WebSphere Virtual Enterprise & WebSphere XD Compute Grid on z/OS

Snehal S. Antani
antani@us.ibm.com
WebSphere XD Technical Lead
SOA Technology Practice, ISSW, SWG, IBM
http://snehalantani.googlepages.com

Agenda

- The Product Formerly Known as “WebSphere XD”
- Application Resiliency with WebSphere Virtual Enterprise
  - Application Editions
  - Health Monitoring
  - Checkpoint restart
- Batch Processing with WebSphere XD Compute Grid
  - What is “Batch” ?
  - Stamping out “Maverick” Batch
  - Exploring Batch “Modernization”
  - Technical Overview of Compute Grid
  - Building Batch Applications
  - The Grand Strategy

Appendices

- WebSphere Virtual Enterprise Details
- WebSphere XD Compute Grid Influences
- WebSphere XD Compute Grid Best Practices
- Using the Compute Grid Infrastructure
- Some example use-cases of Compute Grid
- Approaches for Batch Modernization
- SwissRe and Batch Modernization

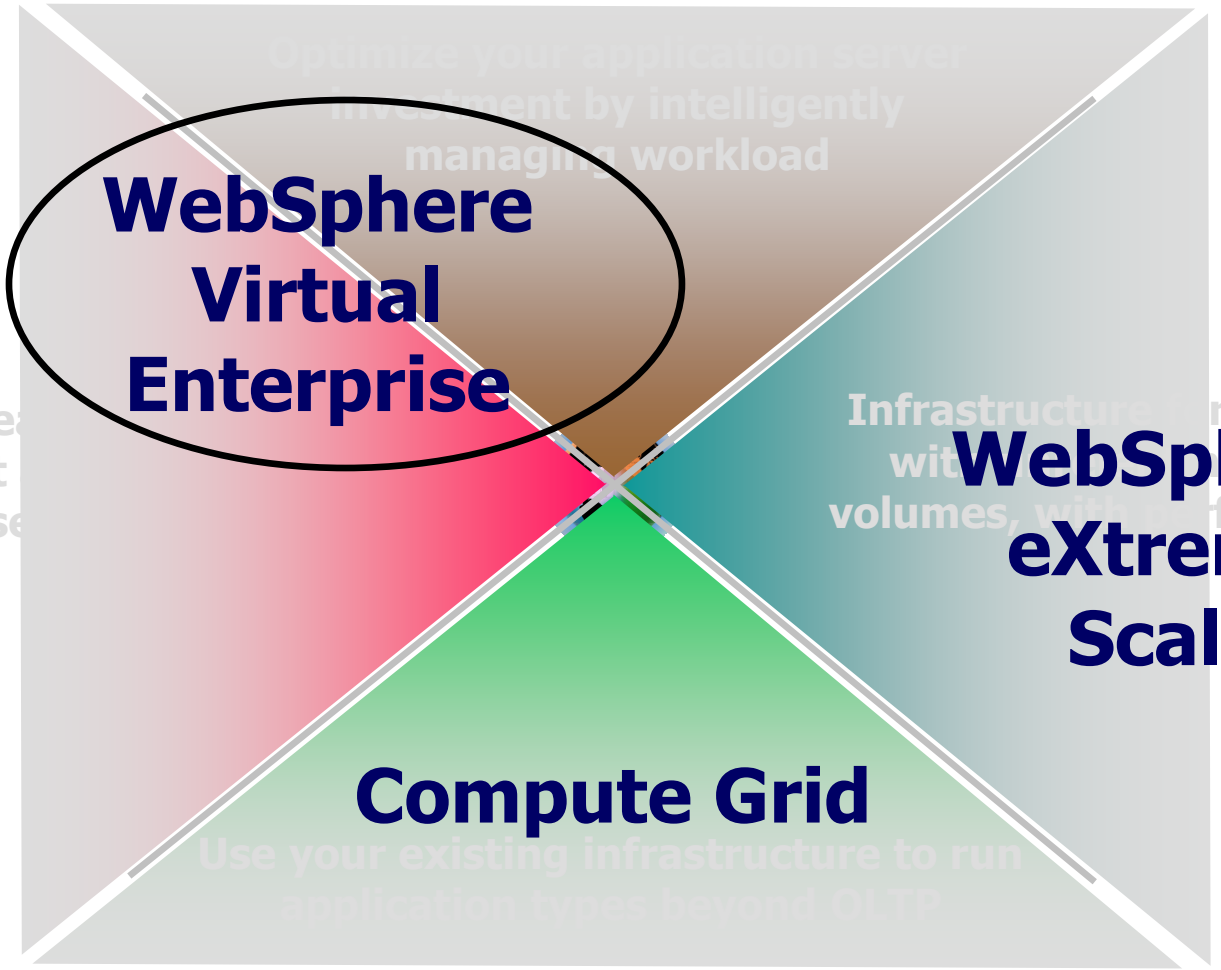
The Product Formerly Known as *WebSphere Extended Deployment (XD)*

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

Compute Grid	Virtual Enterprise	eXtreme Scale
- *Transactional Batch* - *Compute Intensive Tasks* - *Manage non-Java workloads* - *z/OS Integration Patterns*	- *On-Demand Router* - *Extended Manageability* - *Application Editions* - *Health Management* - *Runtime Visualization* - *Virtualization*	- *Distributed Caching* - *Partitioning Facility* - *In-memory Databases*

WebSphere XD Packaging Structure

Available as a single, integrated package or as 3 individual components



WebSphere Virtual Enterprise

- *Improving the **resiliency** of your middleware infrastructure*
  - A health management infrastructure
  - Continuous availability – interruption-free application updates
  - Checkpointing the configuration of the WebSphere runtime
  - Visualization technologies
- *Features for Distributed platforms*
  - Application virtualization services
  - A goals-oriented runtime for WAS and Non-WAS middleware
  - Service policies and relative application priorities
  - multi-media applications over voice and video via SIP

Automatic, Sense & Respond Management

Challenge:

- ◇ Provide operational control so that my IT staff can easily manage my environment
- ◇ Gain insight into the performance and operations of applications & servers across my entire heterogeneous (and distributed) application server infrastructure
- ◇ Proactively address and correct issues before they cause IT and business impacts
- ◇ Give me the information I need to do historical analysis, capacity planning, and chargeback for resource usage
- ◇ Decrease management and administration costs

WVE contains comprehensive and integrated management capabilities

Operational Management: Monitoring

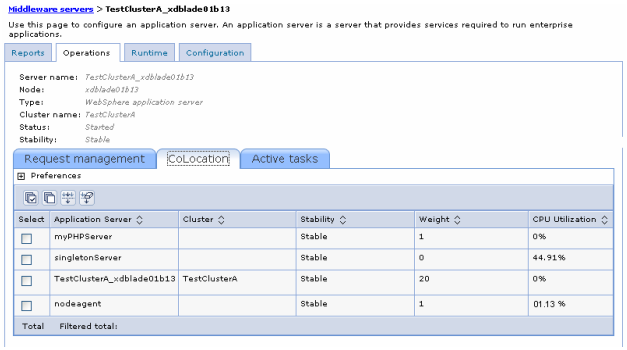
In a Dynamic and Heterogeneous Environment

WVE provides a set of views for understanding and managing the dynamic goals directed environment applications are hosted in

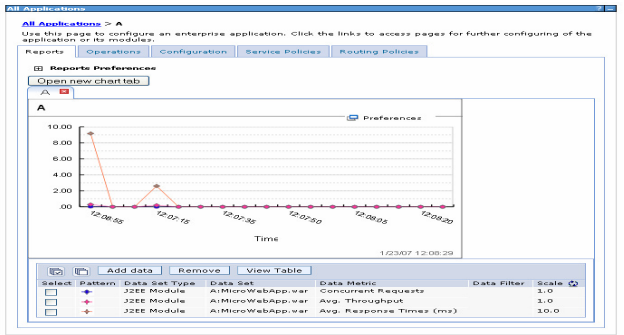
The administrative console is enhanced with Operations and Reporting tabs

Operations tab provides insight into

- The stability of the resource
- How work for the resource is actively being managed
- Outstanding tasks that need operators to act upon
- Where the resource is currently running

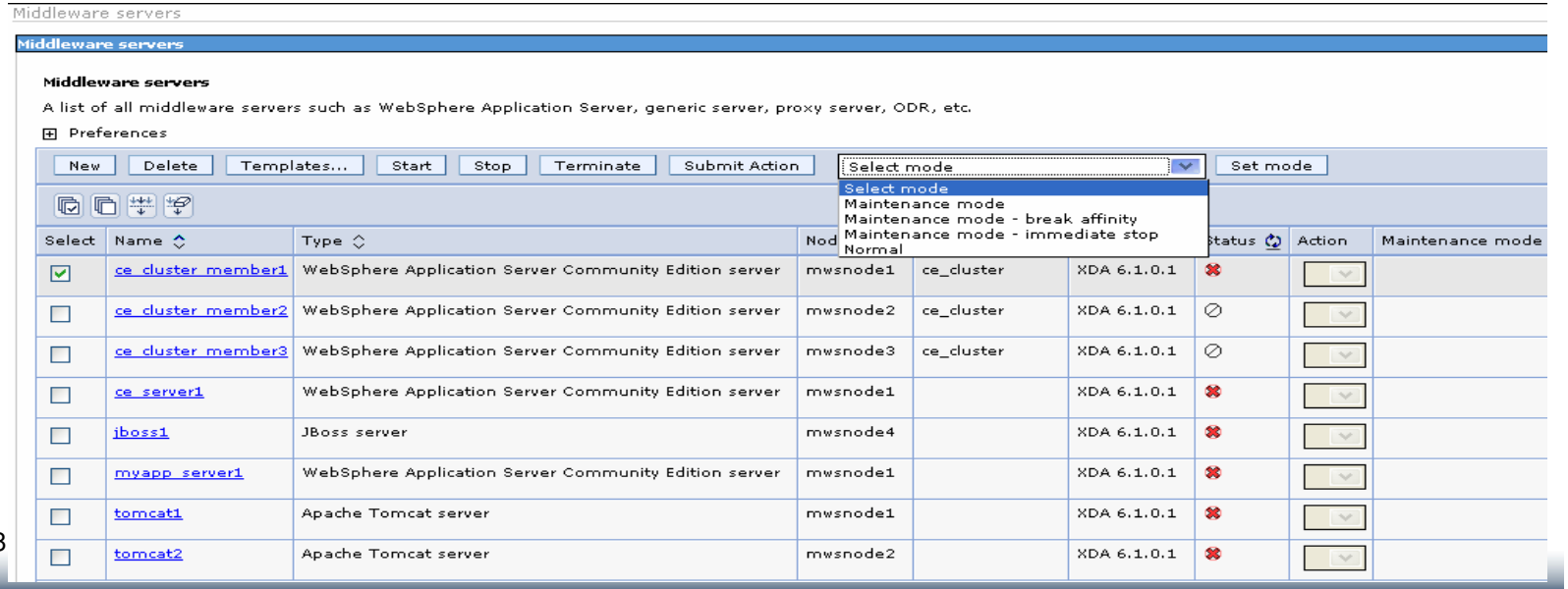


Reporting tab allows for in depth charts to be viewed to understand the performance of the environment



Server Maintenance Mode

- WVE provides the capability to isolate a running server (of any type) from production traffic : server maintenance mode. This allows for problem determination to be performed on the server or other maintenance without disruption to production traffic.
- Options for maintenance mode:
  - Stop server
  - Leave server running and keep affinity
  - Leave server running and break affinity
- If the server is a member of a dynamic cluster, a new cluster member will first be started before the server is placed into maintenance mode in order to assure the minimum policy on the dynamic cluster is met.



