Batch Modernisation with WebSphere Compute Grid

Steve Foley
IBM Software Group
steve_foley@uk.ibm.com

WebSphere Software for System z

Important Disclaimer

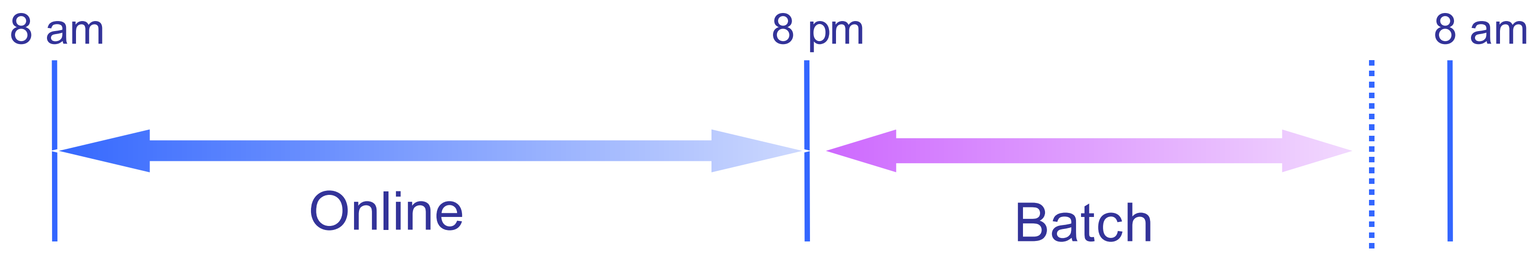
Agenda

- Batch Computing
  - Drivers for modernisation
- WebSphere Compute Grid Overview
  - Batch Platform
  - Application Development & Tooling
- WebSphere Compute Grid on z/OS
  - Benefits
  - Customer example
- Summary

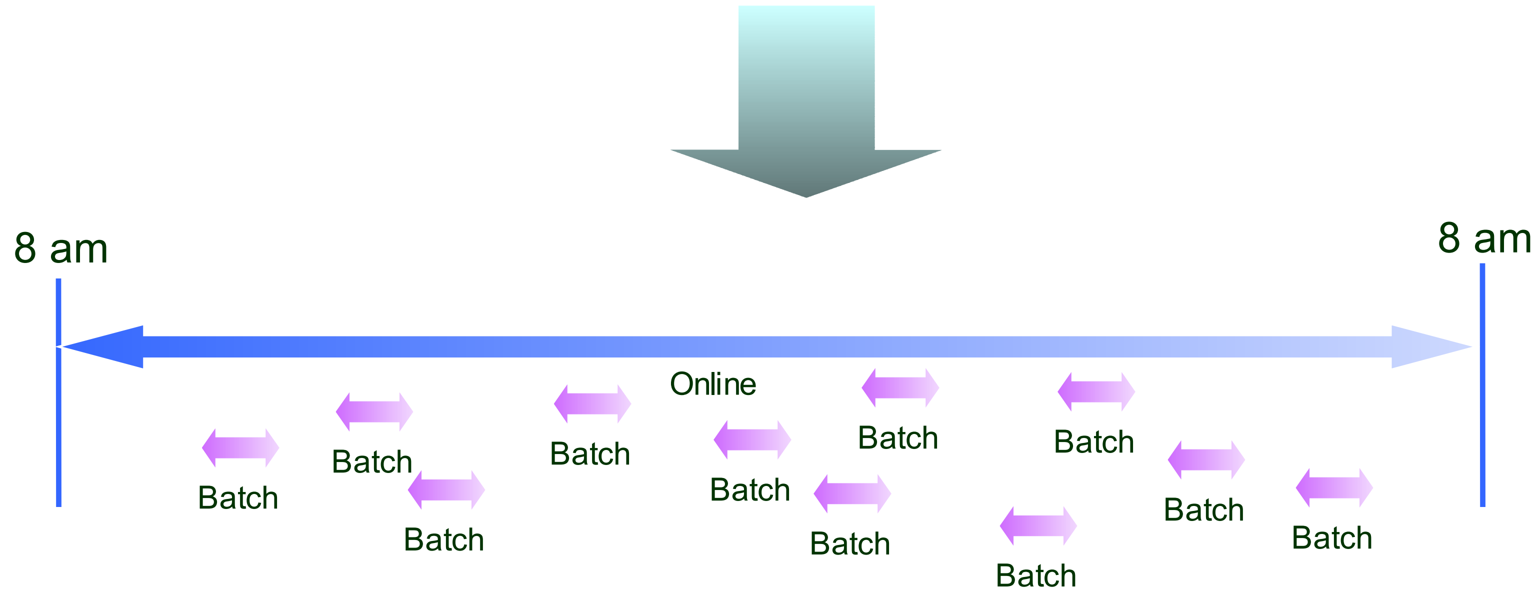
Agenda

- Batch Computing
  - Drivers for modernisation
- WebSphere Compute Grid Overview
  - Batch Platform
  - Application Development & Tooling
- WebSphere Compute Grid on z/OS
  - Benefits
  - Customer example
- Summary

24x7 Batch and OLTP



Current Batch Processing Technique



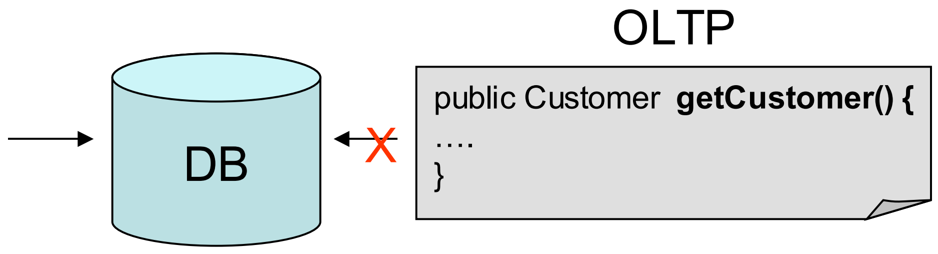
Modern Batch Processing Technique

OLTP and Batch Interleave

```
public void doBatch() {
Session session = sessionFactory.openSession();
Transaction tx = session.beginTransaction();
for ( int i=0; i<100000; i++ ) {
    Customer customer = new Customer(.....);
    Cart cart = new Cart(...);
    customer.setCart(cart) // needs to be persisted as well
    session.save(customer);
    if ( i % 20 == 0 ) { //20, same as the JDBC batch size
        //flush a batch of inserts and release memory:
        session.flush();
        session.clear();
    }
}
tx.commit();
session.close();
}
```

Source: some Hibernate Batch website

BATCH



```
public Customer  getCustomer() {
….
}
```

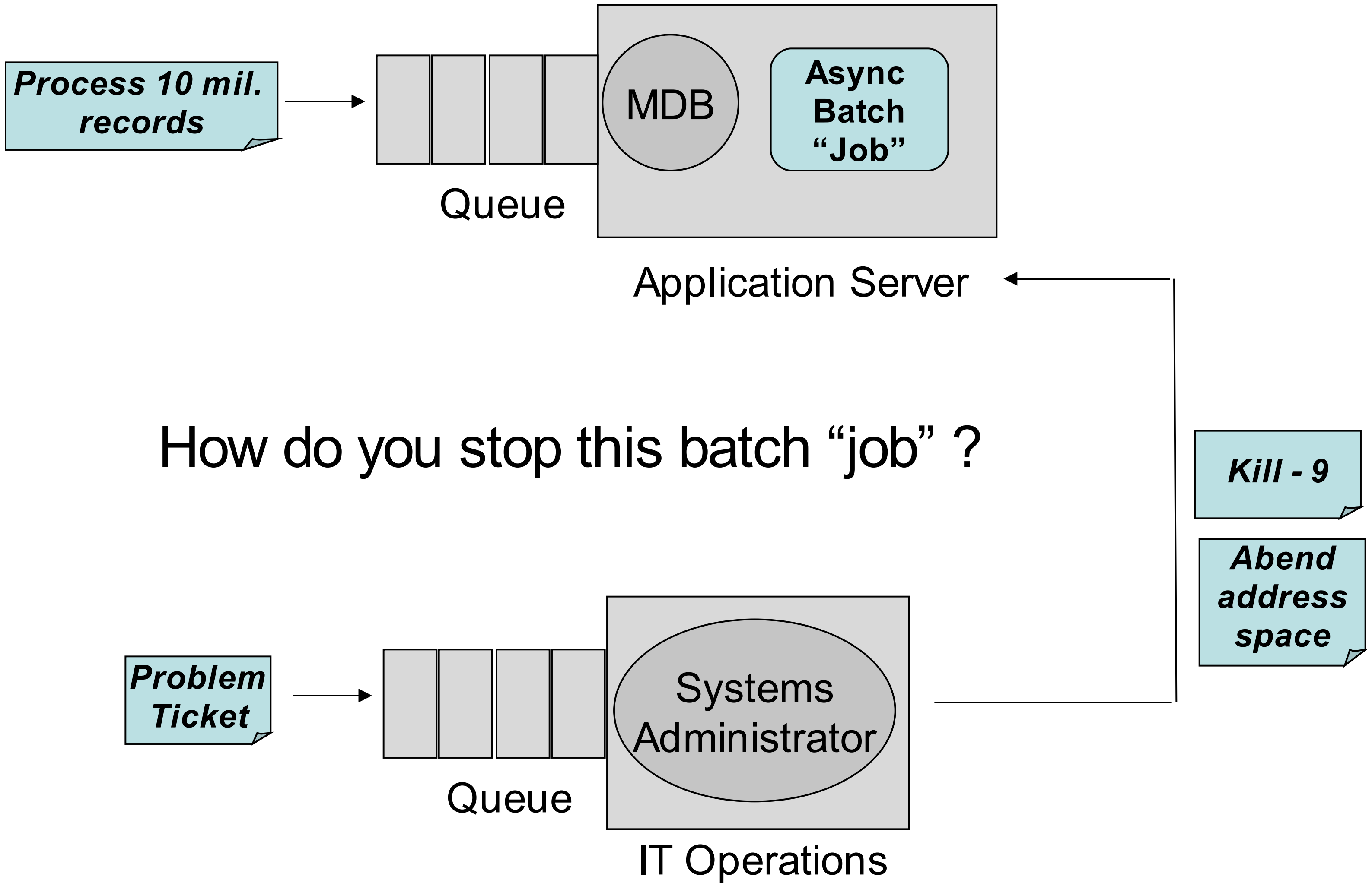
-Batch application's hold on DB locks can adversely impact OLTP workloads

-OLTP Service Level Agreements can be breached

-How do you manage this?

-WLM will make the problem worse!

