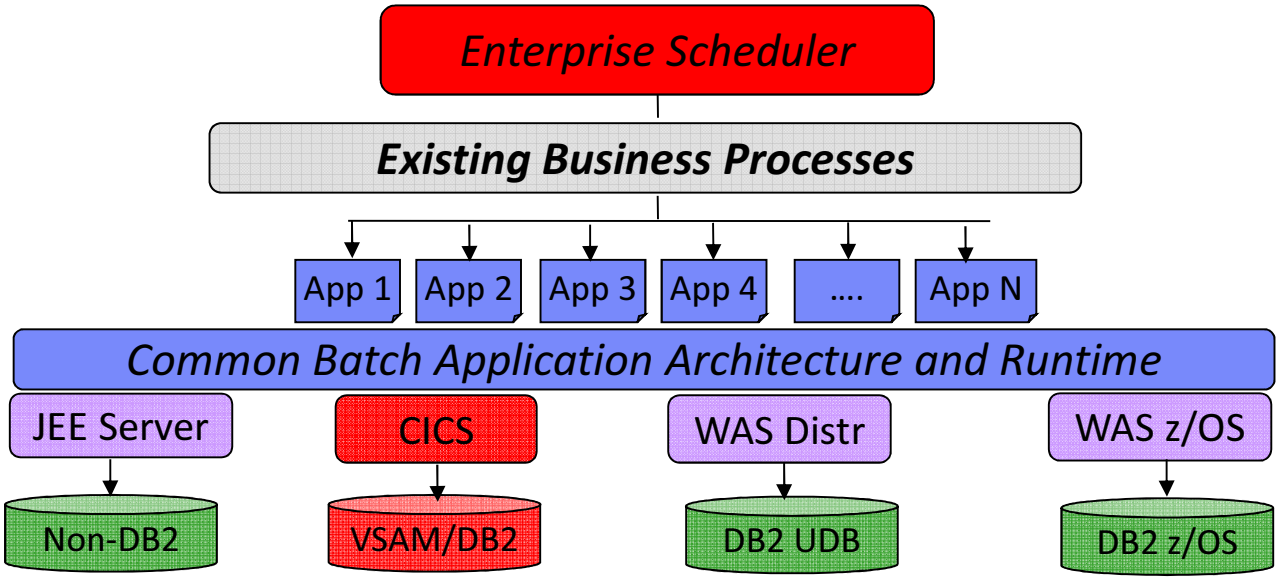
WAS v8.5 in z/OS in 2012…



WAS v8.5….

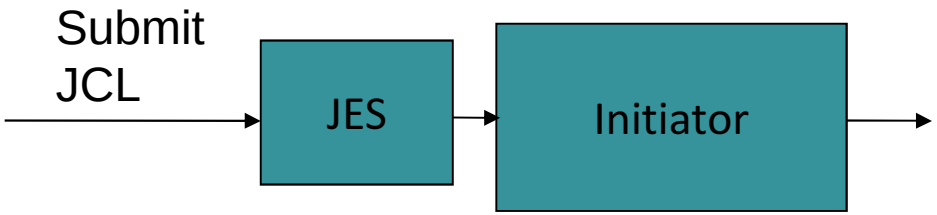
- **Adds JDK (Java) v7 support**
- **Adds WCG & WVE to WAS z/OS**
- **Adds "Liberty" profile**