Health Management

Challenge:
- ◇ Recognize health issues in my environment and automatically correct them
- ◇ Allow me to determine what I consider a health condition and the appropriate corrective action

Solution → **WVE Health Management Framework**

WVE offers out-of-the-box health policies and actions across all supported application environments and allows them to be customized

Comprehensive Health Policies

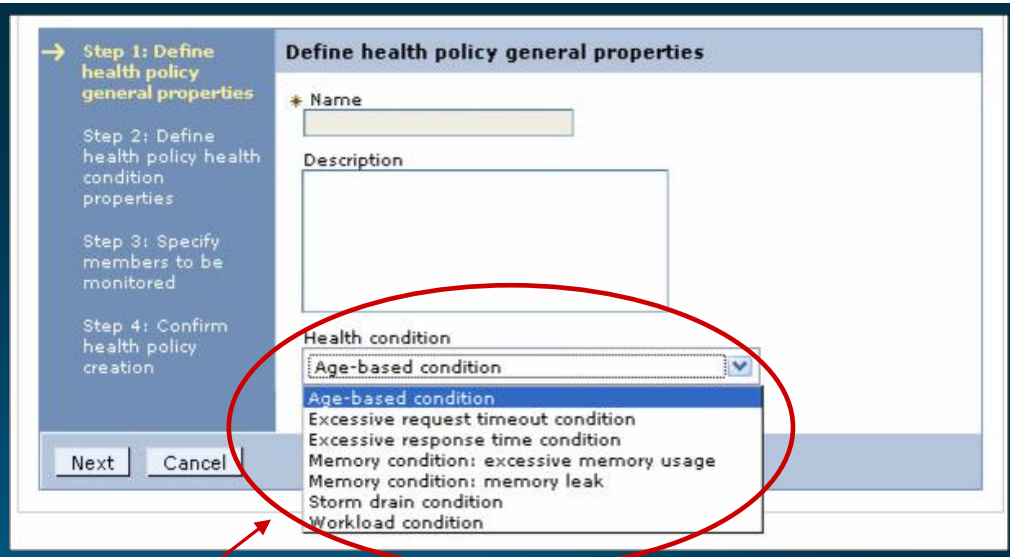
Customizable Health Conditions

Customizable Health Actions

Health Management – Health Policies

Helps mitigate common health problems before production outages occur

- Health policies can be defined for common server health conditions
- Health conditions are monitored and corrective actions taken automatically
  - Notify administrator
  - Capture diagnostics
  - Restart server
- Application server restarts are done in a way that prevent outages and service policy violations



Health Conditions

- **Age-based:** *amount of time server has been running*
- **Excessive requests:** *% of timed out requests*
- **Excessive response time:** *average response time*
- **Excessive memory:** *% of maximum JVM heap size*
- **Memory leak:** *JVM heap size after garbage collection*
- **Storm drain:** *significant drop in response time*
- **Workload:** *total number of requests*

Health Management – Custom Health Conditions

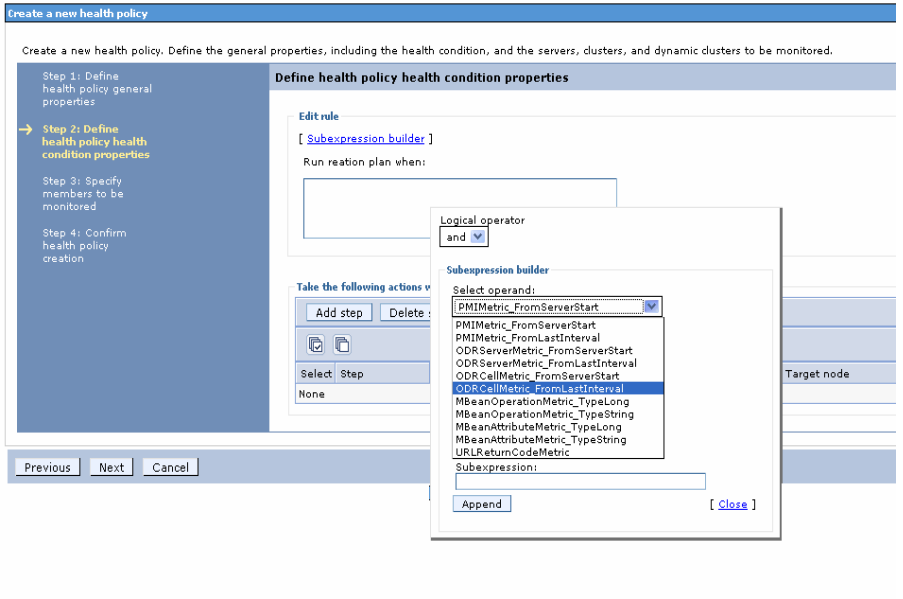
Flexibility to determine what an "unhealthy" condition is…

Custom expressions can be built which use metrics from:

- The On Demand Router, URI return codes
- Base PMI metrics, MBean operations and attributes (WAS only)

Complex expressions using a mix of operands is supported

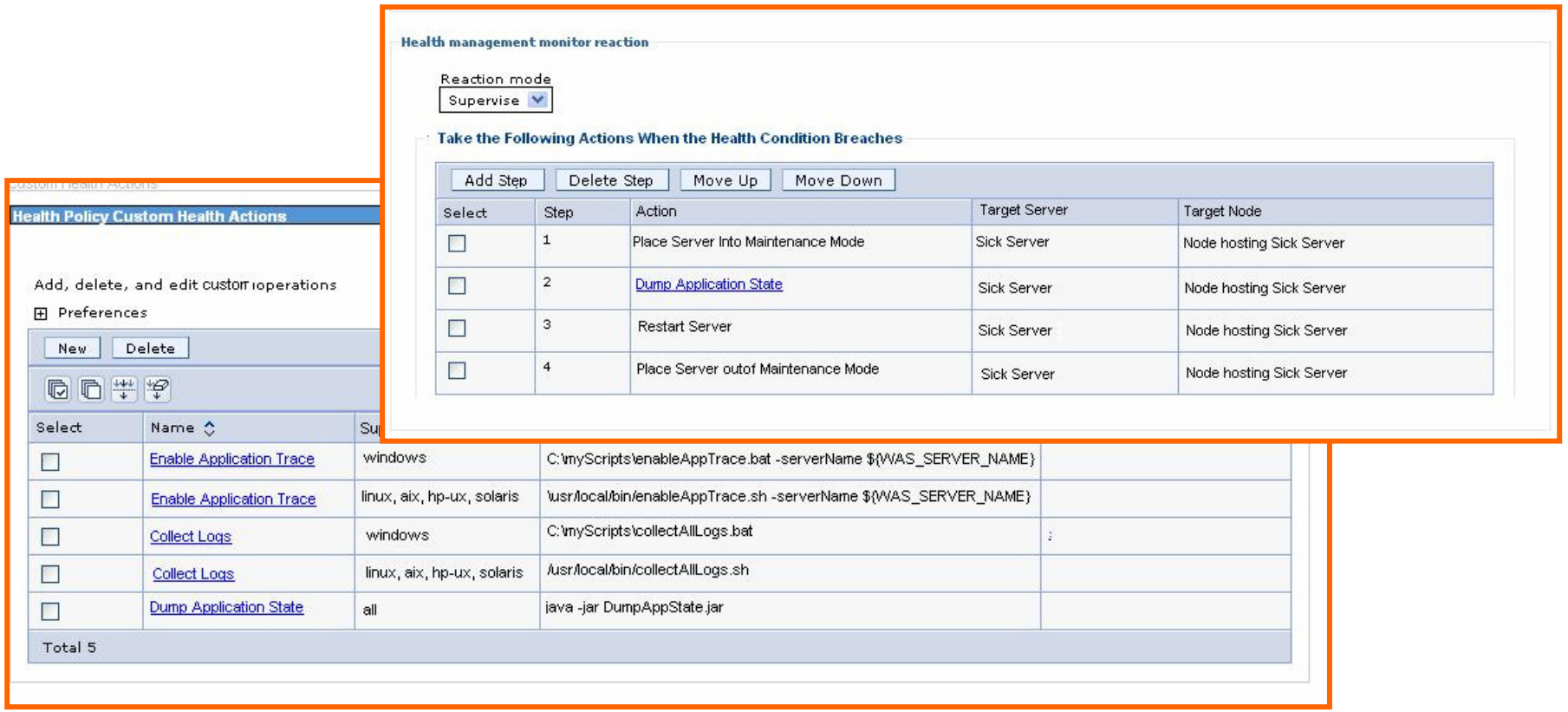
```
C:\WebSphere\AppServer\bin>wsadmin
WASX7209I: Connected to process "dmgr" on node dabtcCellManager01 usi
nector;  The type of process is: DeploymentManager
WASX7029I: For help, enter: "$Help help"
wsadmin>$AdminTask createHealthAction { -name logItAction2 -nonJava {
e logit.exe -osNames "windows" -workingDir c:\logit\temp } }

wsadmin>$AdminConfig save

wsadmin>$AdminTask createHealthPolicy { -name testPolicy -reactionMod
 -addMember { {CELL dabtcCell02} } -addAction { {HEAPDUMP 1}{CUSTOM 2
n2 dabtcNode01 server1} } -addCondition { -type CUSTOM -expression "C
ric_FromServerStart$errors > 100L" } }

wsadmin>$AdminConfig save
```



Health Management – Custom Health Actions

Take Control!

Provides flexibility by allowing the definition of custom actions allowing administrators to define an action plan to be carried out when the unhealthy situation detected.



Supported Health Policies

Predefined Health Policy	WebSphere Application Server	WebSphere Community Edition 2.0
Age-based policy	✓	✓
Workload policy	✓	✓
Memory leak detection	✓	
Excessive Memory usage	✓	✓
Excessive Response Timeout	✓	✓
Excessive Request Timeout	✓	✓
Storm Drain Detection	✓	✓

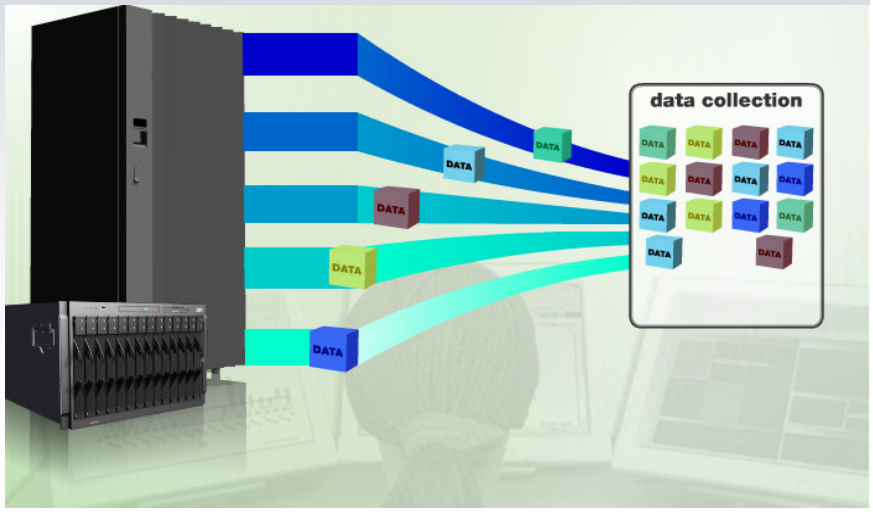
Data Logging

Challenge:

- A lot is going on in my environment. I need to be able to log information so I can do historical trend analysis of how my infrastructure is performing.
- My infrastructure resources are shared across multiple applications and users. I need an easy way to meter usage and appropriately chargeback to users and/or departments.

WVE contains comprehensive data logging of applications, users and resources; in WVE 6.1 content in logs is now configurable and aggregated for easily hooking into accounting and chargeback products

- **Comprehensive logging** of application, resource and workload information across WVE's autonomic systems
- **Historical trend analysis** using either pre-packaged or customized reports with innovative visualization techniques
- **Easily hookup to accounting and chargeback systems** such as Tivoli Usage and Accounting Manager



Manage Multiple Application Versions

Challenge:

- I want to support different versions of my applications for my users or customers for continuous availability
- I need a more agile production deployment process, where I can quickly back-off new application versions to prevent loss of service
- I'd like to better support iterative development; and potentially use my free resources in my production environment for application testing