Operational Control in “Maverick” Batch



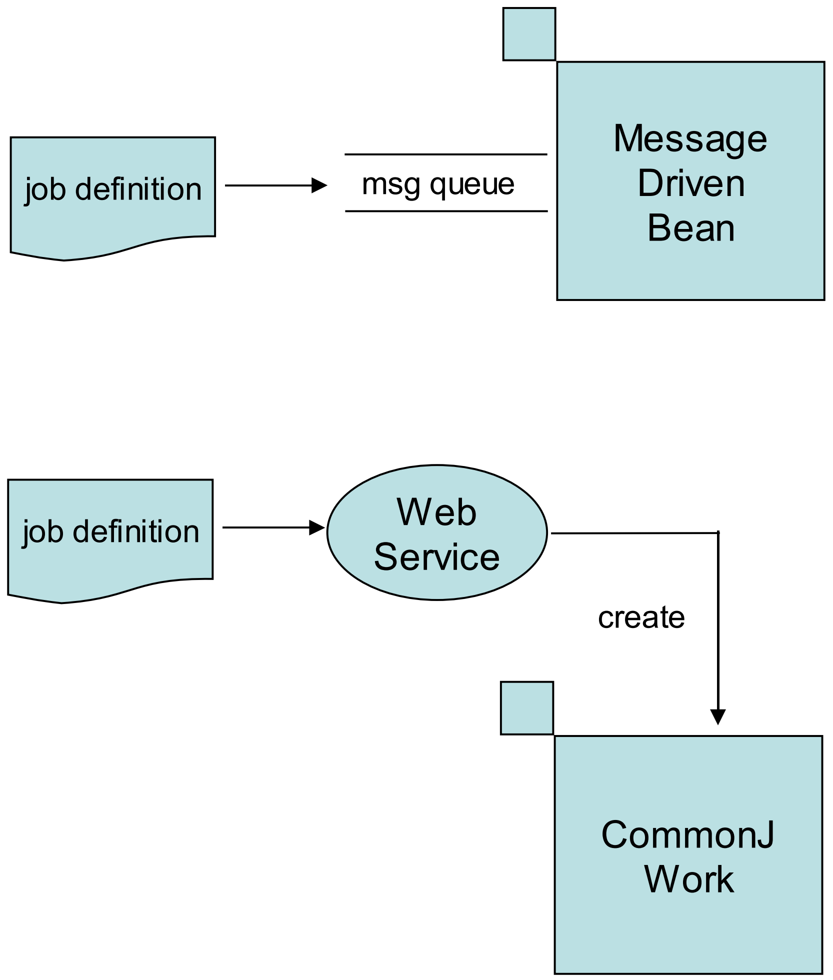
How do you stop this batch “job” ?

The “Maverick” Batch Environment

- Roll Your Own (RYO)
- Seems easy – even tempting ☺
- Message-driven Beans or
- CommonJ Work Objects or …

But …

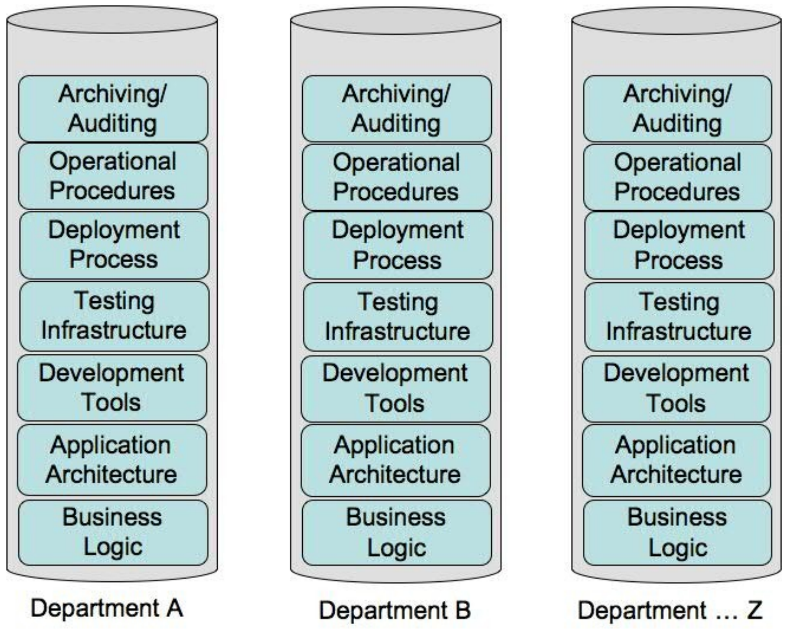
- No job definition language
- No batch programming model
- **No checkpoint/restart**
- No batch development tools
- **No operational commands**
- **No OLTP/batch interleave**
- No logging
- **No job usage accounting**
- **No monitoring**
- No job console
- No enterprise scheduler integration
- **No visibility to WLM**
- No Workload throttling/pacing/piping
- …



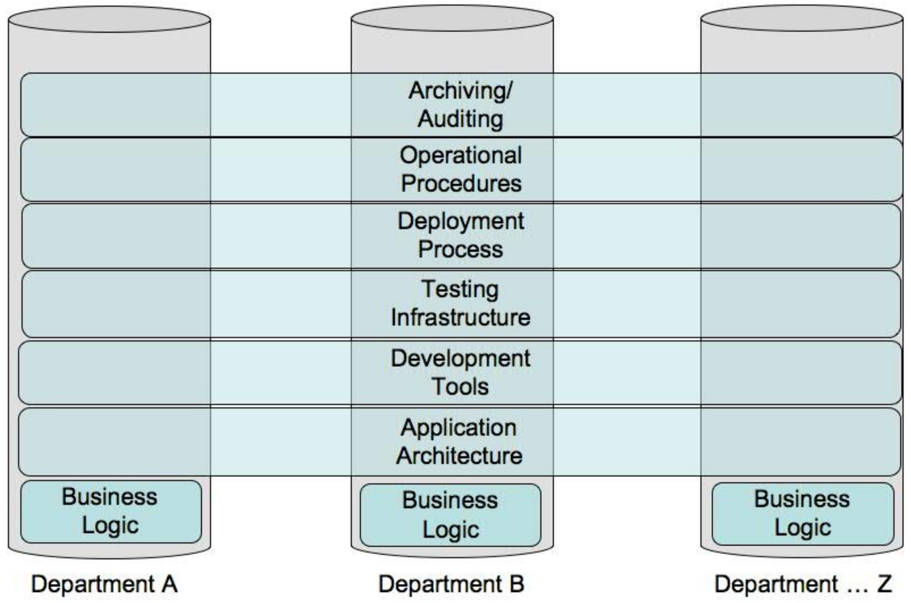
Business Benefit

… Cut IT development, operations, and maintenance costs by pursuing the "Unified Batch Architecture" strategy with Compute Grid

- Today: Batch processing systems exist in silos



Tomorrow: Common Infrastructure for hosting and executing batch applications



Batch Modernization... Why Java?

- Modern language
  - Virtualized
  - Portable
  - Functionally rich
  - zAAP offload (z/OS)
- Standards
  - Programming model
  - Component model
  - J2SE/J2EE
- Skills proliferation
- Choice of Tools and Vendors

Comparing Batch Computing Characteristics

	Traditional Batch Computing	Modern Batch Computing
Platforms	zOS and iSeries some primitive Unix use	Multi platform (with platform unique advantages for differentiated value)
Languages	COBOL, PL/I	Modern languages – Java
Programming Patterns	Outer loop – read input stream Inner loop – checkpoint x records	Container-managed checkpoint processing Variable checkpoint intervals based on competing workloads, contention resolution and deadline SLAs
Concurrency	Batch window separate from OLTP Inherent contention with DB	24 x 7 concurrency Executing micro-batches
Scheduling	TWS, start-time/dependency scheduling	Dead-line / dependency scheduling
Job Flow Control	JCL	xJCL, JSDL
Resource Efficiency	Better than OLTP – transactional commit processing aligned to checkpoint intervals Table-level locking is typical Constrained by hard coded checkpoint size	Variable-length, container managed checkpoint processing can propagate to down-stream resource dependencies – bulk resource bandwidth reservation and allocation / deallocation control
Design Analysis	None	Service Oriented Architecture Derived from business process design requirements – integrated into SOMA or similar methods
Simulation, Tooling	Minimal	Integrated into BPM and IDE tooling
Management	Platform specific	Integrated with ISM and SOA management tools Virtualized and Grid enabled

Agenda

- Batch Computing
  - Drivers for modernisation
- WebSphere Compute Grid Overview
  - Batch Platform
  - Application Development & Tooling
- WebSphere Compute Grid on z/OS
  - Benefits
  - Customer example
- Summary

The Product Formerly Known as *WebSphere Extended Deployment (XD)*

XD contains 3 components, available as a single, integrated package or 3 individual components

Compute Grid

- *Java Batch*
- *Checkpoint / Restart*
- *Parallel job execution*
- *Enterprise Scheduler Integration*
- *z/OS Integration Patterns*

Virtual Enterprise

- *On-Demand Router*
- *Extended Manageability*
- *Application Editions*
- *Health Management*
- *Runtime Visualization*
- *Virtualization*

eXtreme Scale

- *Distributed Caching*
- *Partitioning Facility*
- *In-memory Databases*

WebSphere XD Compute Grid summary

- Leverages J2EE Application Servers (WebSphere today… more tomorrow)
  - **Transactions**
  - **Security**
  - **high availability including dynamic servants**
  - **Leverages the inherent WAS QoS**
  - **Connection Pooling**
  - **Thread Pooling**

- Platform for executing transactional java batch applications
  - Checkpoint/Restart
  - Batch Data Stream Management
  - Parallel Job Execution
  - Operational Control
  - External Scheduler Integration
  - SMF Records for Batch
  - zWLM Integration