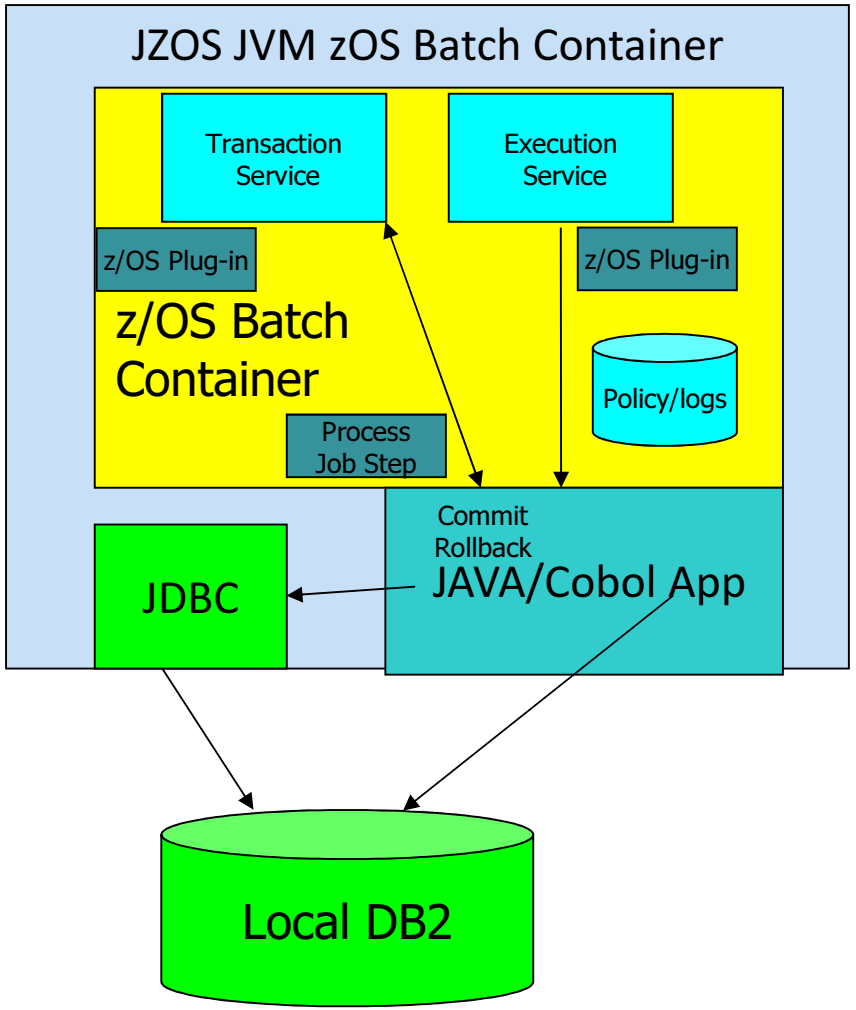
The IBM Batch Vision



1. Batch Containers should run ***everywhere***
2. ***Portable Batch applications*** across platforms and JEE vendors
3. Location of the data dictates the placement of the batch application
4. Centrally managed by your enterprise scheduler
5. Integrating with existing: Disaster Recovery, Auditing, Logging, Archiving