Solution → **Application Edition Manager**

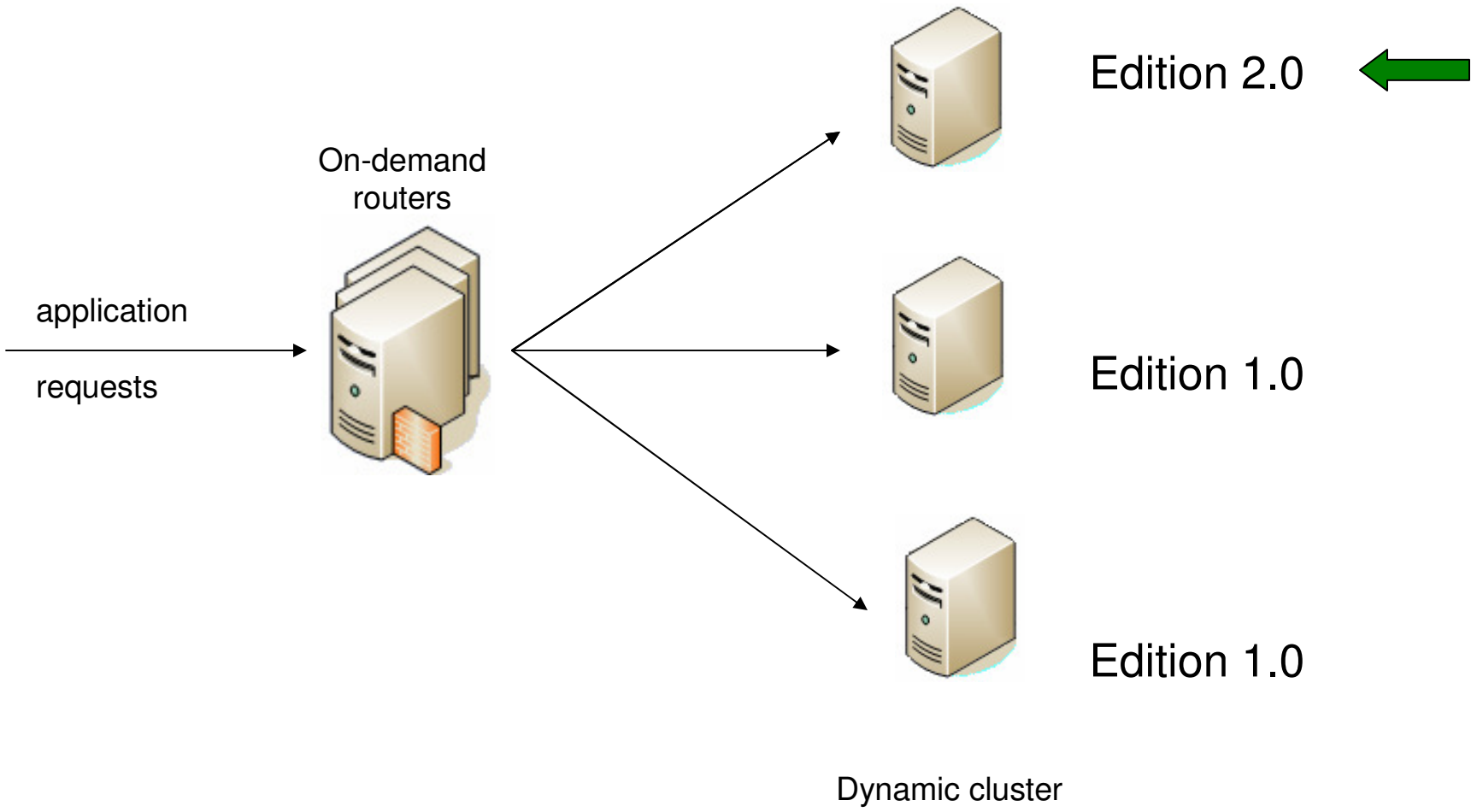
Dynamically introduce, run, and manage multiple versions of the same application in your infrastructure

- **Coordinates the activation of application editions** and the routing of requests to the application
- **Validation Mode** enables final pre-production testing of an application edition by a select group of users
- **Routing Rules** allow intelligent routing to multiple application editions in production

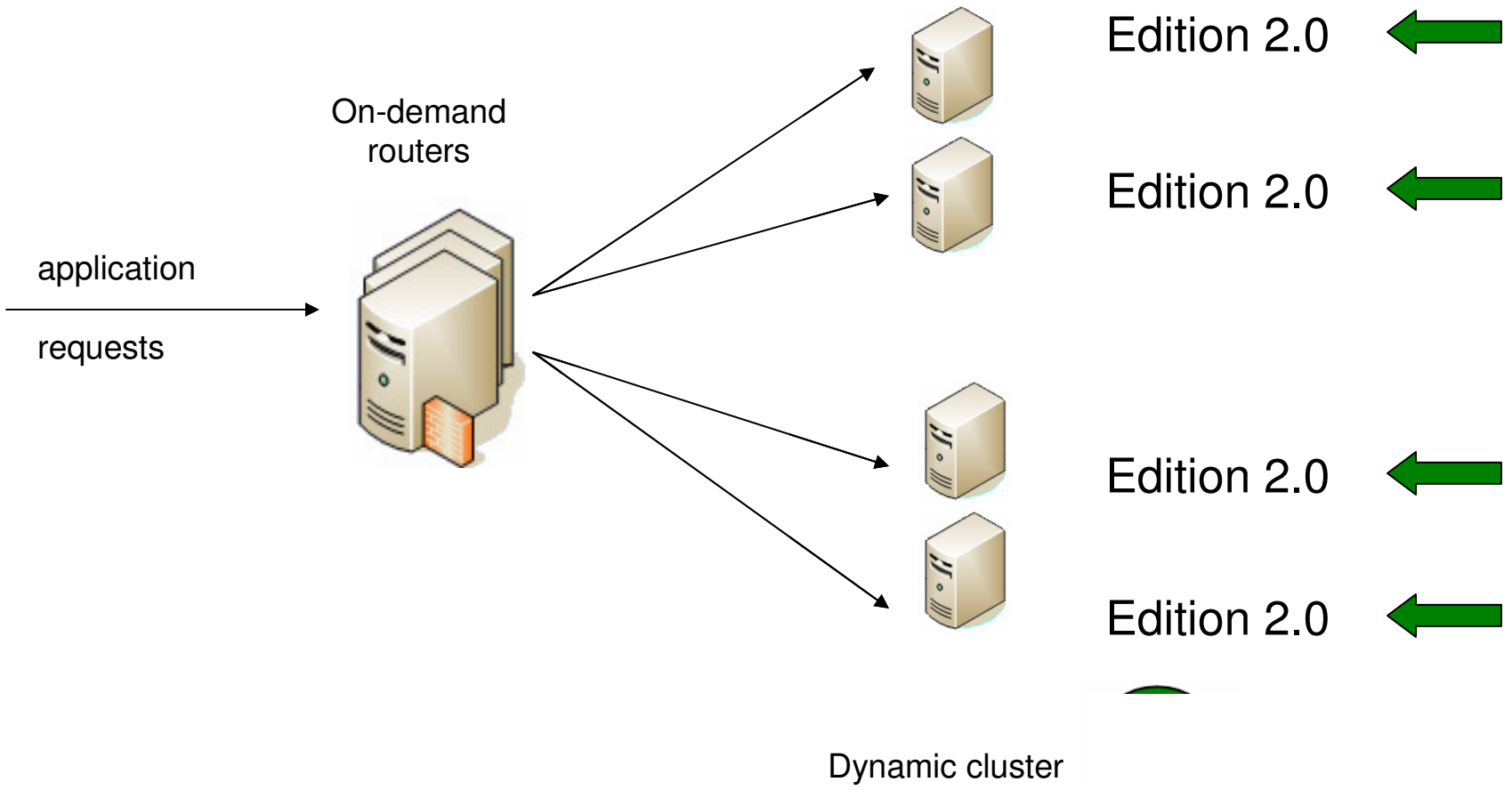
Application Versioning Support Matrix

Feature	Managed Web applications (HTTP)	IIOP, EJB, or JMS applications	PHP applications	Unmanaged Web applications
Multiple editions of applications	Supported	Supported	Supported	Supported
Application rollout	Supported	Supported	Supported	Not Supported
Interruption-free application update, driven by the ODR and HTTP communication	Supported	Not supported (If the EJB, JMS, or IIOP components are directly exposed to an external client. The ODR does not support communication with IIOP)	Supported	Not Supported
Validation mode	Supported	Not Supported	Supported	Not Supported
Concurrent activation	Supported	Not Supported	Supported	Supported

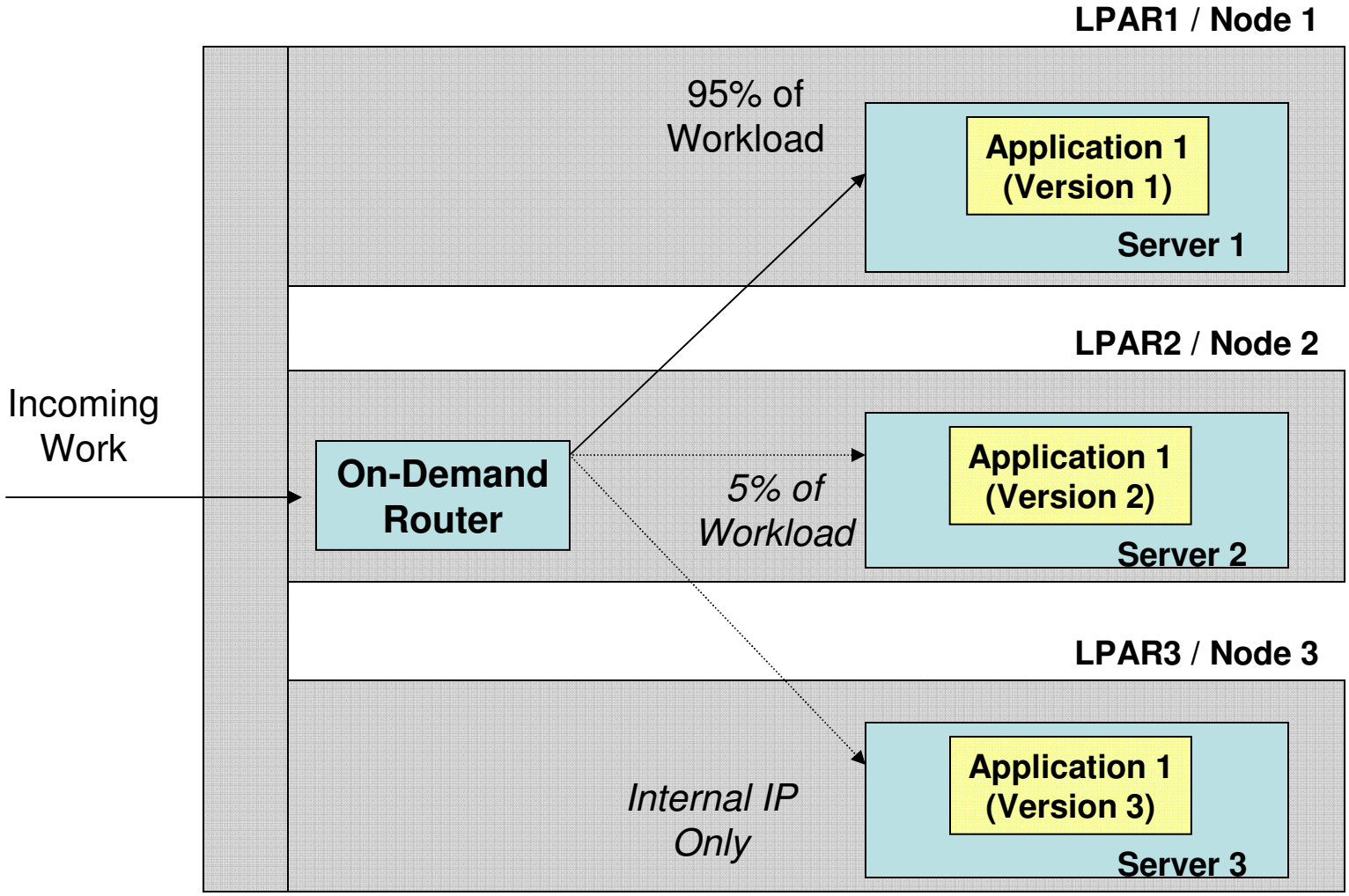
Application Editions – Group Rollout



Application Editions – Atomic Rollout



Application Edition Management & Application Test



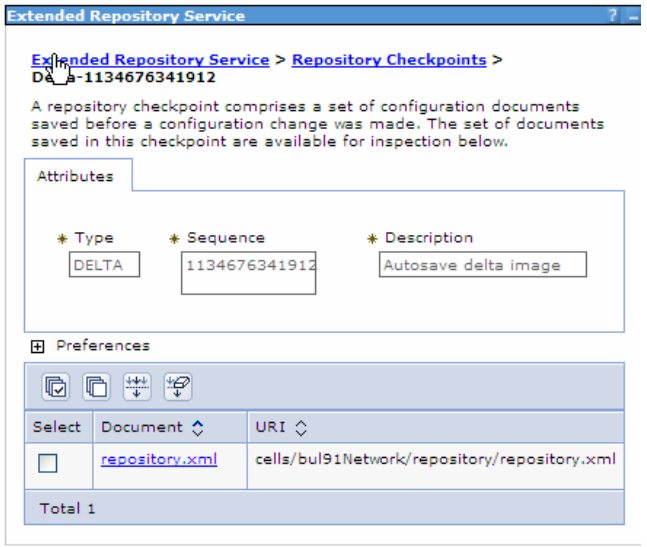
Extended Repository Service

- Ability to keep automatic or manual checkpoint of the configuration repository of WebSphere.
- Full checkpoints are manually created and named and contain a full copy of the repository
- Delta checkpoints are kept automatically and contain a subset of the repository that was changed in a given save operation
- Repository changes can be unrolled back to a previous state.