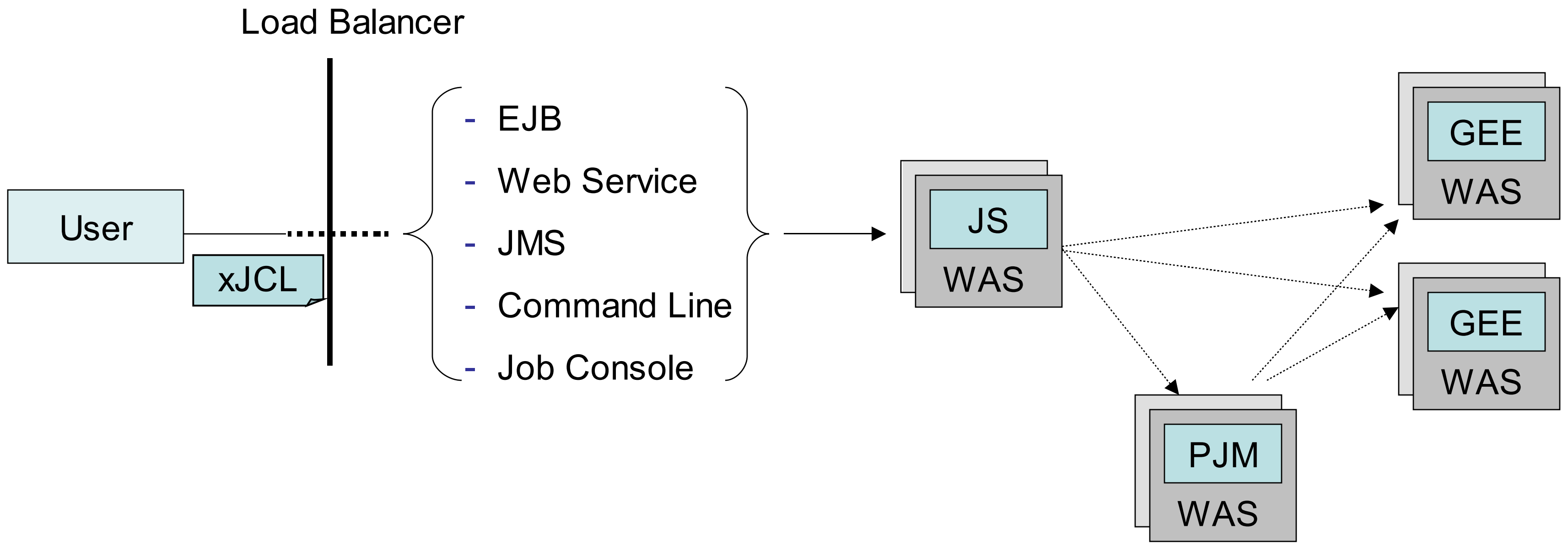
XD Compute Grid Components

- Job Scheduler (**JS**)
  - The job entry point to XD Compute grid
  - Job life-cycle management (Submit, Stop, Cancel, etc) and monitoring
  - Dispatches workload to either the PJM or GEE
  - Hosts the Job Management Console (JMC)

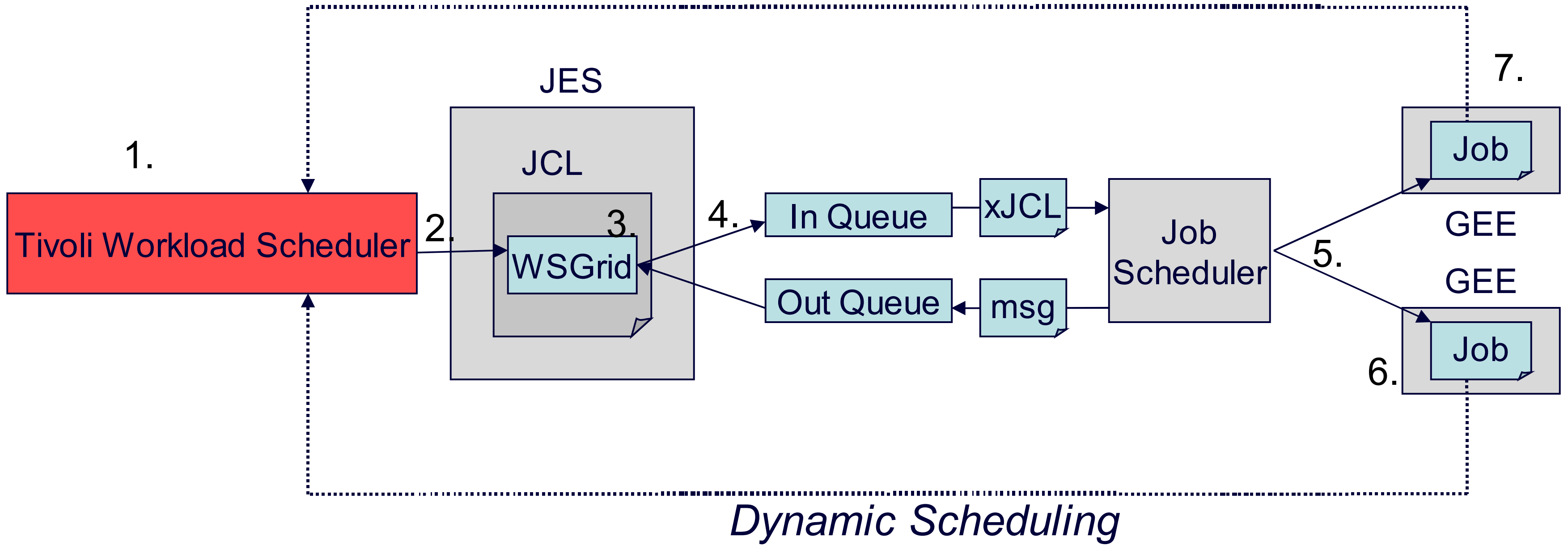
- Parallel Job Manager (**PJM**)-
  - Breaks large batch jobs into smaller partitions for parallel execution
  - Provides job life-cycle management (Submit, Stop, Cancel, Restart) for the single logical job and each of its partitions
  - Is *not* a required component in compute grid

- Grid Endpoints (**GEE**)
  - Executes the actual business logic of the batch job

XD Compute Grid Components



Enterprise Scheduler Integration



- *enterprise scheduler for operational control*
- *Jobs and commands are submitted from WSGRID*
- *Jobs can dynamically schedule ES via its EJB interface*

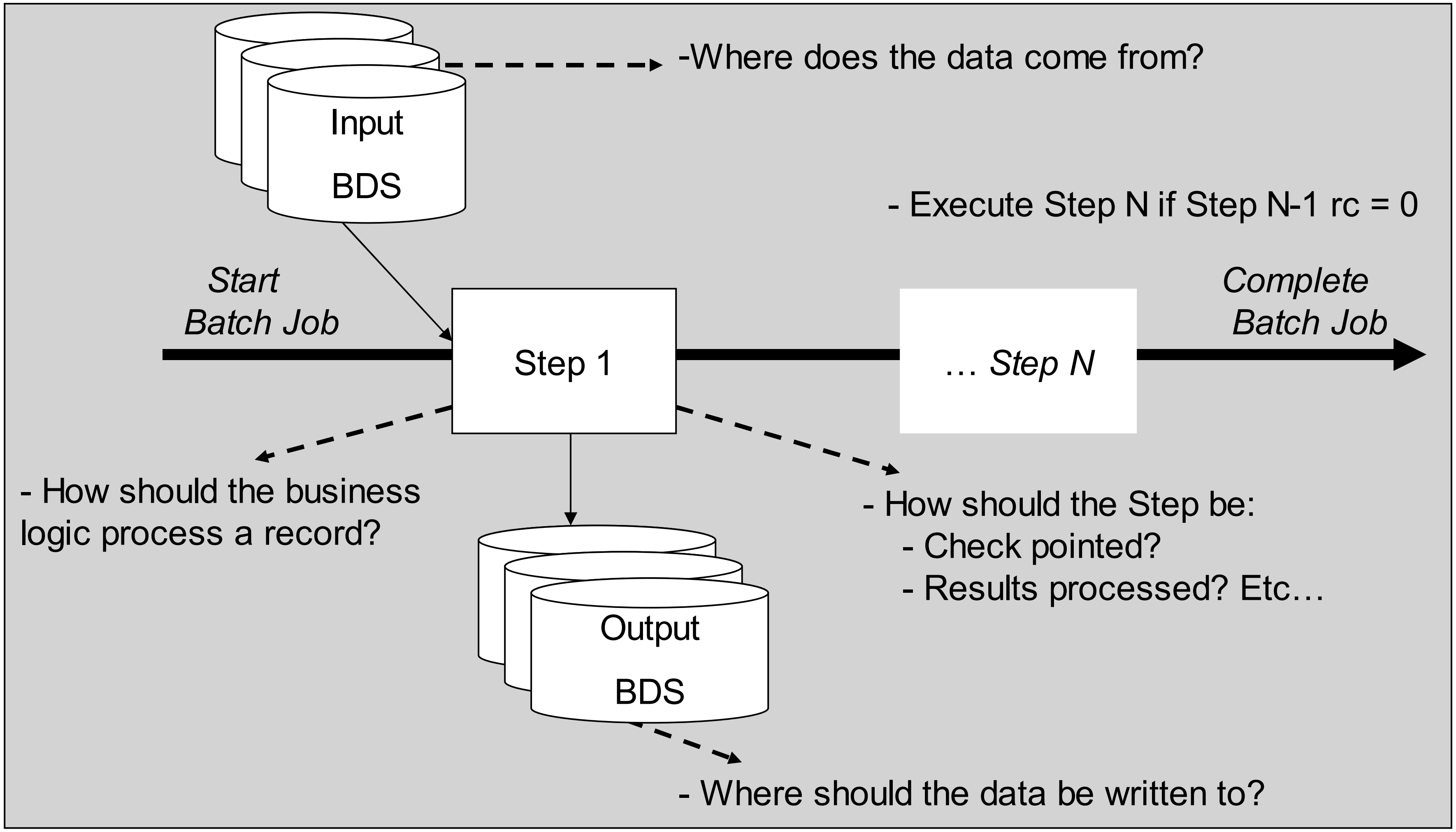
Enterprise Features summary…

- Check Point Restart
  - Container-Managed ***Checkpoint Strategies***
    - Keep track of the current input and output positions on behalf of the batch step
    - Store these values as part of the same global transaction as the business logic
    - Provide flexible options: Time-based, Record-based, Custom algorithms
  - Container-Managed ***Restart Capabilities***
    - Seek to the correct positions in the input and output streams
    - Restart should be ***transparent*** to the application
  - Dynamically adjust the checkpoint strategies based on Workload Management metrics, OLTP load, and application priorities
- Integrated Operation control
  - Provide an operational infrastructure for starting/stopping/canceling/restarting/etc batch jobs
  - Integrate with existing enterprise schedulers such as Tivoli Workload Scheduler
  - Provide log management and integration with archiving and auditing systems
  - Provide resource usage monitoring
  - Integrate with existing security and disaster recovery procedures
- High Availability
  - Clustered Job Scheduler
  - Clustered Endpoints
- Disaster recovery through multisite topology