z/OS 1.13 Batch Runtime Environment

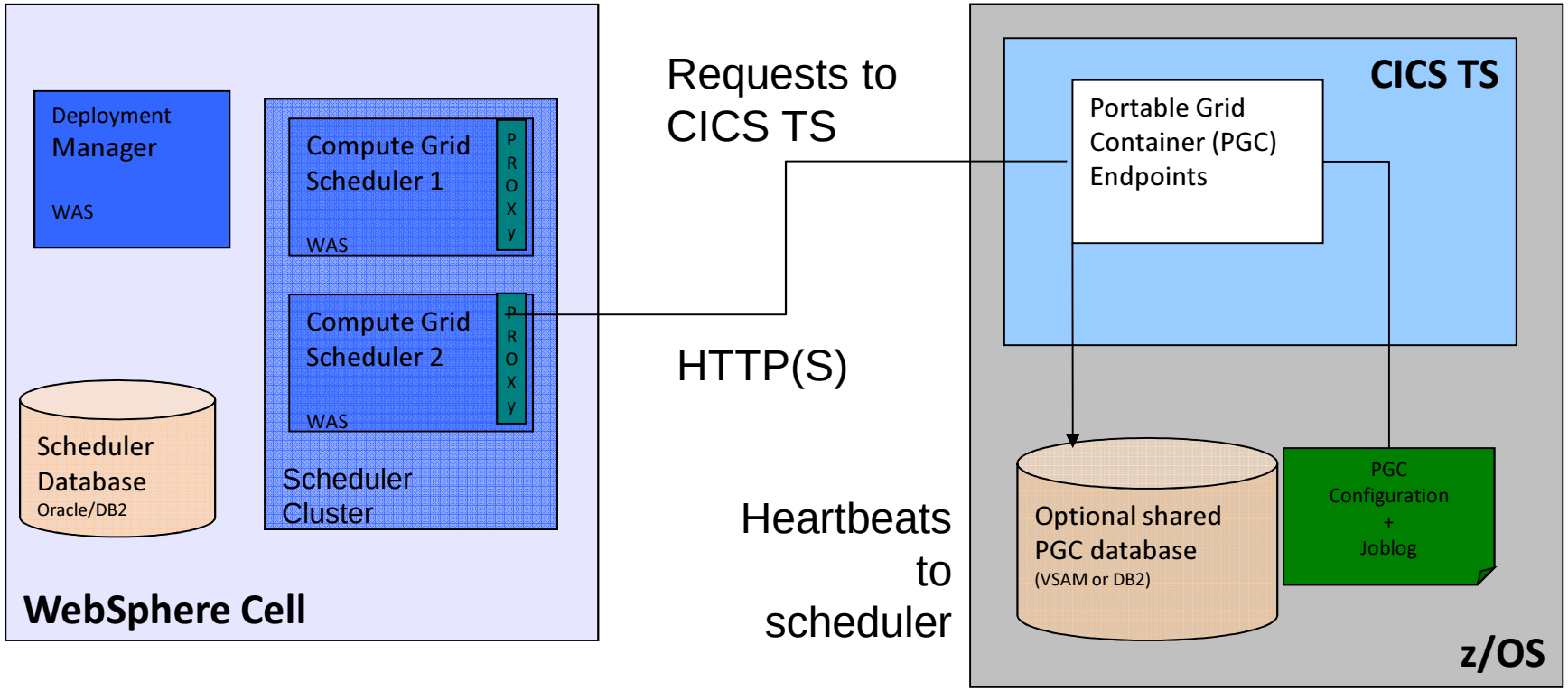




- A newly announced component integrated part of z/OS (see ENUS211-007)
- An option for running batch work; provides a managed environment for integration of Java and COBOL
- Consistent with IBM WebSphere based batch
  - A subset of the WebSphere programming model
  - Based upon WCG Batch Container, and the new WCGv8 COBOL invocation facility
- This is a simple “Java launcher” environment
  - Not a “hot running Java batch” environment.
  - No connection to WebSphere Batch scheduler
  - DB2 is the only supported resource manager currently supported
  - No support for COBOL invocation
  - Used typically for scenarios for extending traditional COBOL batch jobs with Java language steps