New | Delete | Restore

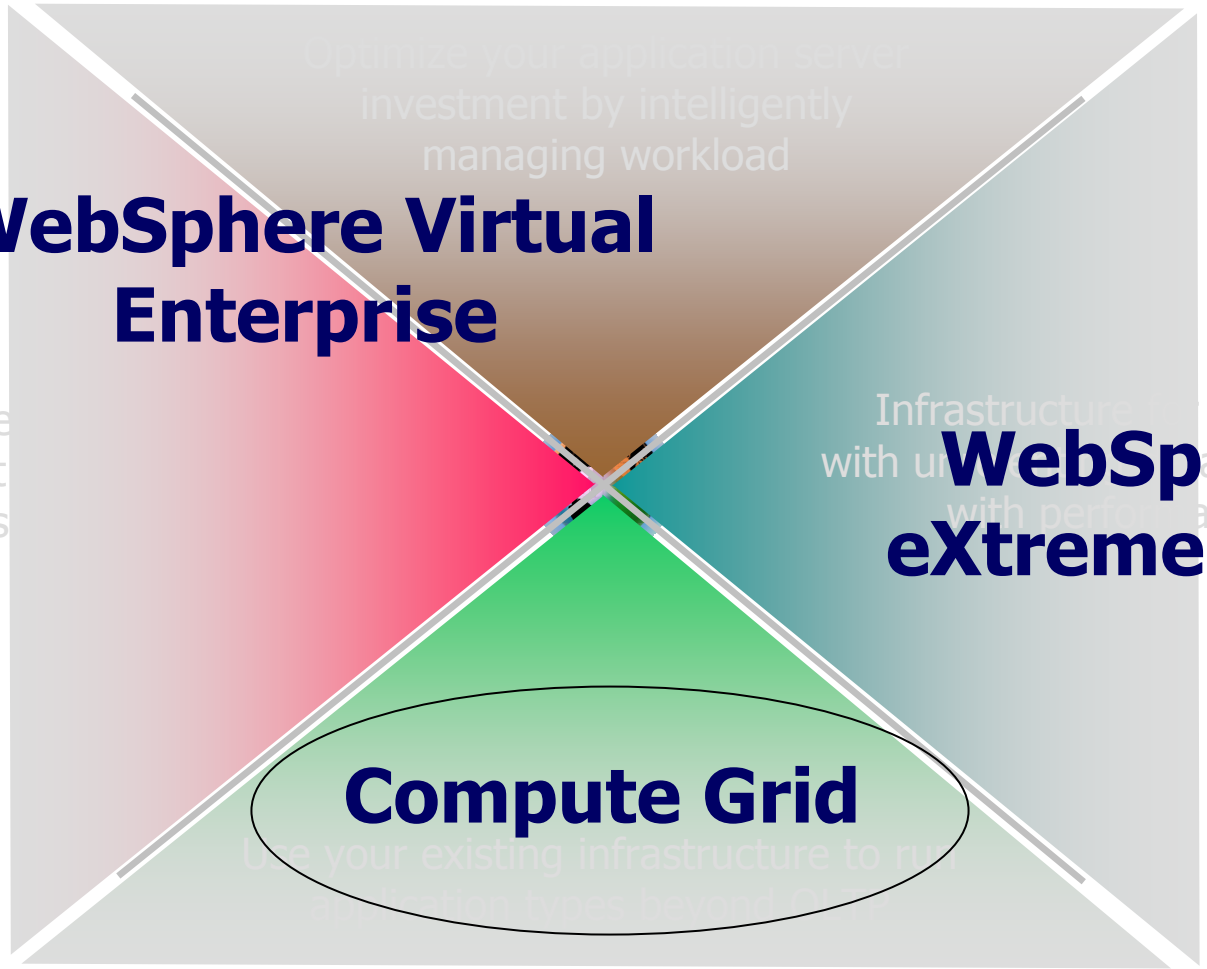
Select	Name	Documents	Type	Sequence	Description
☐	Delta-1134676341912	1	DELTA	1134676341912	Autosave delta image
☐	Working	1116	FULL	1134676365924	A Working Config

Total 2



WebSphere XD Packaging Structure

Available as a single, integrated package or by 3 individual components



What is “Batch” ?

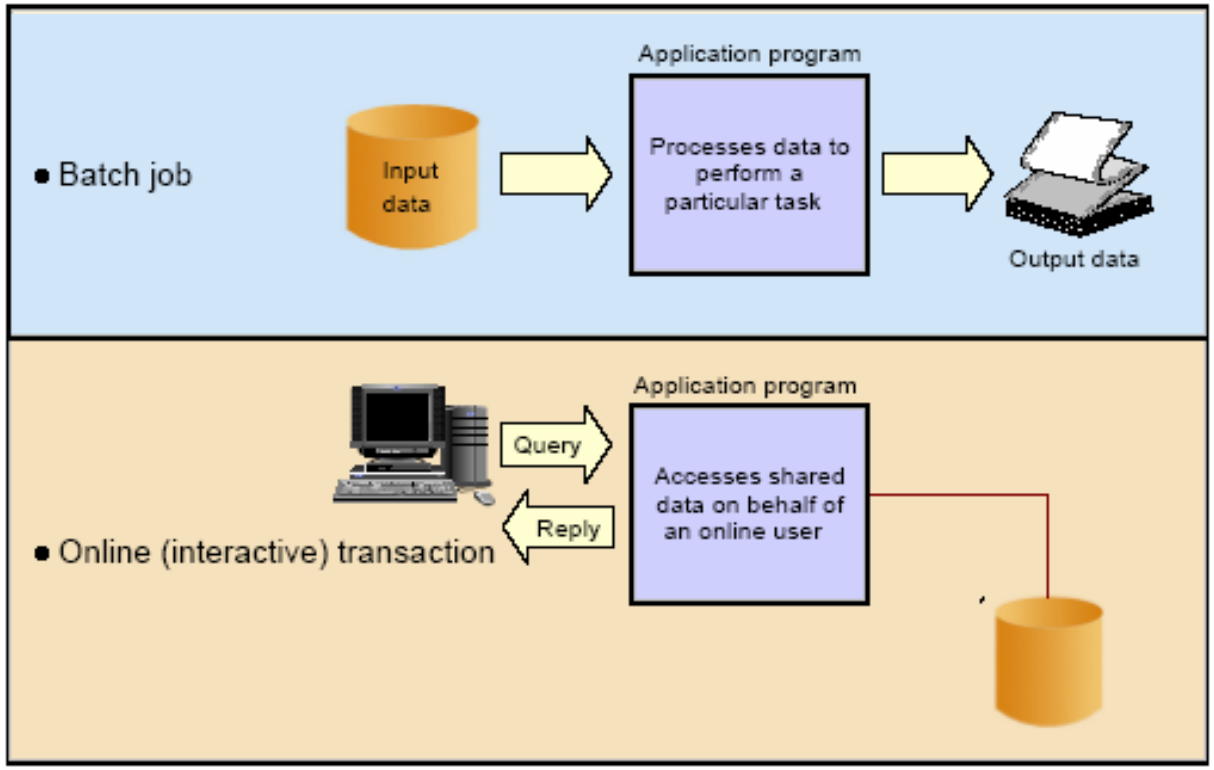
What are “Batch Jobs”?

- Batch processing is the task of processing a *tremendous* amount of data within a *narrowing* window of time.

- Batch has some high expectations….
  - High Availability
  - Scalability
  - Reliability
  - Security
  - Operational Management
  - Performance

- Tremendous amount of infrastructure and operational procedures have been built around batch systems

Some Examples of Batch Jobs…



Examples

- Payment Processing
- Shipment Processing
- Report Generation
- Claims Processing
- Inventory Management
- End of Day/Month/Quarter/year processing

Batch and SOA

Reusing business services is a fundamental principle of SOA

Batch workloads are an integral part of any IT infrastructure

How do you integrate your batch & OLTP environments with a common services infrastructure?

How do you eliminate “maverick” batch and deliver an enterprise-wide batch infrastructure?

Stamping out “Maverick” Batch

Maverick Batch…

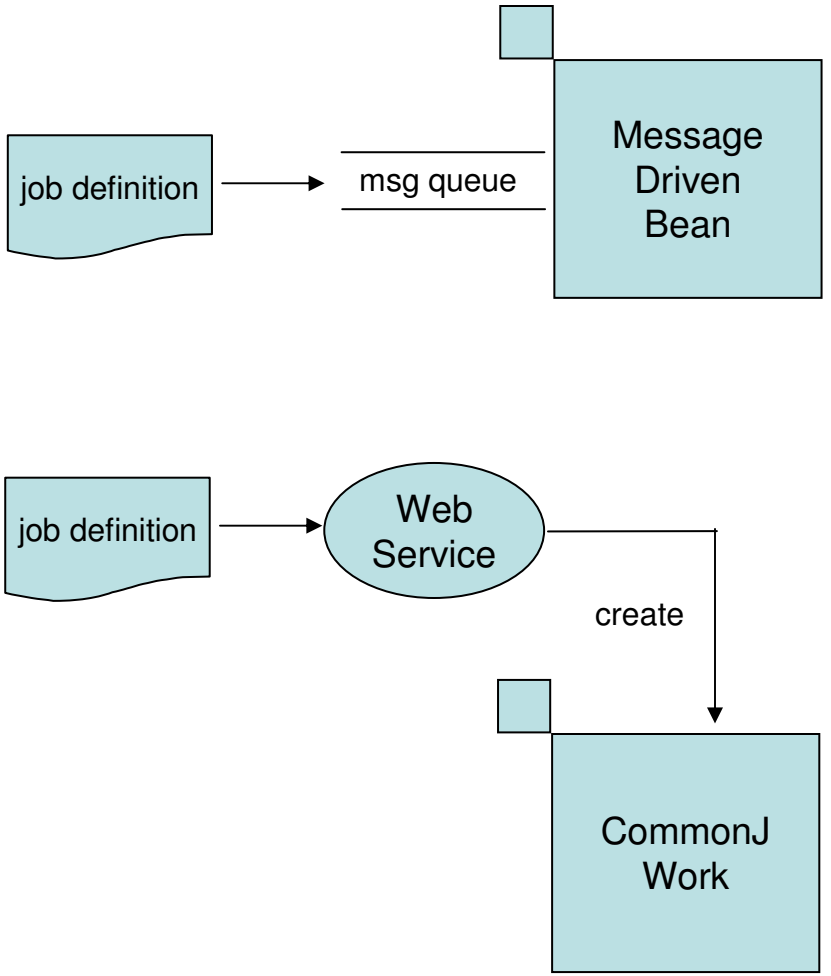
- “Maverick” Batch is ***BAD***
- “Maverick” Batch is an anti-pattern
- “Maverick” Batch distracts customers from solving business problems
- “Maverick” Batch can be expensive
- You probably have “Maverick” Batch….