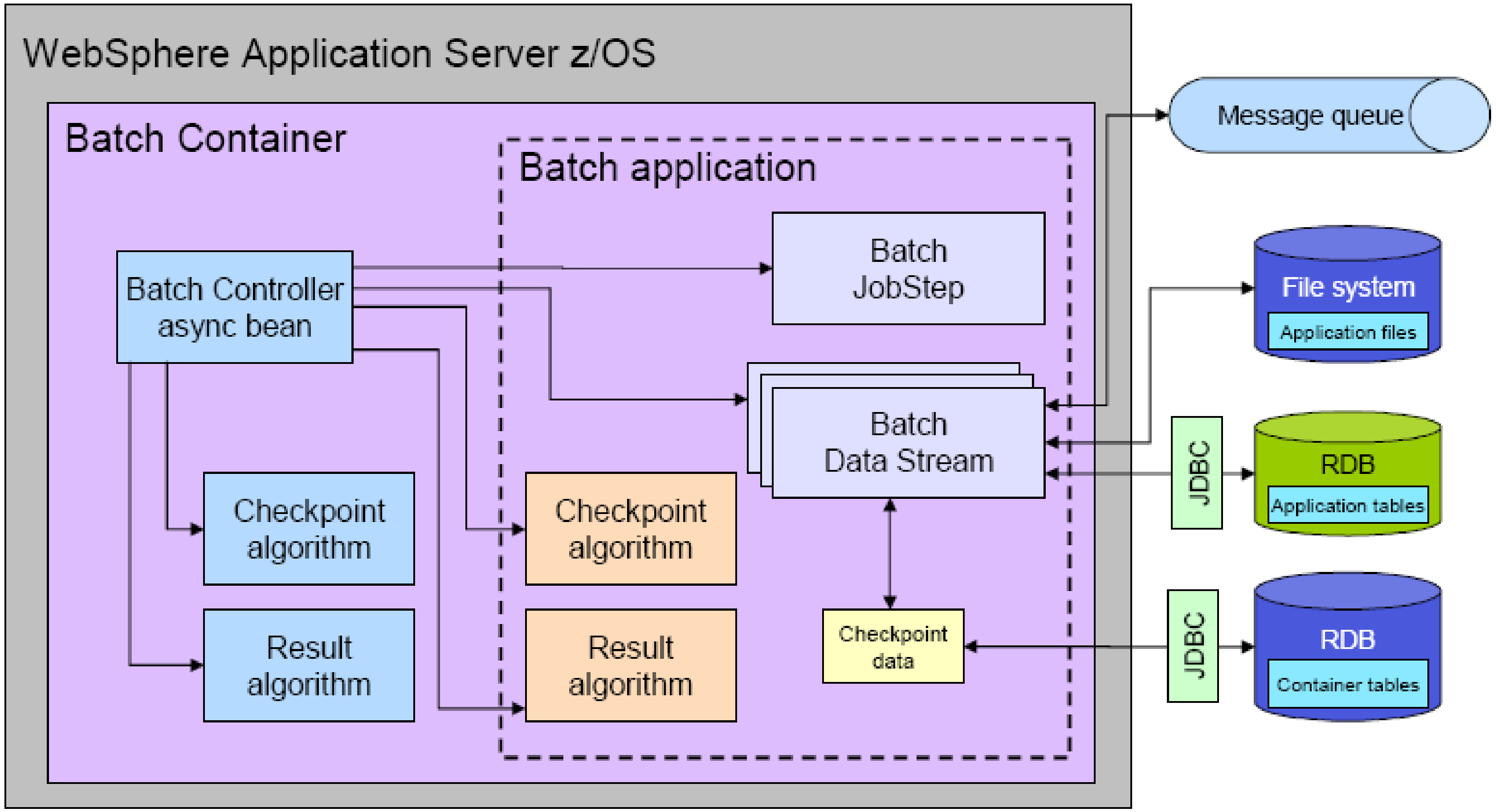
Agenda

- Batch Computing
  - Drivers for modernisation
- WebSphere Compute Grid Overview
  - Batch Platform
  - Application Development & Tooling
- WebSphere Compute Grid on z/OS
  - Benefits
  - Customer example
- Summary

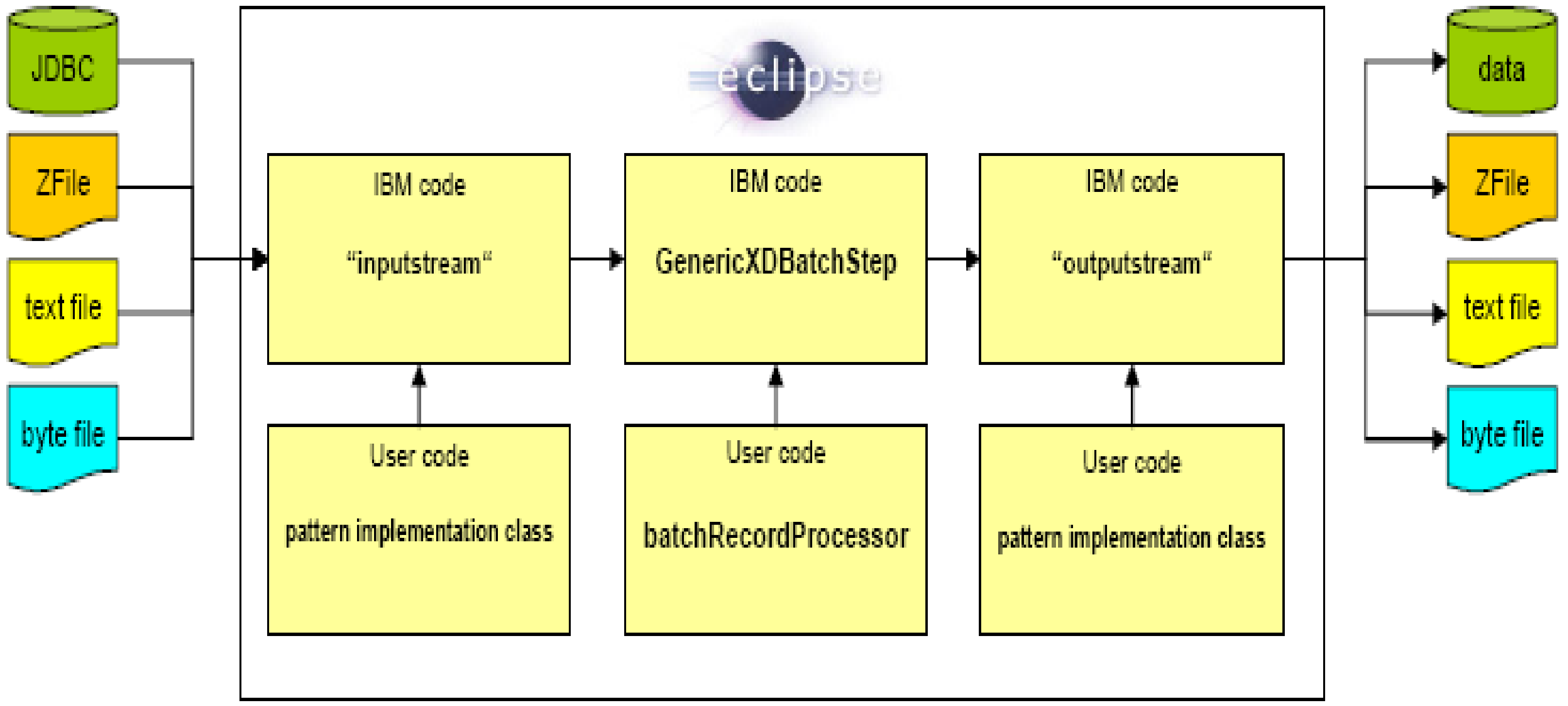
Components of a Batch Application


Input
BDS
-Where does the data come from?
- Execute Step N if Step N-1 rc = 0
Start
Batch Job
Step 1
… Step N
Complete
Batch Job
- How should the business logic process a record?
- How should the Step be:
- Check pointed?
- Results processed? Etc…
Output
BDS
- Where should the data be written to?

Programming model



BDS Framework



Development Tooling Story for WebSphere XD Compute Grid

- 1. The **Batch Datastream (BDS) Framework**. This is a development toolkit that implements the XD batch programming model for common use-cases: e.g. accessing MVS Datasets, Databases, files, JDBC Batching, and provides all of the restart logic specific to XD Batch programming model. The following post goes into more details.

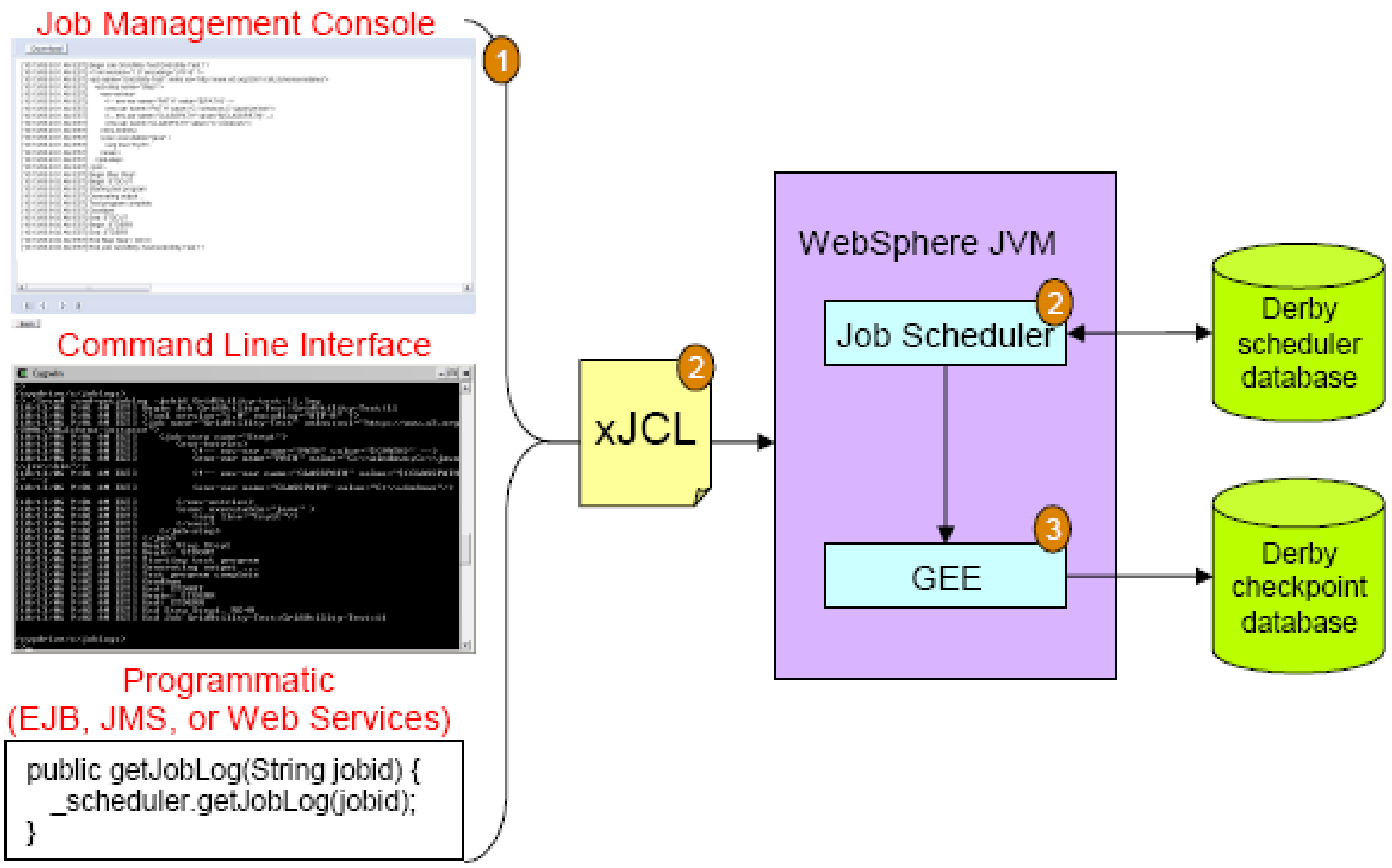
  2. a **Pojo-based application development model**. As of XD 6.1, you only have to write Pojo-based business logic. Tooling executed during the deployment process will generate the necessary Compute Grid artifacts to run your application. The following developerworks article goes into more details:
  Intro to Batch Programming with WebSphere XD Compute Grid