WebSphere XD Compute Grid and CICS with CICS SupportPac* CN11

Typically used for CICS batch scenarios that need to add Java



* Note: this is an ***unsupported*** SupportPac

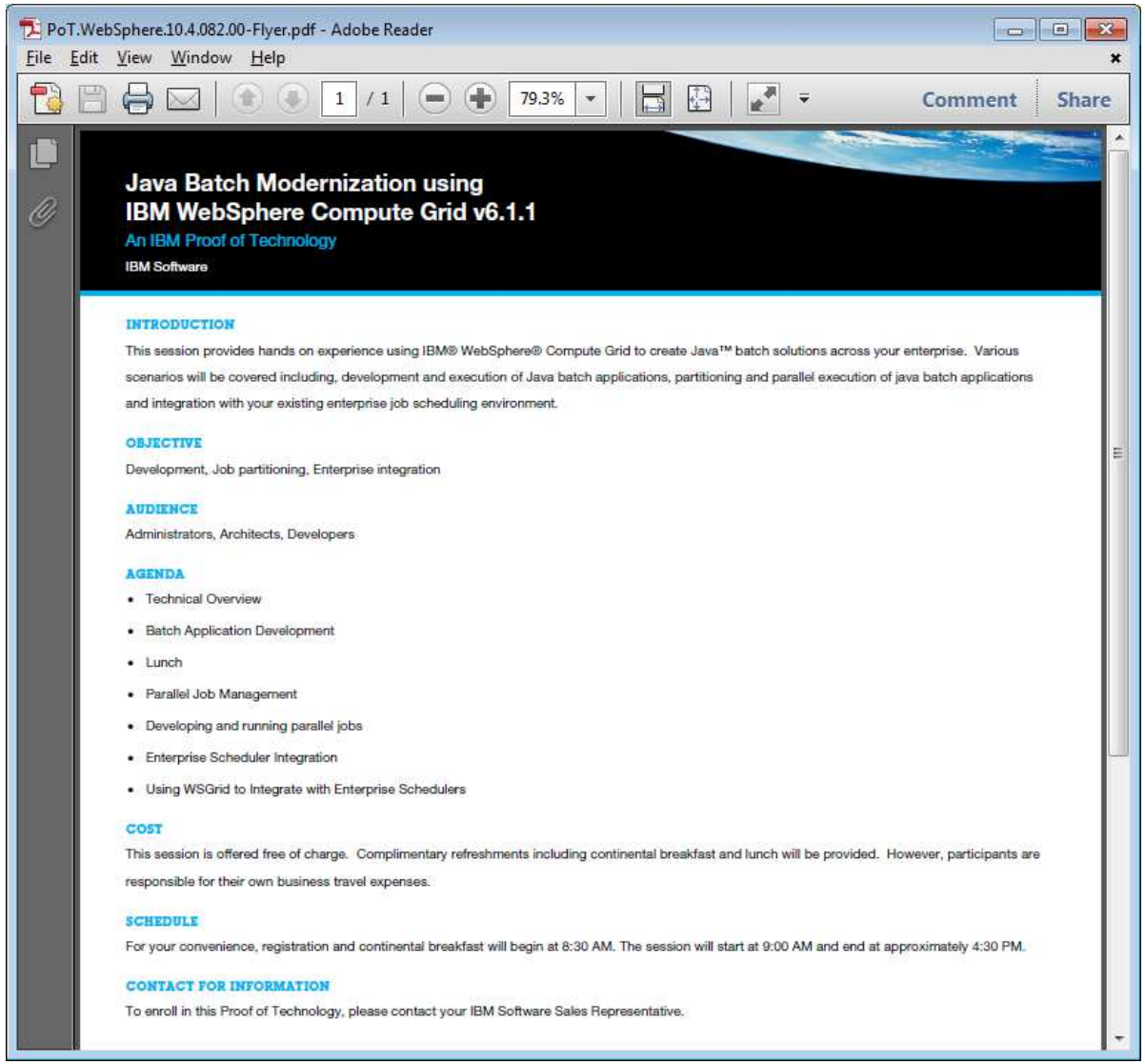
Java Batch Container offerings

	WCG or WebSphere Batch	CICS CN11 Batch Container	z/OS 1.13 Java Batch Container
JVM	Reused across multiple jobs	Reused across multiple jobs	Restarted with each job step
Programming model support (eg. checkpoints, BDSF..)	Supported	Supported	Supported
Scheduler support (Web interface, pause, WSGRID, etc)	Supported	Supported	Not Supported (eg. JES only)
Java API	Java SE and full EE	Java SE and JCICS	Java SE and subset EE
COBOL invocation	Tool-generated call stubs; inside WAS	CICS LINK mechanism	Hand written JNI code
Parallel Batch	Supported	Supported	Not Supported
IBM support status	Full	Not supported	Full

Summary

- Java (like COBOL, PL/I, etc.) is a perfectly viable programming language for batch processing with excellent price/performance today
- There are several “environments” for Java batch processing, each with its particular QOS
- The richer Java environment on z/OS is WAS-based
- WASv7 includes the free-of-charge the Feature Pack for Modern Batch for a basic WAS Java Batch environment
- WASv8.5 integrates the complete WCG offer natively into WAS
- The WebSphere Compute Grid on z/OS provides industrial-strength features on top of the WAS Feature Pack, such as….
  - WLM and SMF exploitation
  - All the performance and robustness advantages of data co-location
  - Integration with enterprise schedulers (eg. TWS, etc.) (MQ or SIBus)
  - A heterogeneous execution environment
  - Unlimited scalability with a parallel execution framework

WCG Proof-Of-Technology



Backup

Bibliography

- Batch Modernization on z/OS (SG24-7779) http://www.redbooks.ibm.com/redbooks/pdfs/sg247779.pdf
- BPXBATCH documentation
  http://publib.boulder.ibm.com/infocenter/zos/v1r9/topic/com.ibm.zos.r9.bpxa500/bpxza5802428.htm#wq3034
- JZOS Batch Launcher and Toolkit – Installation and User's Guide (SA23-2245)
- Java Stand-alone Applications on z/OS (Vols 1, 2) (SG24-7177, SG24-7291)
- Java enabled Batch Compute Grid Whitepaper (WSW14113USEN)
- WebSphere Extended Deployment Information Center
  http://publib.boulder.ibm.com/infocenter/wxdinfo/v6r1/index.jsp
- IBM WebSphere Application Server V7 Feature Pack for Modern Batch
  http://www.ibm.com/software/webservers/appserv/was/featurepacks/modernbatch/
- Batch Processing with WebSphere Compute Grid: Delivering Business Value to the Enterprise
  http://www.redbooks.ibm.com/Redbooks.nsf/RedpieceAbstracts/redp4566.html?Open
- WebSphere Extended Deployment Compute Grid ideal for handling mission-critical batch workloads
  http://www.ibm.com/developerworks/websphere/techjournal/0804_antani/0804_antani.html
- Washington Systems Center documents on WAS Feature Pack and Compute Grid
  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101783
- YouTube introductions to WebSphere Batch (4 parts) – Part 1
  http://www.youtube.com/watch?v=fczMsZBlJko&feature=youtu.be
- Java Batch Programming with XD Compute Grid
  http://www.ibm.com/developerworks/websphere/techjournal/0801_vignola/0801_vignola.html
- WebSphere Compute Grid Frequently Asked Questions
  http://www.ibm.com/developerworks/forums/thread.jspa?threadID=228441&tstart=0
- Development Tooling Summary for XD Compute Grid
  http://www.ibm.com/developerworks/forums/thread.jspa?threadID=190624
- Compute Grid Discussion forum
  http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1240