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
- …

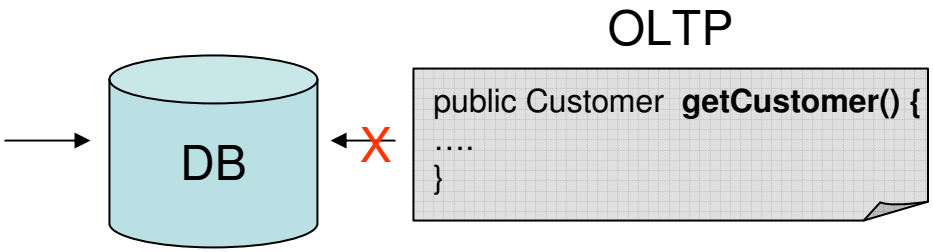


OLTP and Batch Interleave

```
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



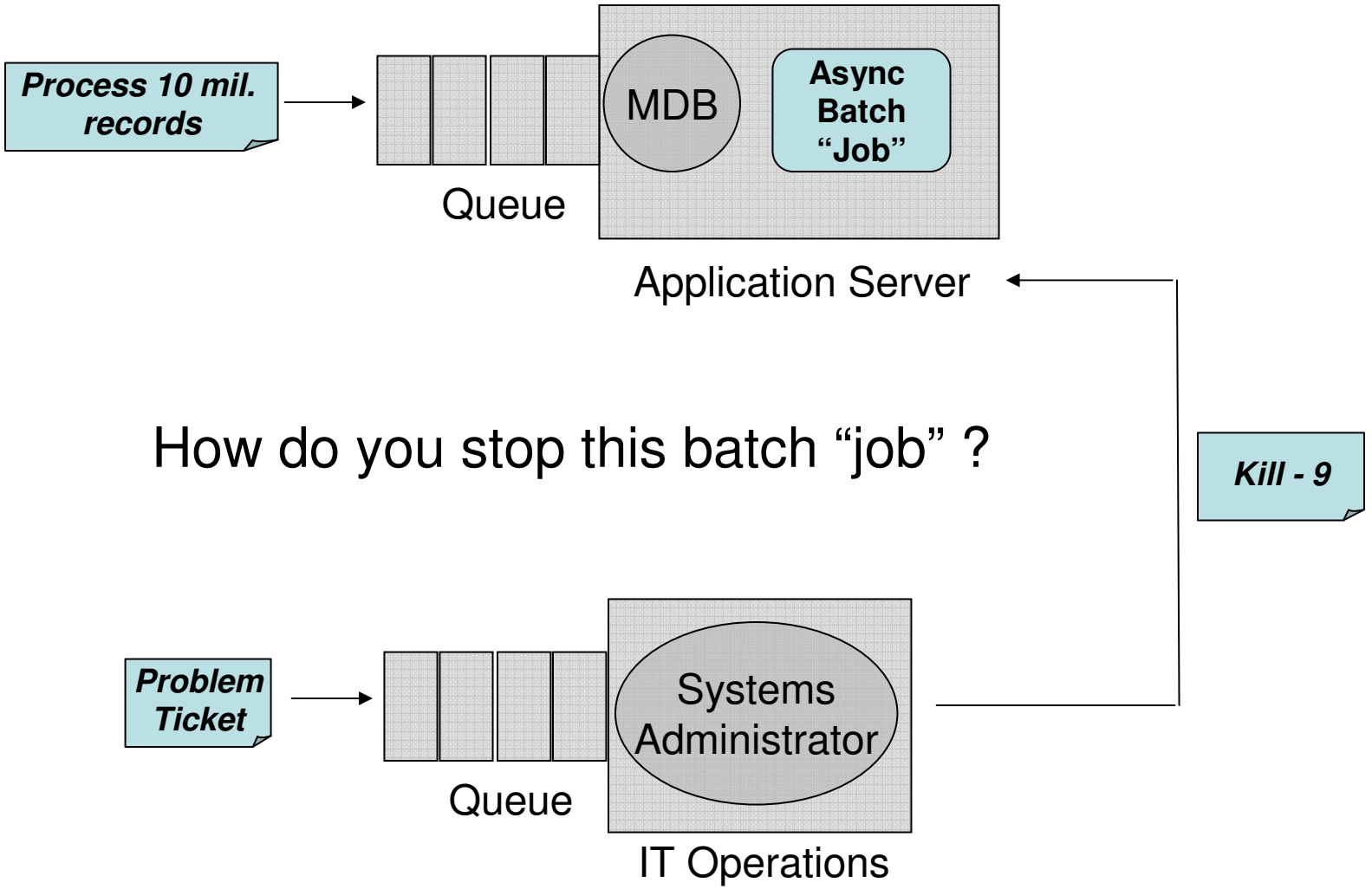
-Batch application's hold on DB locks can adversely impact OLTP workloads

-OLTP Service Level Agreements can be breached

-How do you manage this?

-WLM will make the problem worse!

Operational Control in “Maverick” Batch

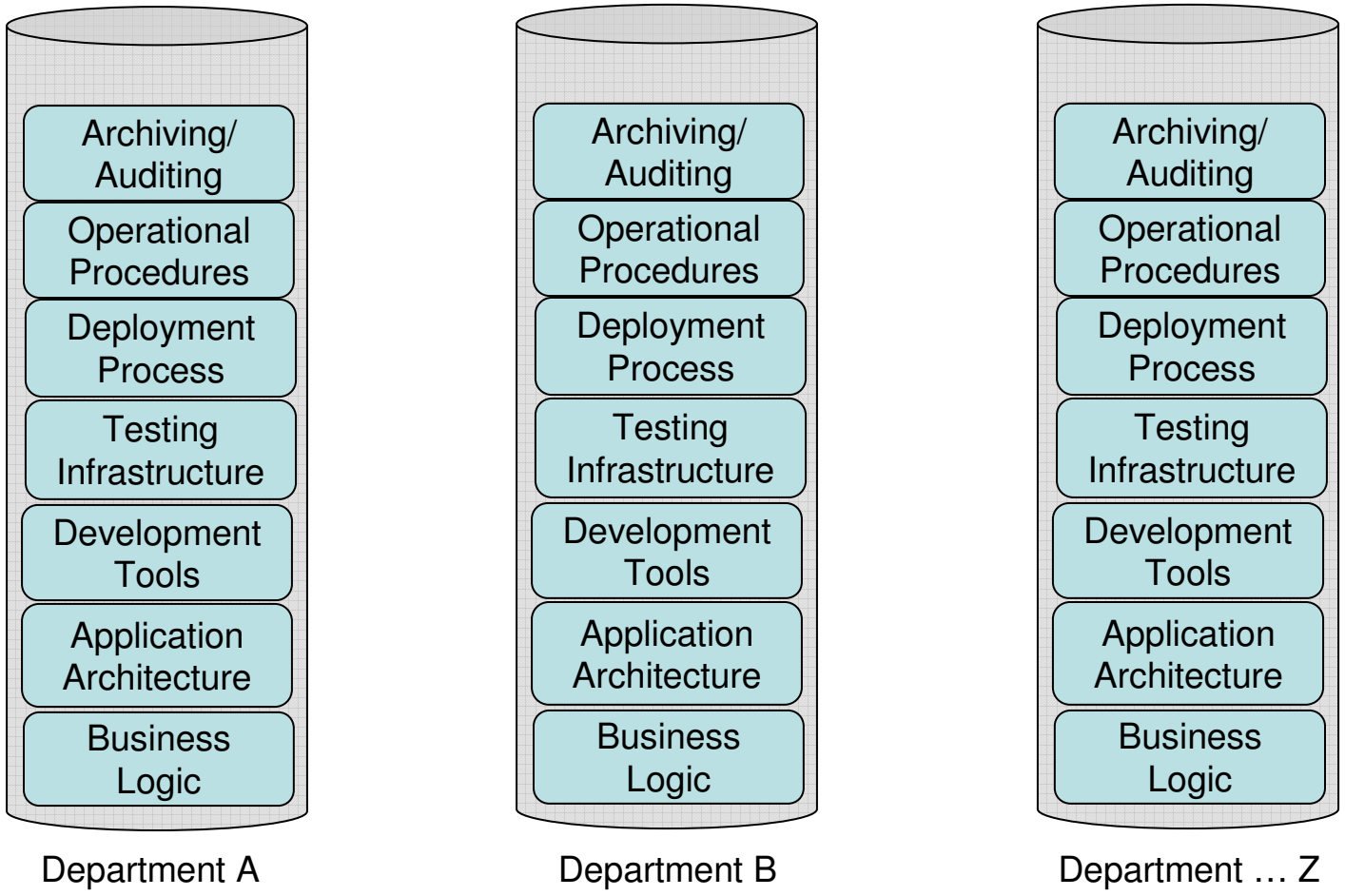


Why Maverick Batch is ***BAD...***

- Customers are not in the business of building/owning/maintaining infrastructure code
  - Developers ***love*** writing infrastructure code
  - IT Managers ***avoid*** owning and maintaining infrastructure code
  - IT Executives ***hate*** paying for code that doesn't support the core business
- Learn from history…
  OLTP has evolved, now it's time for Batch
  See: Arrival of Application Servers for OLTP ~15 years ago

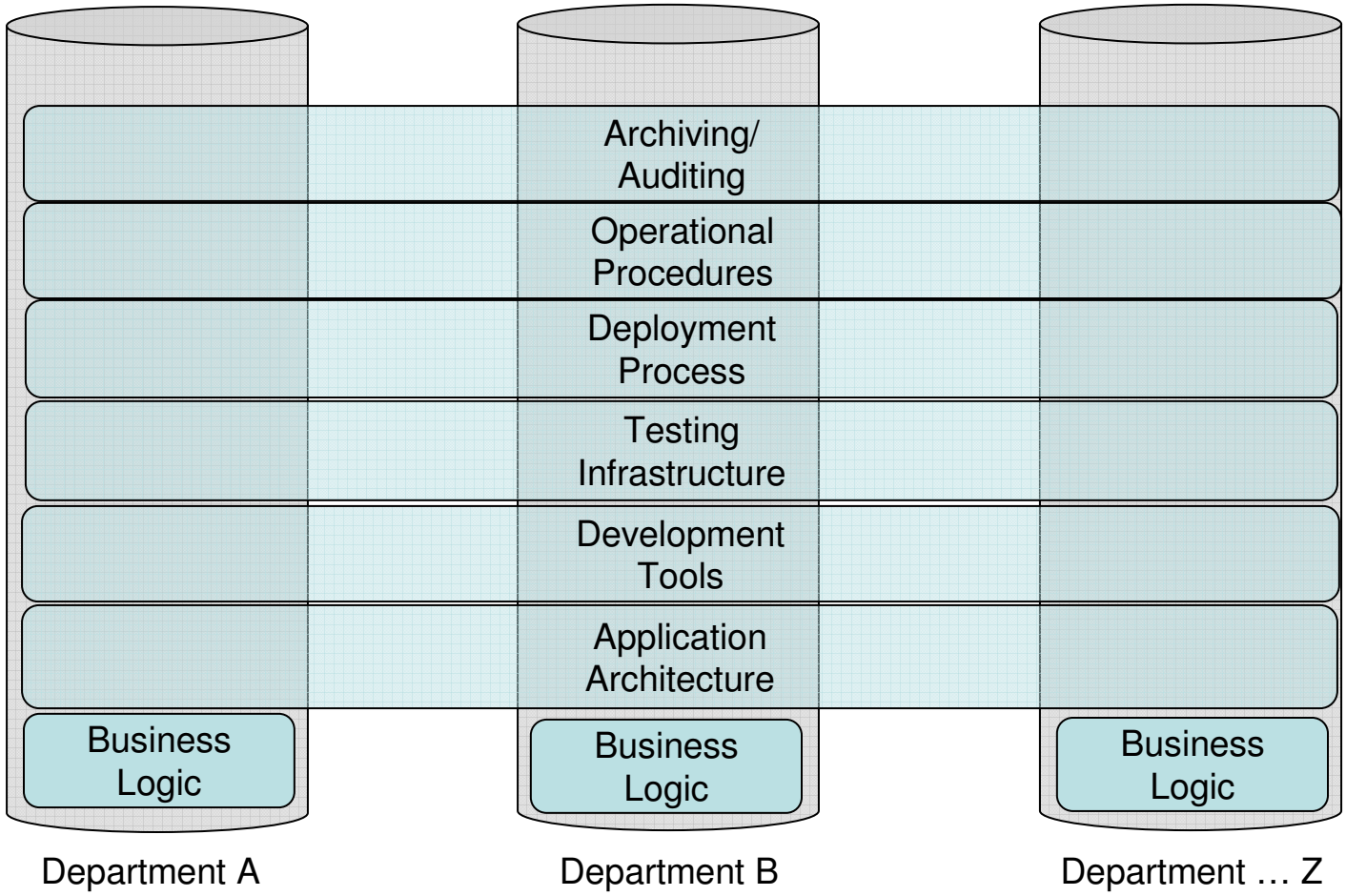
Modern Batch…Towards "Unified Batch Architecture"

- Today: Batch processing systems exist in silos



Modern Batch…Towards "Unified Batch Architecture"

- Tomorrow: Common Infrastructure for hosting and executing batch applications



- “Unified Batch Architecture” is the vision…
  “Batch Modernization” is a hurdle along the way

Exploring Batch “Modernization”

Why Modernize?

What's my motivation?

$$$ - three C's of the IT Budget: cost, cost, cost

- **Cost #1 – Infrastructure**
  - Pressure to reduce operational costs
- **Cost #2 – Skills**
  - Development resource drawn from shrinking pool.
- **Cost #3 – Competitiveness**
  - Failure to seize new opportunities due to lack of agility.