  3. The **Batch Simulator**. A light-weight, non-J2EE batch runtime that exercises the Compute Grid programming model. This runs in any standard Java development environment like Eclipse, and facilitates simpler application development since you're only dealing with Pojo's and no middleware runtime. The Batch Simulator is really for developing and testing your business logic. Once your business logic is sound, you would execute function tests, system tests, and then deploy to production. You can download this from batch simulator download

  4. The **Batch Packager**. This utility generates the necessary artifacts for deploying your Pojo-based business logic into the Compute Grid runtime. The packager is a script that can be integrated into the deployment process of your application. It can also be run independently of the WebSphere runtime, so you don't need any heavy-weight installs in your development environment.

  5. The **Unit-test environment (UTE)**. The UTE package is described in the following post. The UTE runs your batch application in a single WebSphere server that has the Compute Grid runtime installed. It's important to function-test your applications in the UTE to ensure that it behaves as expected when transactions are applied.

Compute Grid UTE



```
public getJobLog(String jobid) {
   _scheduler.getJobLog(jobid);
}
```

End-to-end Development tooling

-Customer develops business service POJO's

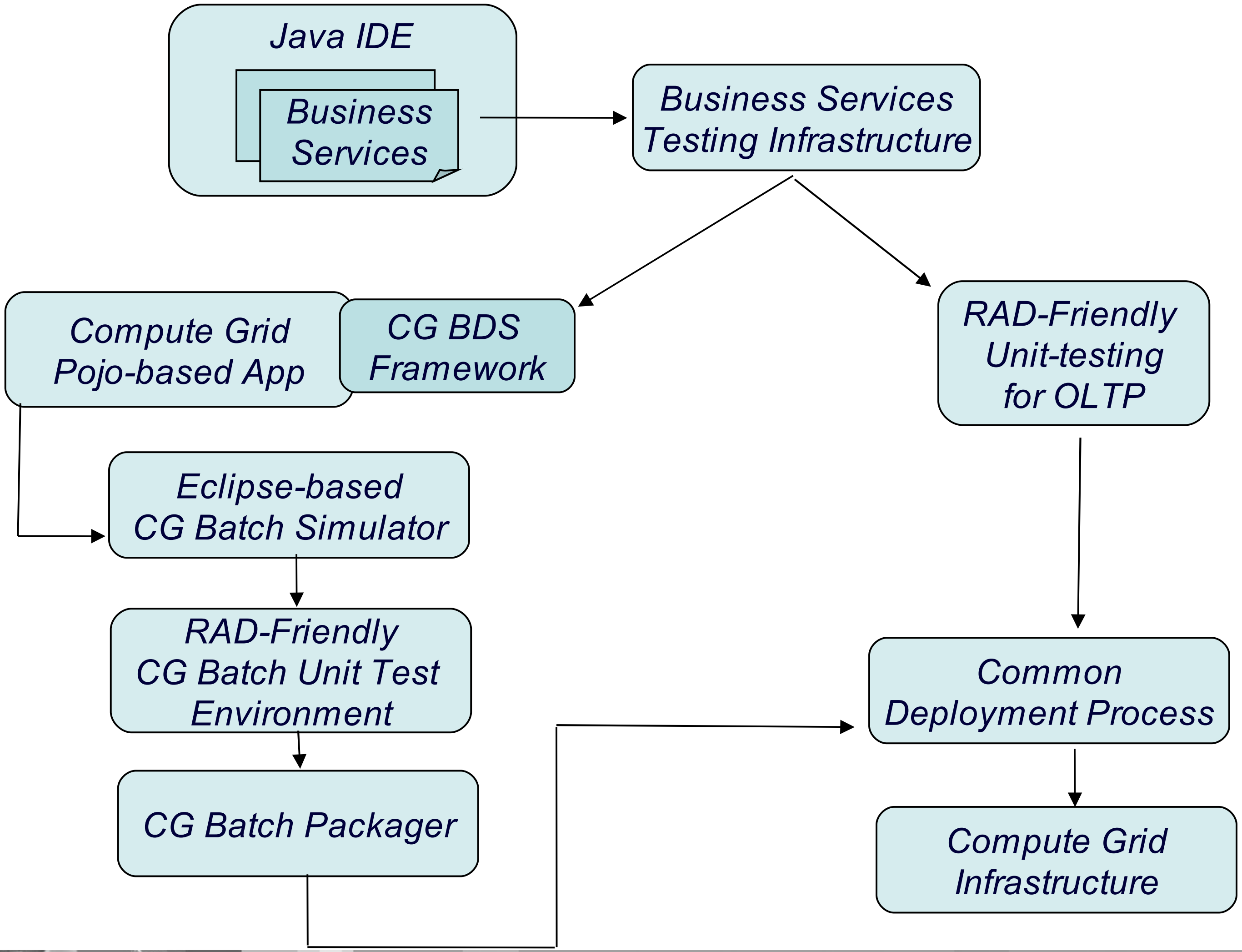
-Applications are assembled via IOC Container

-XD BDS Framework acts as bridge between job business logic and XD Compute Grid programming model

-XD Batch Simulator for development

-XD Batch Unit test environment for unit testing

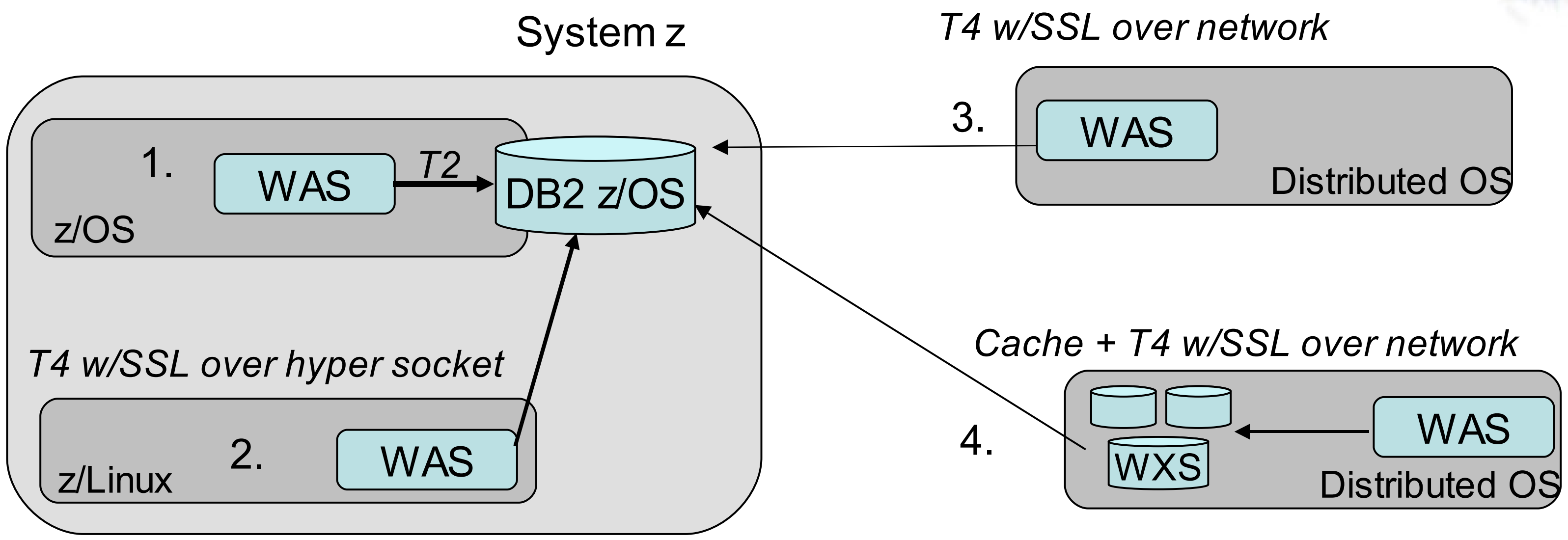
-XD batch packager for .ear creation


Java IDE
Business Services
Business Services Testing Infrastructure
Compute Grid Pojo-based App
CG BDS Framework
RAD-Friendly Unit-testing for OLTP
Eclipse-based CG Batch Simulator
RAD-Friendly CG Batch Unit Test Environment
CG Batch Packager
Common Deployment Process
Compute Grid Infrastructure

Agenda

- Batch Computing
  - Drivers for modernisation
- WebSphere Compute Grid Overview
  - Batch Platform
  - Application Development & Tooling
- WebSphere Compute Grid on z/OS
  - Benefits
  - Customer example
- Summary

Proximity of Data - Options



- WAS z/OS using optimized mem-to-mem JDBC Type-2 Driver
- WAS z/Linux using JDBC Type-4 driver and SSL over optimized z network stack
- WAS distributed (unix/linux/windows/etc) using JDBC Type-4 driver and SSL over traditional network stack
- WAS distributed coupled with WebSphere eXtreme Scale cache

If the data is on z/OS, the batch application should run on z/OS.

XD Compute Grid and z/OS Integration

XD Compute Grid leverages and integrates with native z/OS workload management (WLM) to enhance job execution & management

SMF accounting records for J2EE batch jobs

- SMF 120 (J2EE) records tailored to jobs
- Record includes: job id, user, CPU time, job state, server, etc.

Dynamic Servants for J2EE batch job dispatch

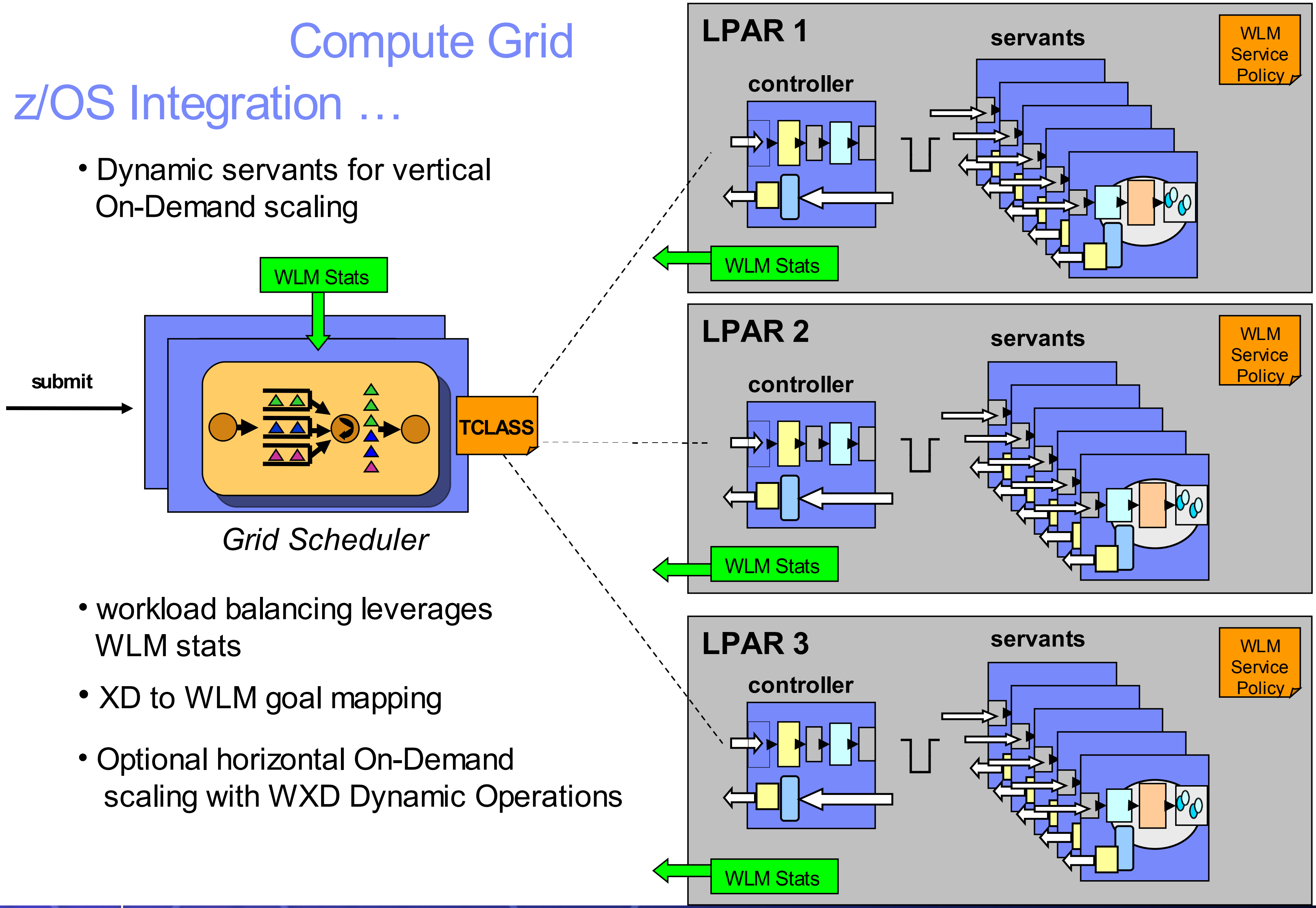
- Exploit WLM to start new servants to execute J2EE batch jobs on demand

Service policy classification and delegation

- Leverage XD classification to select z/OS service class by propagating transaction class from scheduler to z/OS app server for job registration with WLM

Compute Grid
z/OS Integration …

- Dynamic servants for vertical On-Demand scaling



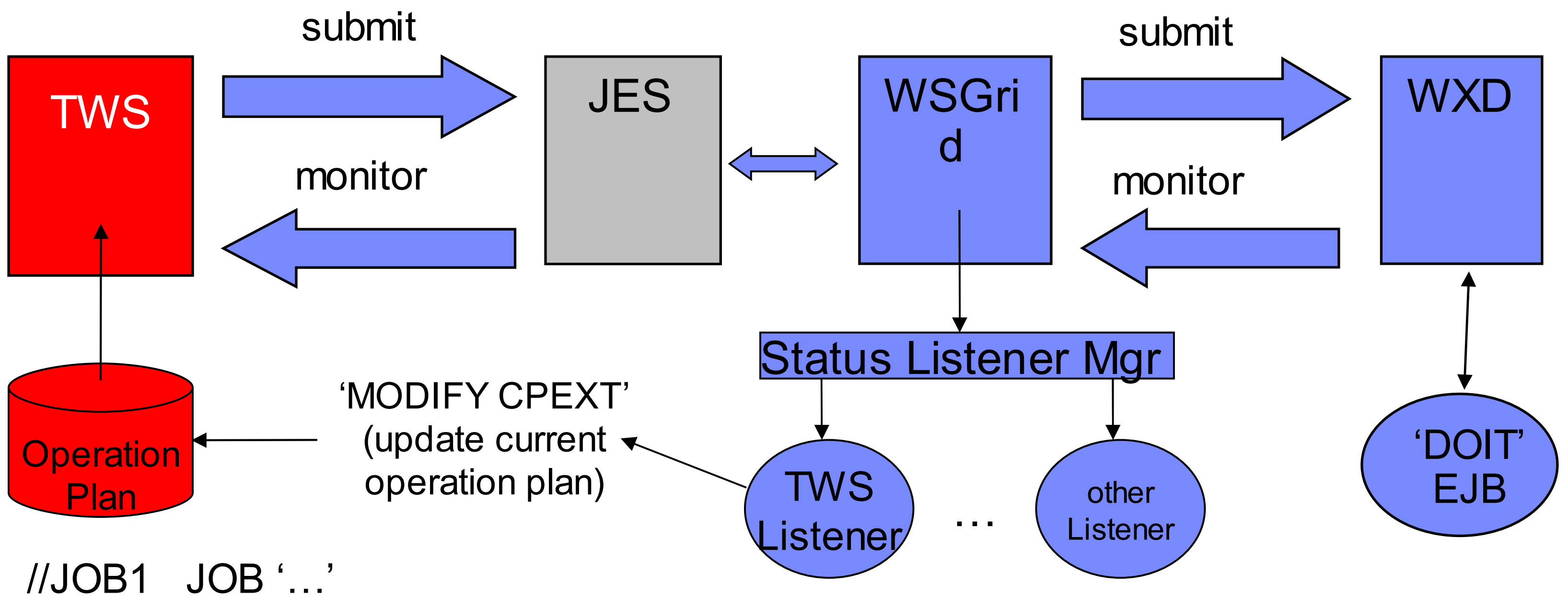
- workload balancing leverages WLM stats
- XD to WLM goal mapping
- Optional horizontal On-Demand scaling with WXD Dynamic Operations

Execution Environment – z/OS WLM Integration

- WAS uses WLM to control the number of Servant Regions
- Control Regions are MVS started task
- Servant Regions are started automatically by WLM an a as-needed basis
- WLM queues the user work from the Controller to the Servant region according to service class
- WLM queuing places user requests in a servant based on same service class
- WLM ensures that all user requests in a given servant has been assigned to the same service class
- A Servant running no work can run work assigned to any service class
- WLM and WAS Worker thread : WLM dispatch work as long as it has worker threads
- Behavior of WAS Worker Threads (ORB workload profile)
  - ISOLATE : number of threads is 1. Servants are restricted to a single application thread
  - IOBOUND : number of threads is 3 * Number of CPUs)
  - CPUBOUND : number of threads is the Number of CPUs)
  - LONGWAIT : number of threads is 40
- XD service policies contain one or more transaction class definition
- XD service policies create the goal, while the job transaction class connects the job to the goal
- XD service policy transaction class is propagated to the Compute Grid Execution Environment
- Transaction class is assigned to a job during by the Scheduler during dispatch/classification phase
- When a job dispatch reaches GEE the Tclass is extracted from the HTTP request
- Tclass is mapped to WLM service class. An enclave is created.
- XD Service policies are not automatically defined in the z/OS WLM.

External Scheduler Integration

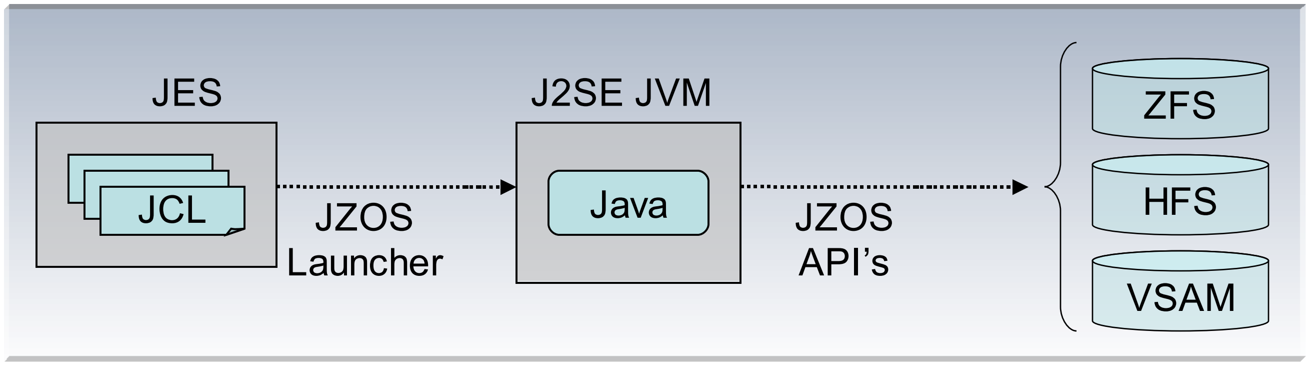
XD v6.1 job control by external workload scheduler (e.g. TWS, Control-M, etc)



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

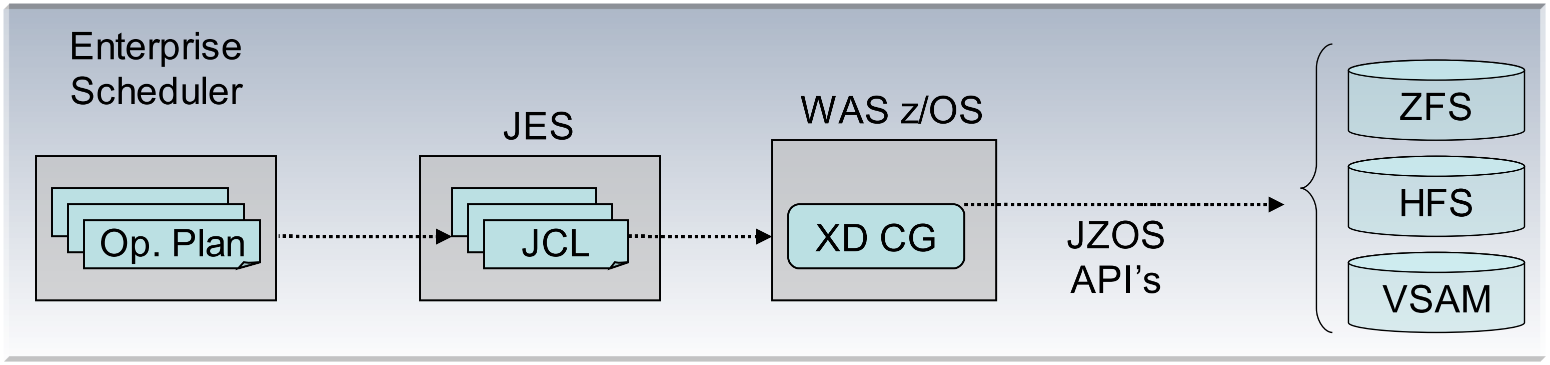
```
<job name="JOB1" …
<job-step name="STEP2">
<jndi-name>"ejb/DOIT"</jndi-name>
…
```