Not for everybody !!

Satisfying the Requirements: Why Java?

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

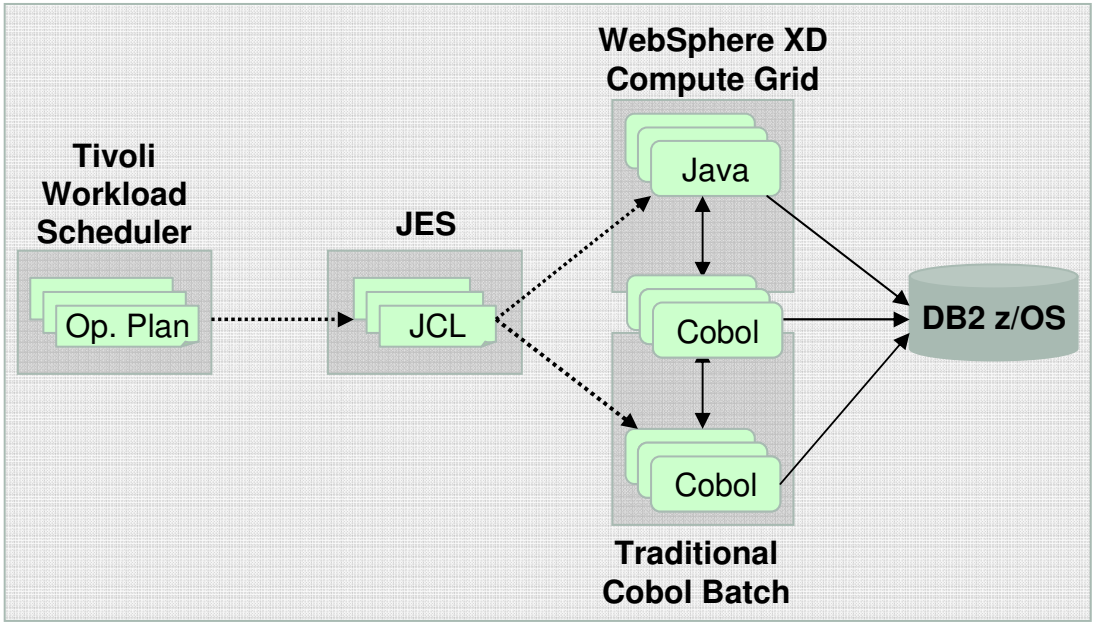
Swiss Re

Approach

Current Status

We plan to go productive with the first java batch on XD in summer 2008. But there remains a lot to be done to make this possible (stability, integration, architecture)

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

To Summarize…

- Stamp out “Maverick” Batch
  - To adopt a “Unified Batch Architecture” that encompasses all platforms and J2EE vendors
  - Eliminate “Maverick” Batch infrastructures, replace with common runtime infrastructure and application architecture
  - Governed by common operational procedures

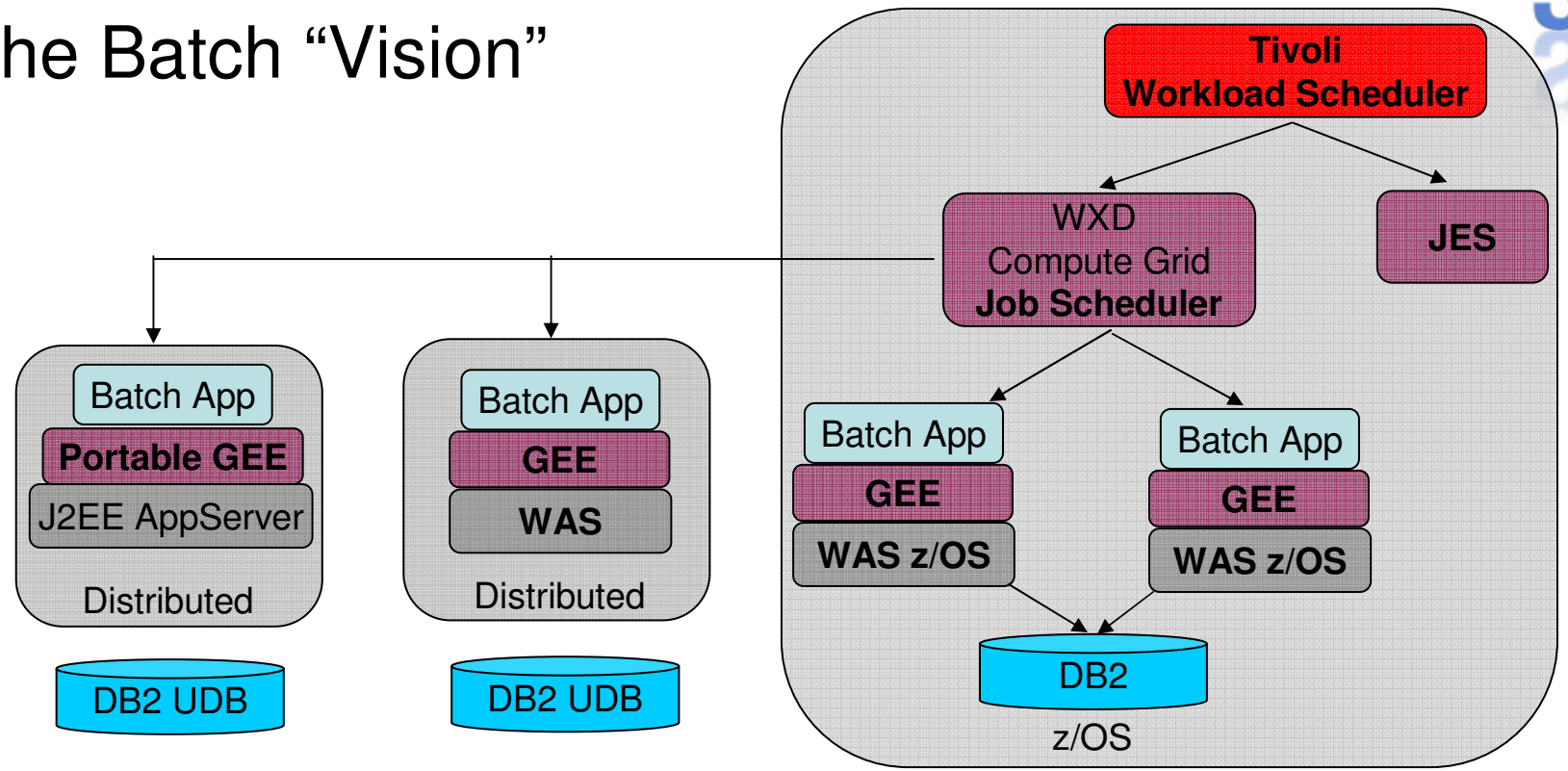
- Enterprise Batch Modernization
  - Facilitates the transformation of existing COBOL/PLX/C/C++ batch to java
  - Focus should be on developing ***NEW*** batch applications in modern languages, then ***assessing*** if existing batch should be transformed.

Grand Batch Strategy…

- Two Key Strategic Objectives: ***Ubiquity*** and ***Integration***
- ***Ubiquity:***
  - Batch applications should transcend J2EE vendor and platform
  - Application Placement should be dictated by the ***location of its data***.
  - Compute Grid Batch Containers should run ***everywhere***

- ***Integration:***
  - Existing infrastructure and operational procedures should be embraced and leveraged
  - Differentiate from our competitors through value-added integration
  - Integrate with:
    - The operating system (z/OS specifically)
    - The enterprise scheduler
    - The JVM
    - The WebSphere Product Family
    - Etc…..