Positioning XD Compute Grid - Role of JZOS



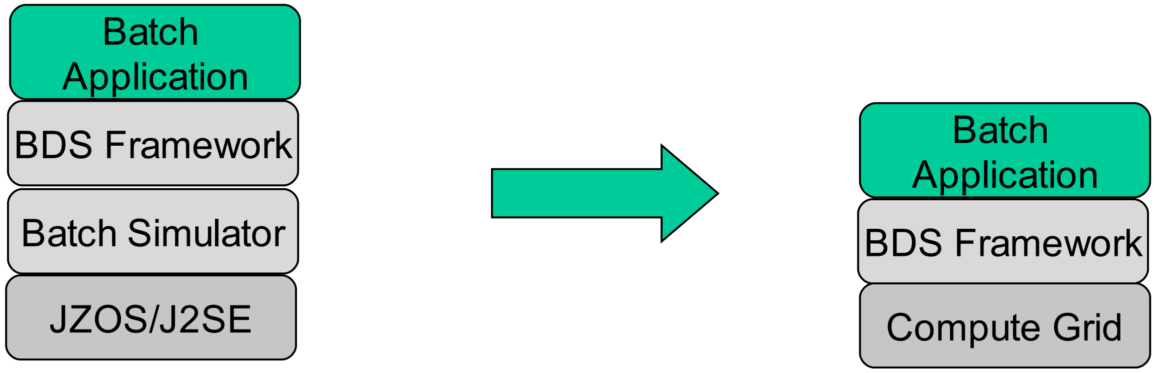
- JZos delivers 2 technologies:
  1. ***JZOS Launcher-*** seamless way to initialize a ***J2SE*** runtime from JCL
  2. ***JZOS API's-*** set of library functions for accessing traditional z/OS resources (MVS datasets, VSAM files, etc) from Java
- ***JZOS launcher*** not efficient for 1000's of batch jobs to be run within a batch window
  - J2SE JVM has no:
    - security, transaction, or connection management
    - checkpoint or restart facility for batch jobs
    - inherent high availability, or other WAS z/OS qualities of service
  - JVM is not persistent or reusable.
  - ***Great to get started…***

Positioning XD Compute Grid- Role of JZOS



- XD Compute Grid is ***built on WebSphere z/OS***
  - leverages QoS and services provided by the WAS z/OS runtime *(security, transaction, connection management; thread pooling; HA, etc)*
  - Runs within a persistent, reusable JVM and Execution Container
- JZOS Api's can be leveraged from XD CG applications
- JZOS Api's provide ***a strong integration point*** for Java and traditional z/OS

Grow into Compute Grid

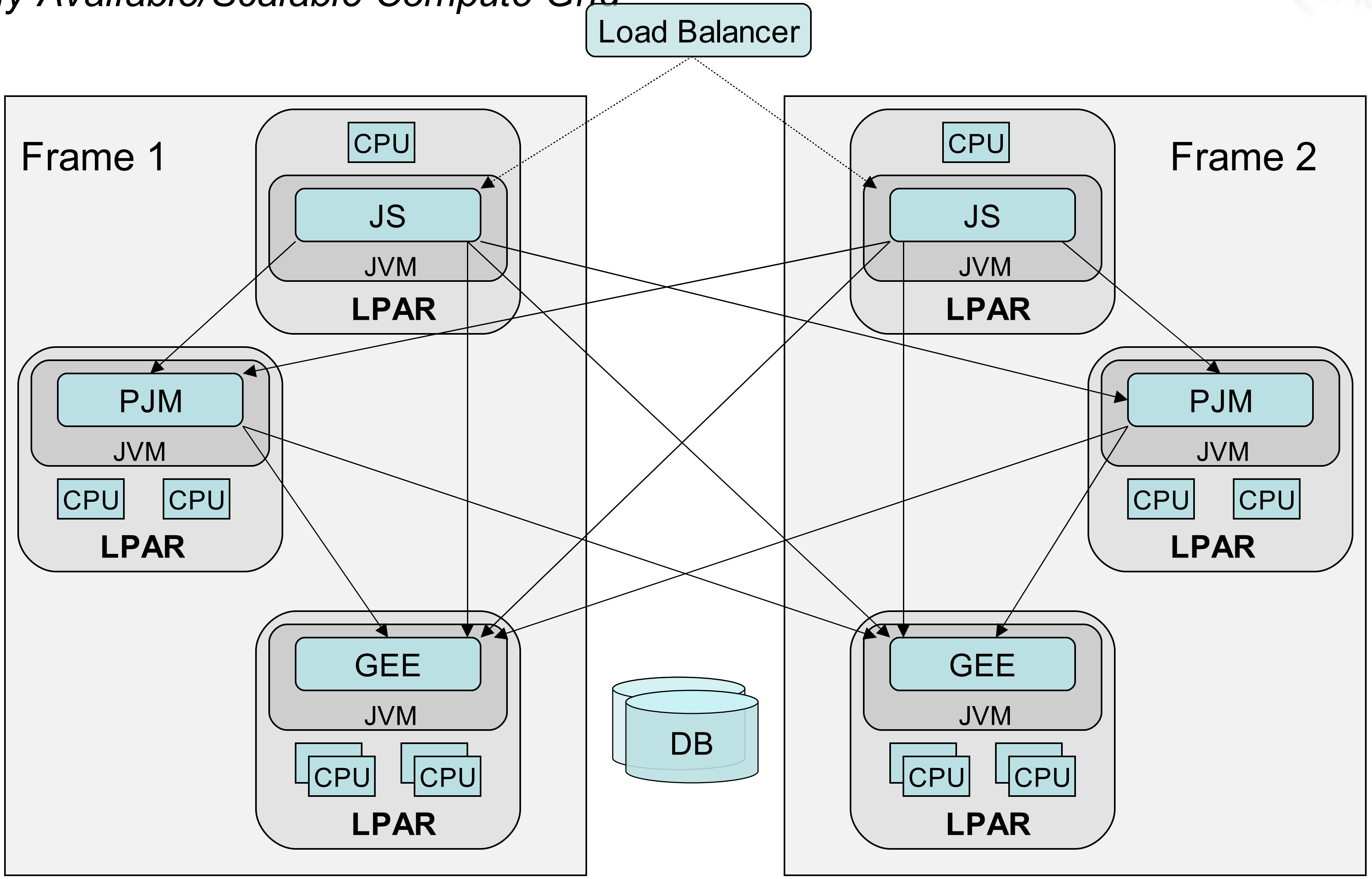


-Start with JZOS or J2SE-based Java batch infrastructure

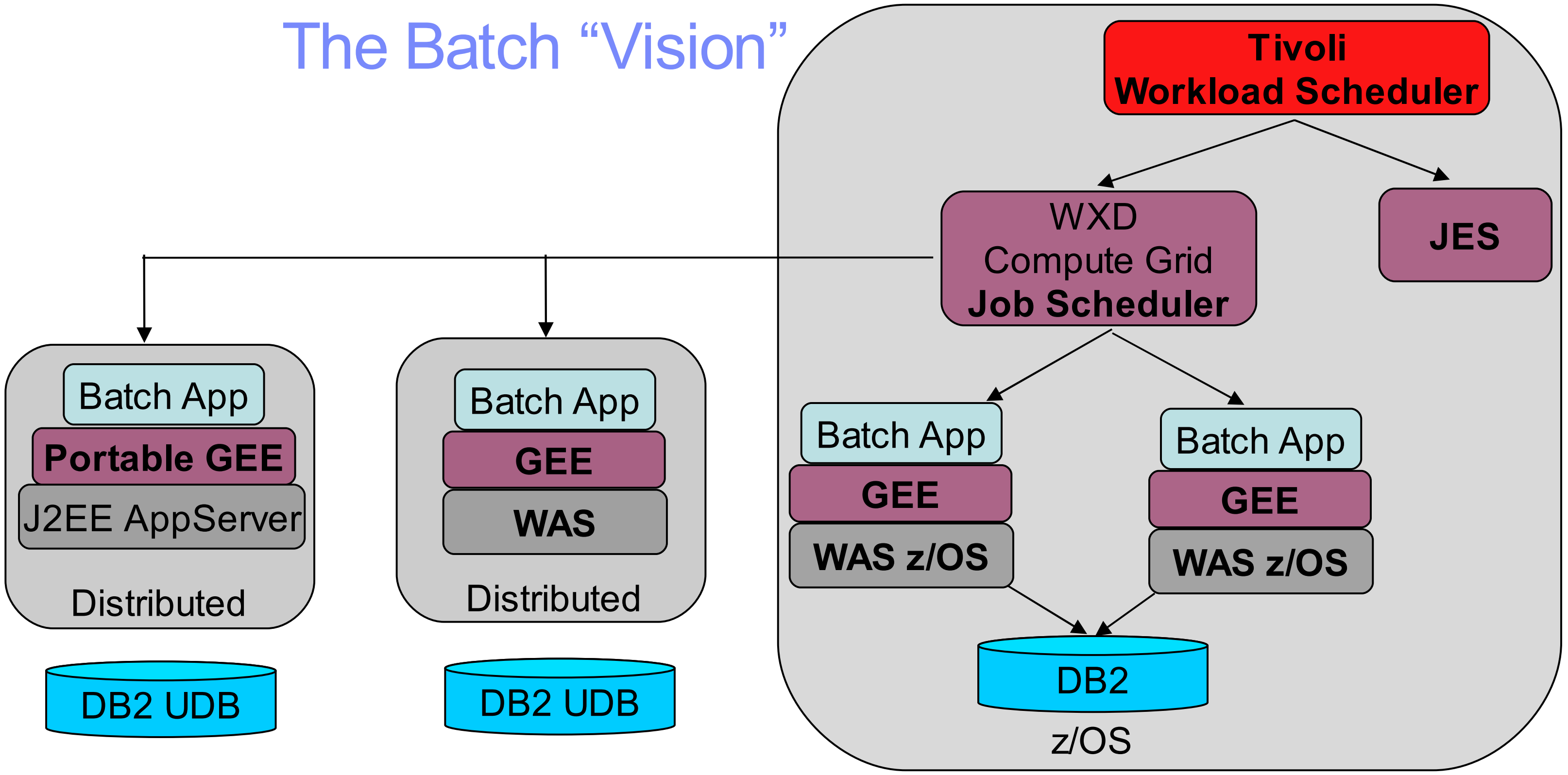
-Grow into Compute Grid-based Java batch infrastructure

-Leverage ***FREE*** Compute Grid development tools and frameworks to build Compute-Grid-Ready batch applications

Example Production Topology-
Highly Available/Scalable Compute Grid



The Batch “Vision”



- **Portable Batch applications** across platforms and J2EE vendors
- Location of the data dictates the placement of the batch application
- Flexible programming model, will host Spring Batch, JZOS, Compute Grid apps
- Centrally managed by your enterprise scheduler
- z/OS operational procedures manage batch across all platforms

Swiss Re –*Enterprise Batch Modernization via Java batch*

Challenge

- Maintain System z as strategic application and service delivery platform for batch and OLTP
- 15 TB of data with high rate of growth
- Decreasing population of COBOL developers

Solution

- WebSphere XD Compute Grid (Java batch) as the next generation batch execution platform
- zAAP and zIIP processor usage
- Strategic partnership between IBM and Swiss Re: shared development

Business Benefits

- Single view of business logic written as reusable services to be shared across OLTP & batch environments
- Strategic shared services infrastructure

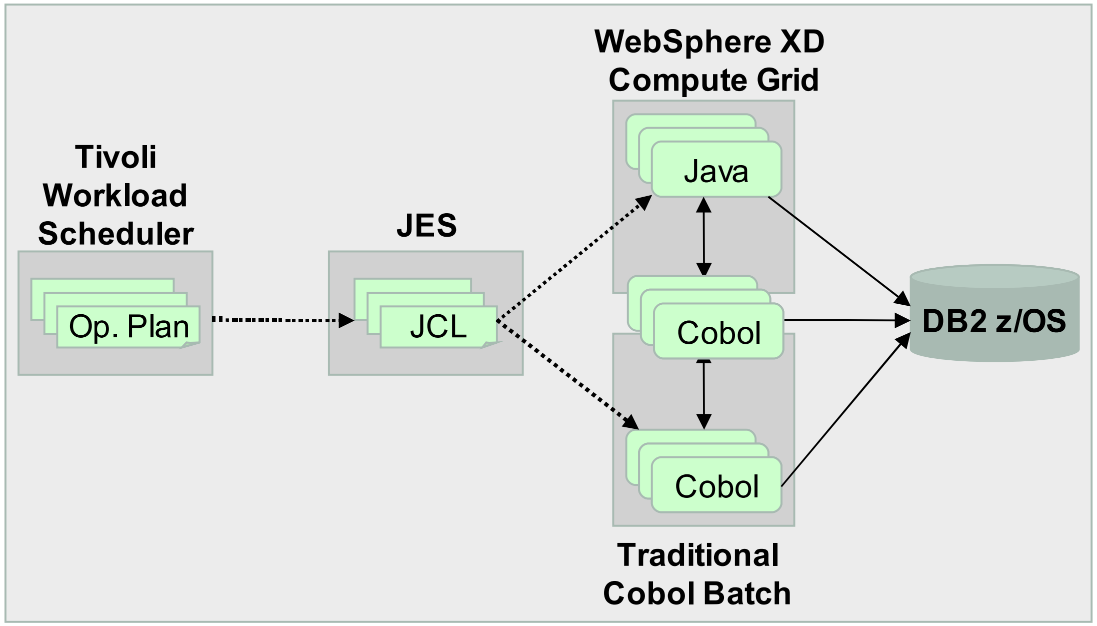
Swiss Re

Approach

Current Status

Production java batch running on WebSphere Compute Grid since April 2008.

System Z with z/OS



Today: Executing traditional batch with COBOL

Phase 1: Implement all new business logic in Java with XD Compute Grid

Phase 2: Share existing COBOL modules across both Java and COBOL domains

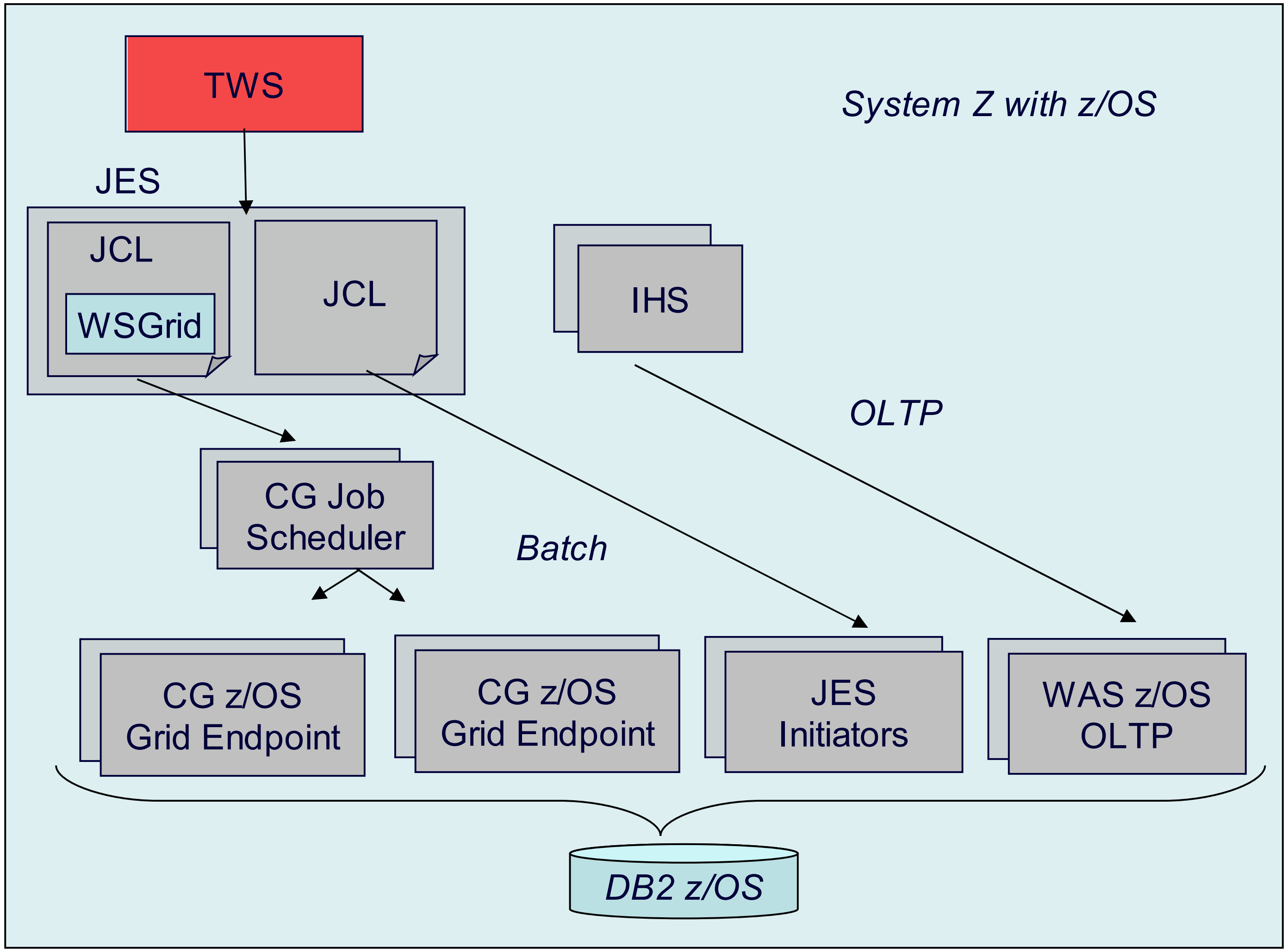
Phase 3: Incrementally migrate COBOL modules to Java with XD Compute Grid

Completion: All COBOL batch modules are replaced with Java, running in XD

SwissRe Batch and Online Infrastructure

- *Maintains close proximity to the data for performance*

- *Common:*
  - *-Security*
  - *-Archiving*
  - *-Auditing*
  - *-Disaster recovery*

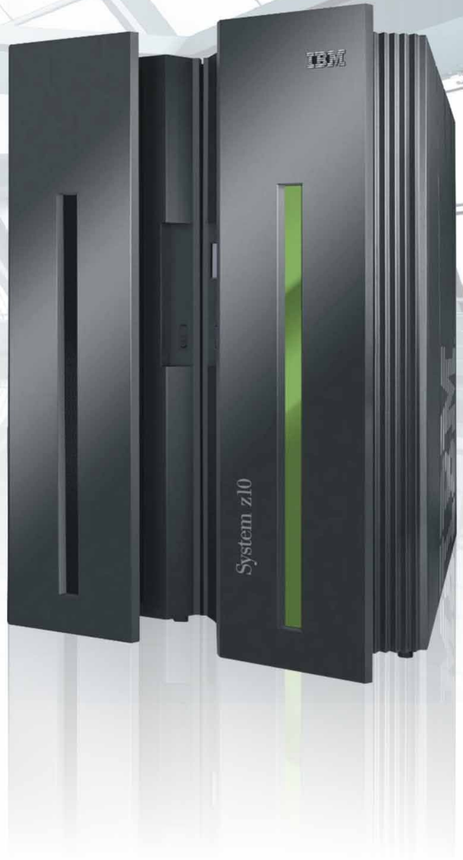


Agenda

- Batch Computing
  - Drivers for modernisation
- WebSphere Compute Grid Overview
  - Batch Platform
  - Application Development & Tooling
- WebSphere Compute Grid on z/OS
  - Benefits
  - Customer example
- Summary

IBM System z and WebSphere Compute Grid – The ultimate batch environment

- **Maximize Performance**
  - Benefit from z/OS optimizations for data access on the mainframe
  - Apply massively parallel execution with Compute Grid
- **Assure Recoverability**
  - Batch Checkpoints are backed by JTA transactions with Compute Grid
- **Ensure Availability**
  - Leverage WebSphere and platform (System Z, P, etc) High Availability
- **Reduce Operations Costs**
  - Integrated with WebSphere Virtual Enterprise for Virtualized Distributed Runtimes
  - Leverages zAAP processors on System Z
- **Reduce Maintenance Costs**
  - Integrate processes for both OLTP and Batch
  - Share business logic across both domains
  - Leverage existing batch processing artifacts such as enterprise schedulers.



IBM
Thank
You
IBM

Trademarks