The Batch "Vision"

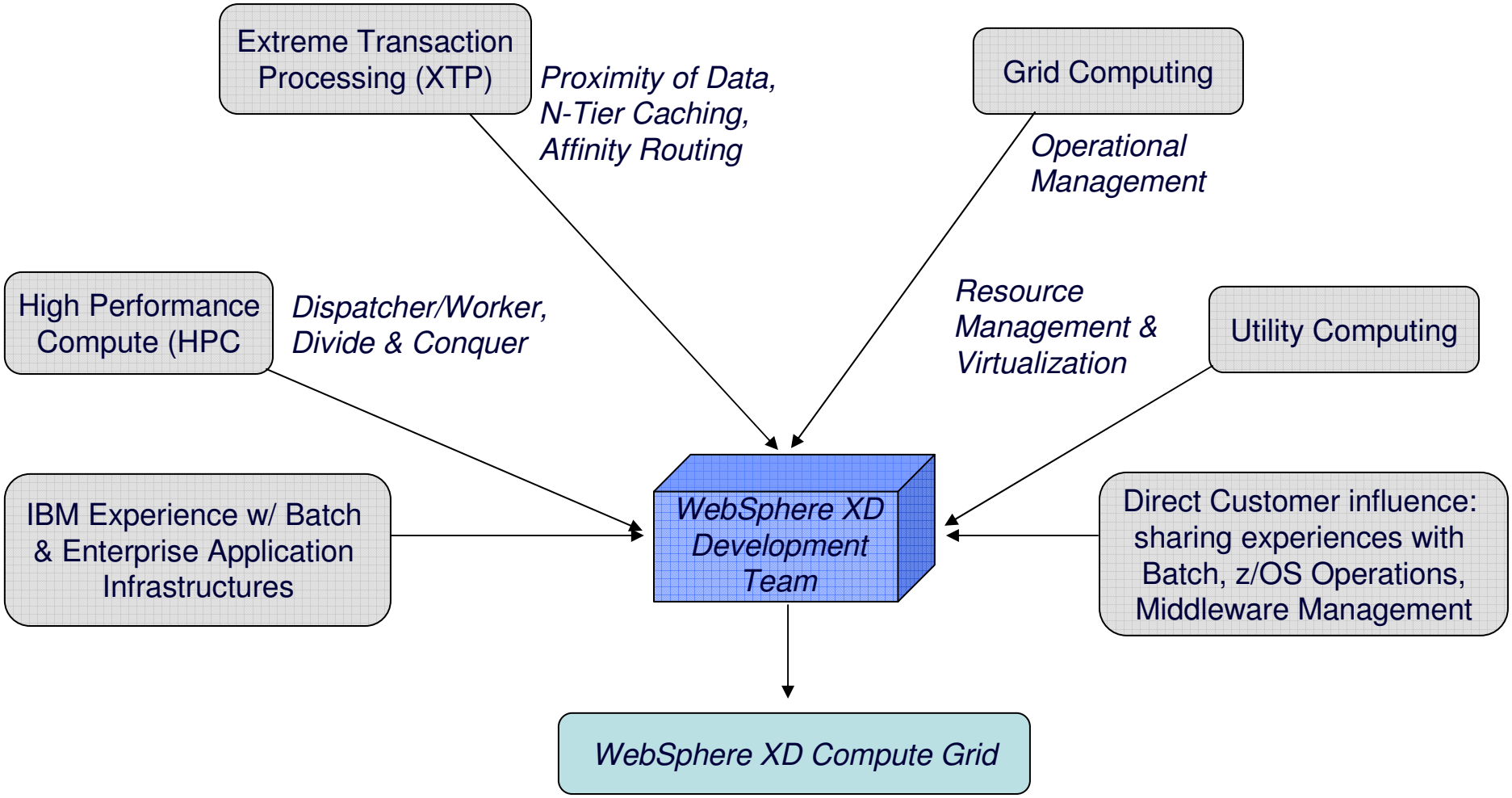


- **Portable Batch applications** across platforms and J2EE vendors
- Location of the data dictates the placement of the batch application
- Flexible programming model, will host Spring Batch, JZOS, Compute Grid apps
- Centrally managed by your enterprise scheduler
- z/OS operational procedures manage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

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

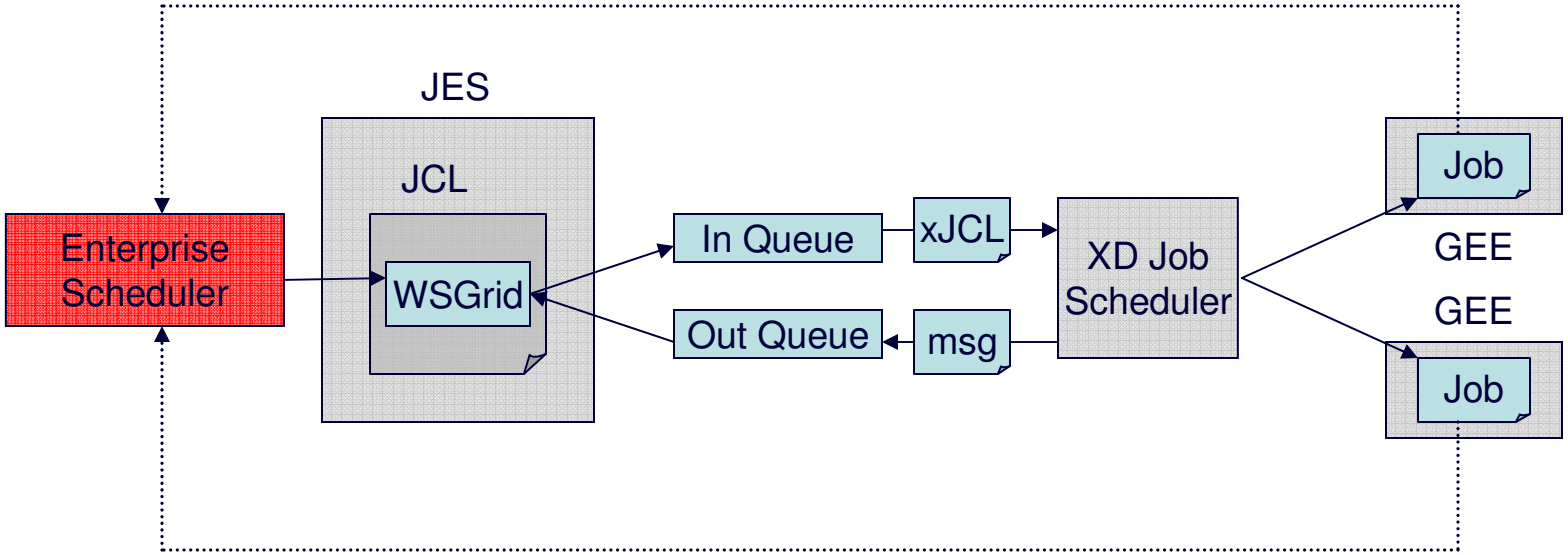
Origins of WebSphere XD Compute Grid



WebSphere XD Compute Grid and Traditional Batch Assets

- *Tivoli Workload Scheduler (Enterprise Schedulers Generally)*
- *JZOS*

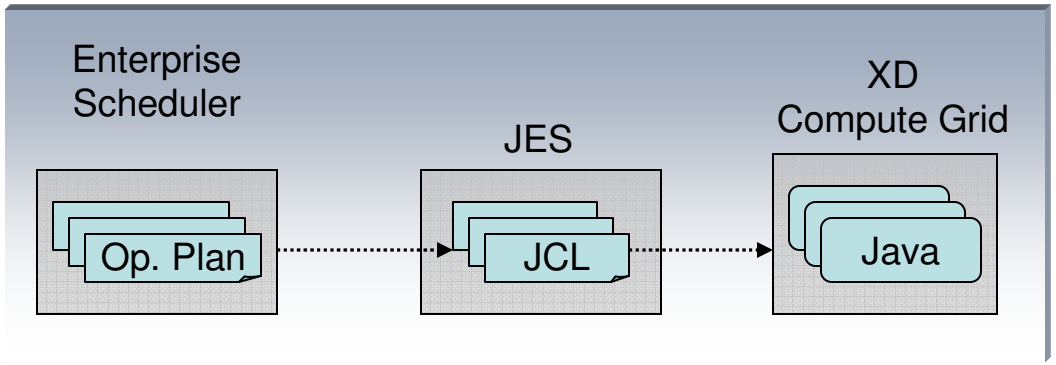
Enterprise Schedulers and Compute Grid



- *Central enterprise scheduler (TWS, etc) !! Compute Grid is told what to execute.*
- *Jobs and commands are submitted from Enterprise Scheduler to CG via WSGRID*
- *Jobs can dynamically schedule to Enterprise Scheduler (TWS) via EJB interface*

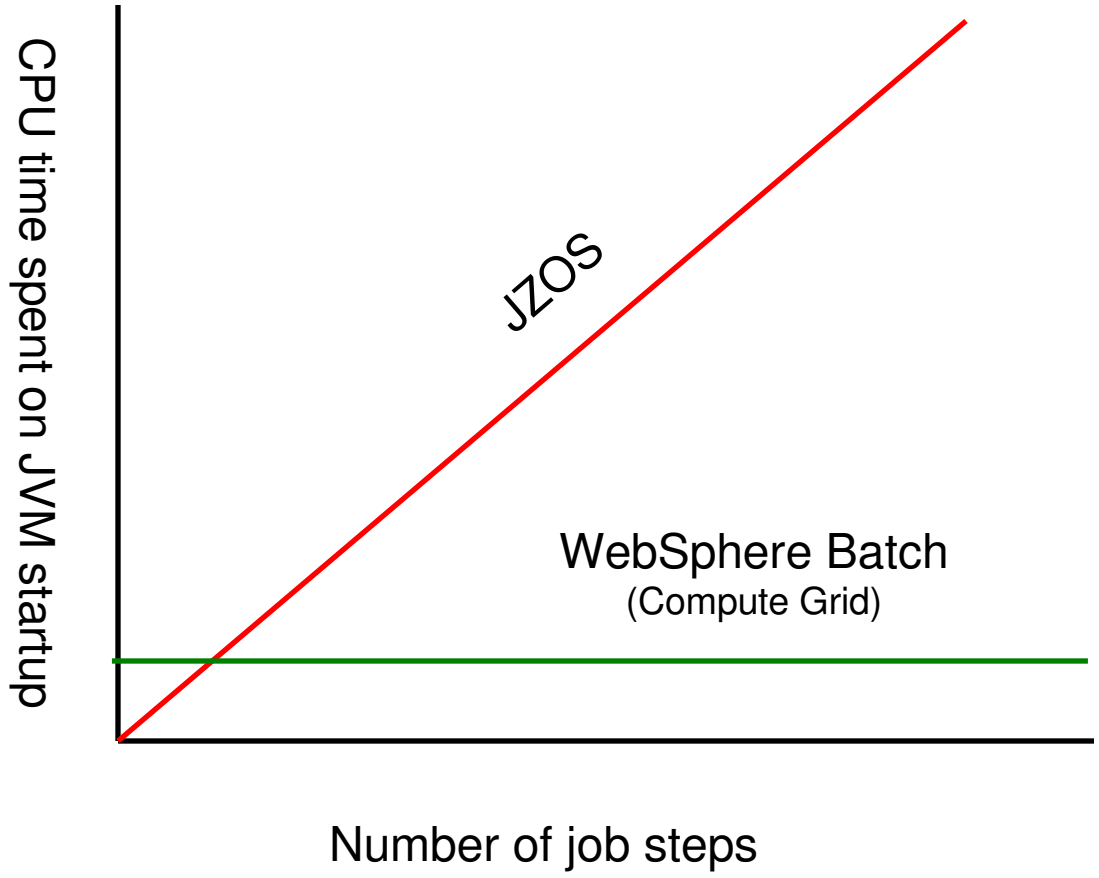
XD Compute Grid and Enterprise Schedulers

Role of Enterprise Schedulers

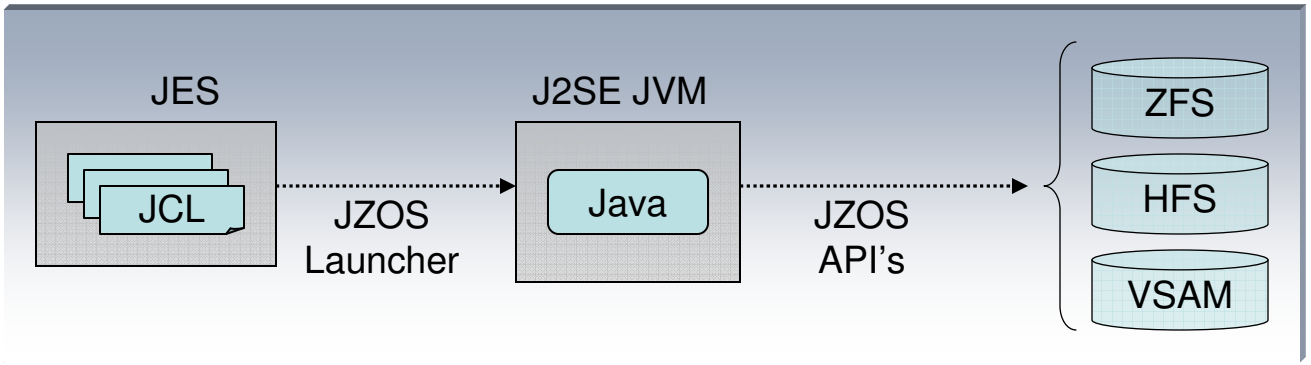


- Connector is the bridge between the enterprise schedule and Compute Grid
- On z/OS, JES is used as a proxy, pro's and con's for this.
- On z/OS, High-performance connector is in progress
- On Distributed (and z/OS for now), connector is a java-based JMS client

Mainframe Java Batch – Relative Base-line Cost



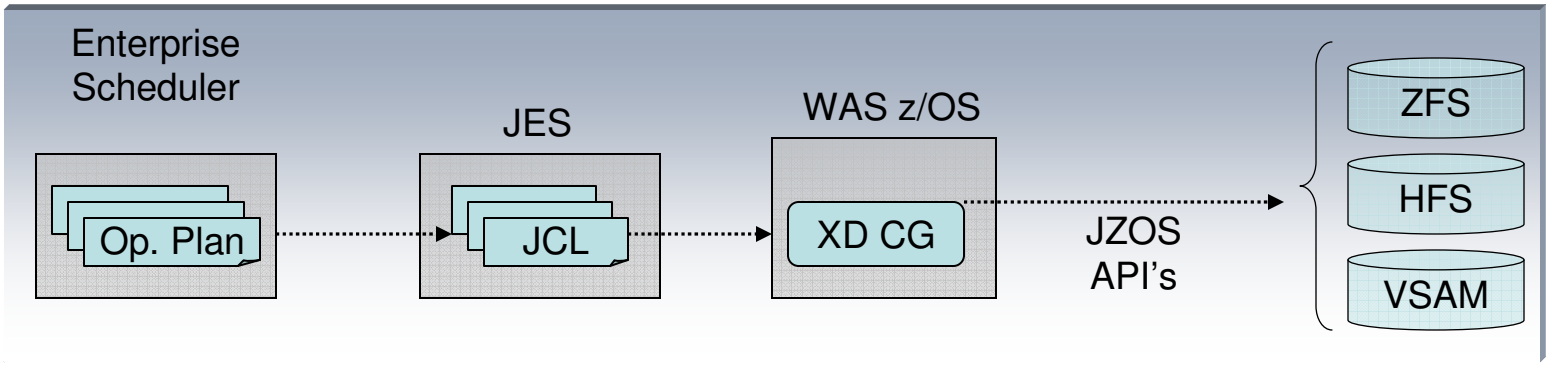
Positioning XD Compute Grid- Role of JZOS



- JZos delivers 2 technologies:
  1. ***JZOS Launcher-*** *s*eamless 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

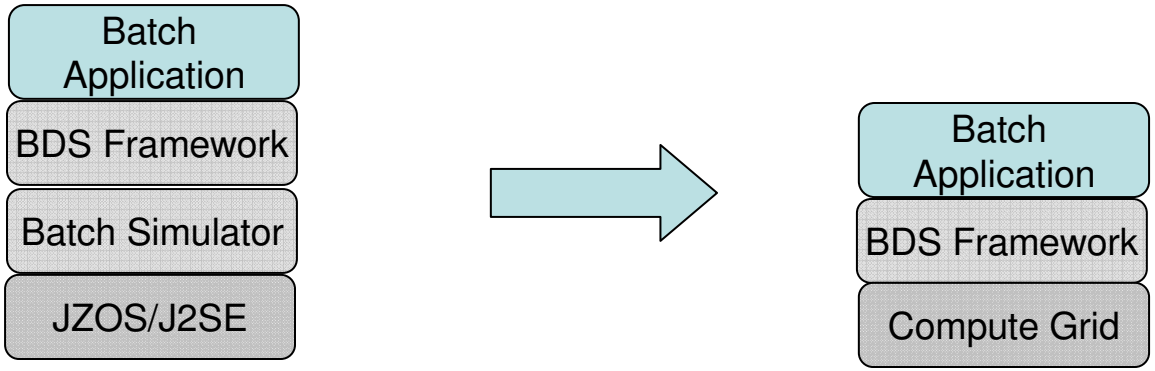
- ***JZOS launcher*** not efficient for 1000's of batch jobs to be run within a batch window
  - J2SE JVM has no:
    - security, transaction, or connection management
    - checkpoint or restart facility for batch jobs
    - inherent high availability, or other WAS z/OS qualities of service
  - JVM is not persistent or reusable.

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



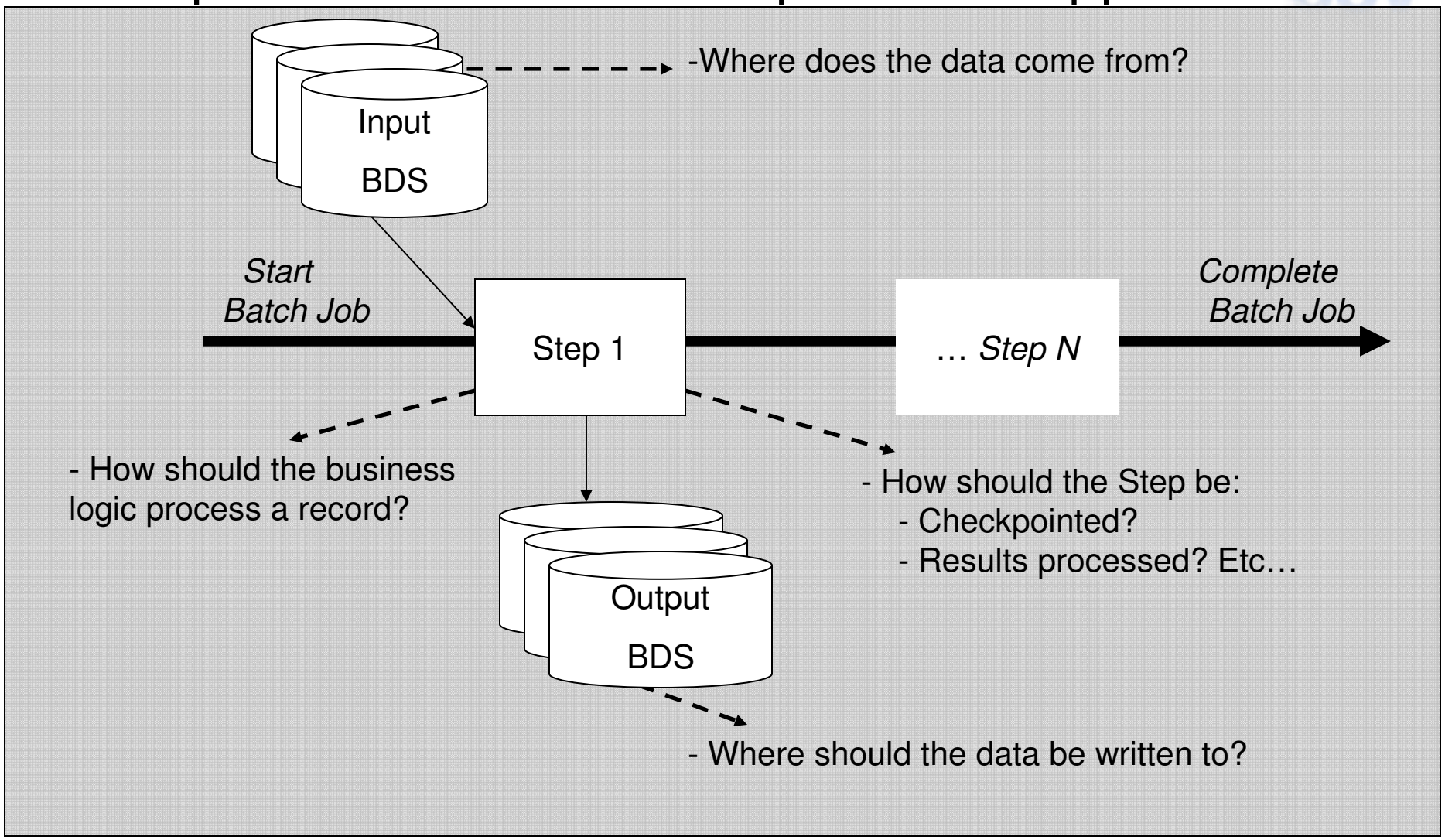
-Start with JZOS or J2SE-based Java batch infrastructure

-Grow into Compute Grid-based Java batch infrastructure

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

Building XD Compute Grid Applications

Components of an XD Compute Grid Application



Simple Programming Model

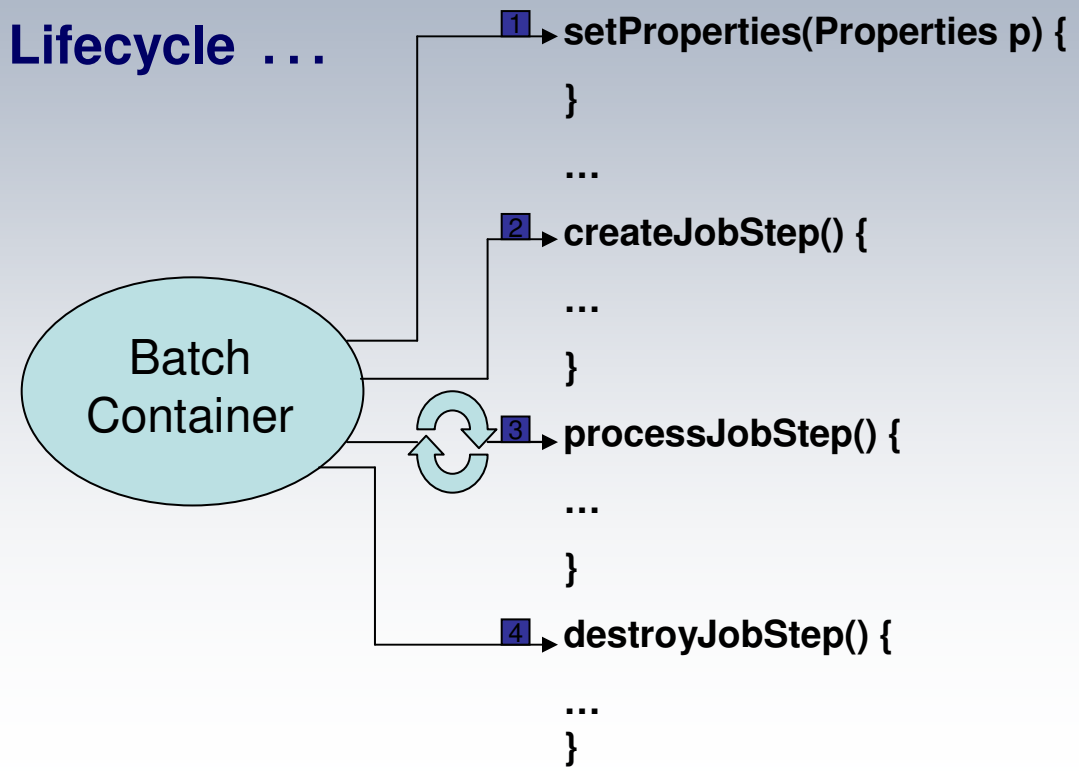
The anatomy of a transactional batch application – batch job step

XD Compute Grid makes it easy for developers to create transactional batch applications by allowing them to use a streamlined POJO model and to focus on business logic and not on the batch infrastructure

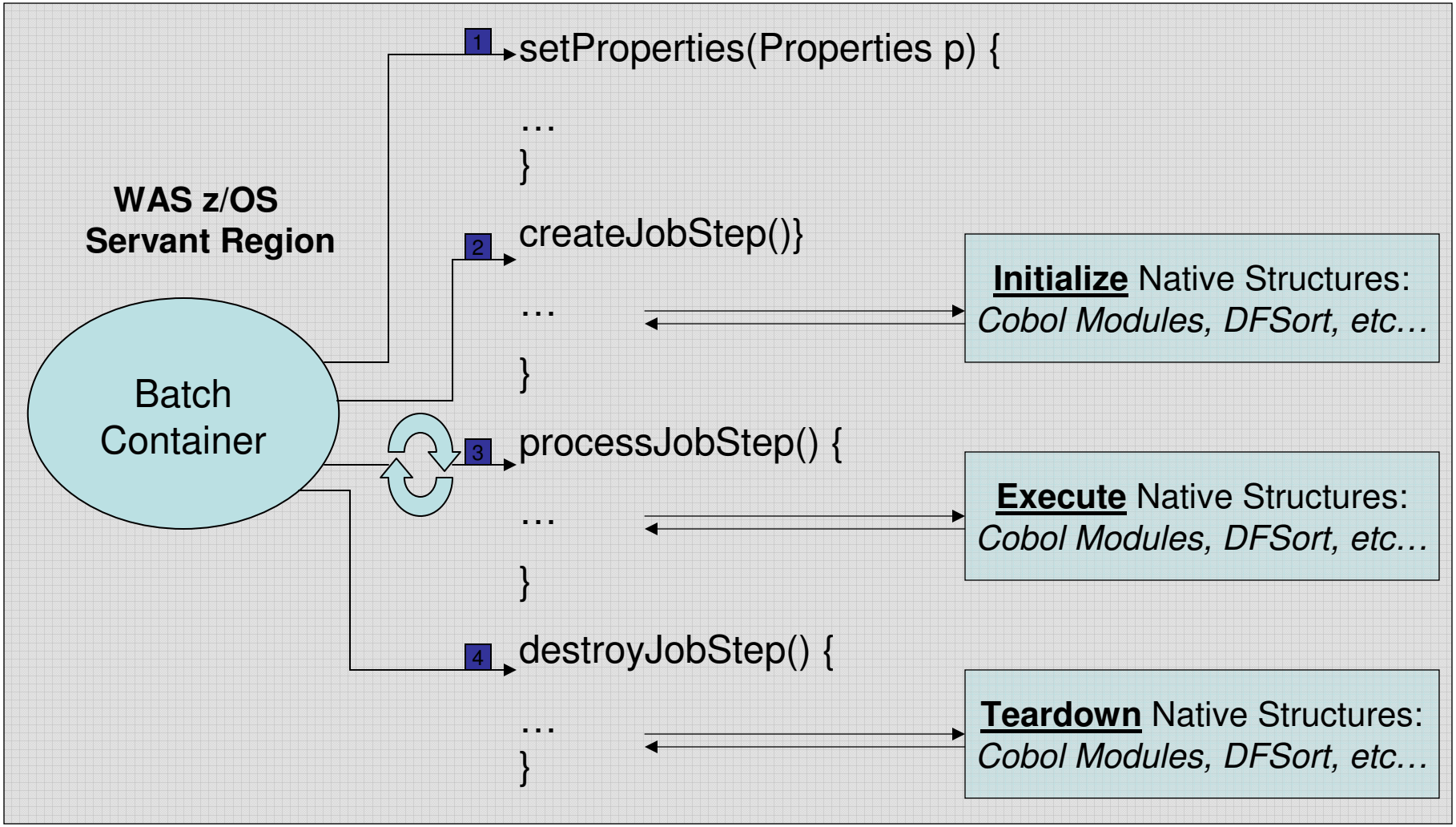
XD V6.0.1
- CMP EJB programming
- J2EE package & deployment
- Develop using RAD

XD V6.1
- POJO programming
- J2EE package and deployment
- Develop using Eclipse

XD V6.1 supports both models



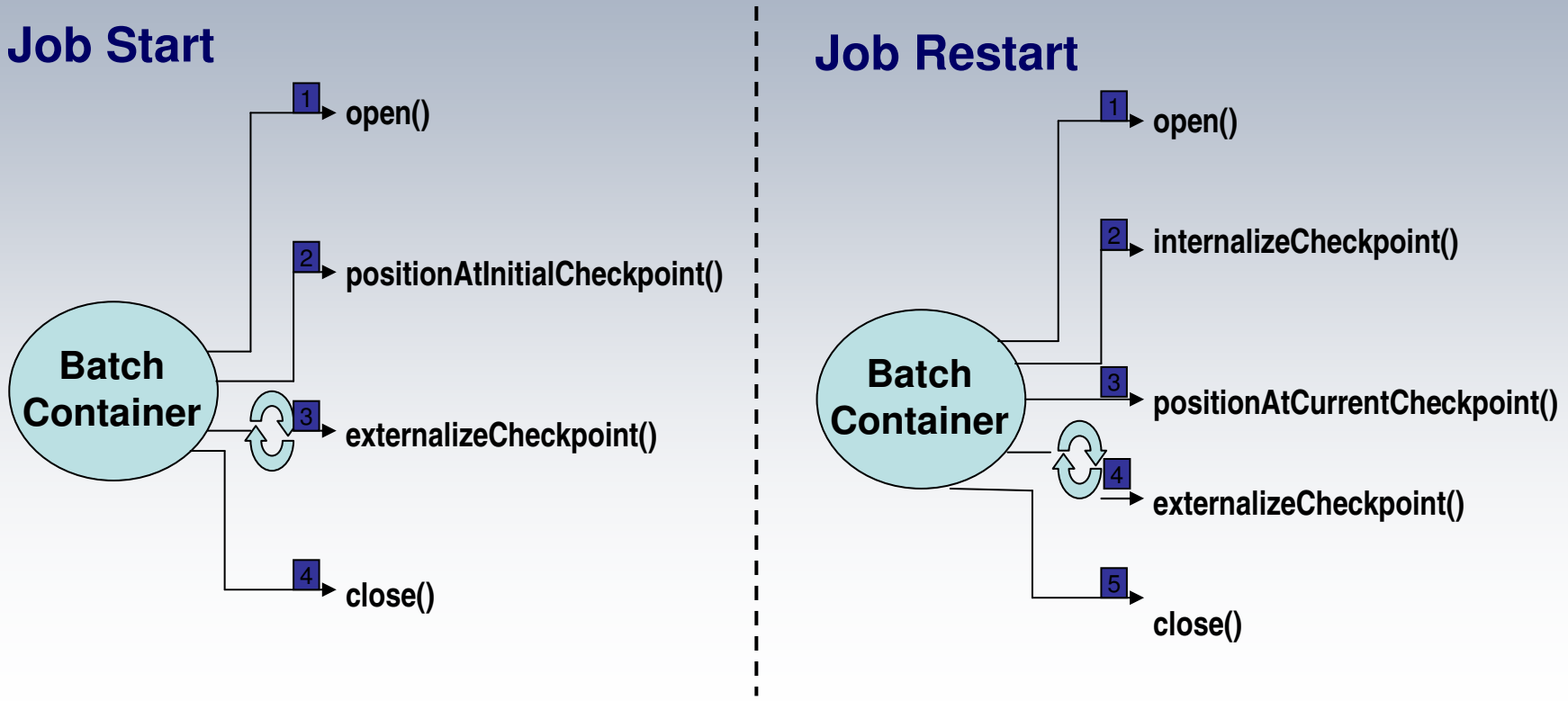
XD Batch and Traditional z/OS Interoperability



Simple Programming Model …

The anatomy of an transactional batch application – batch data stream

XD Compute Grid makes it easy for developers to encapsulate input/output data streams using POJOs that optionally support checkpoint/restart semantics.



Unified Development, Testing, Deployment Infrastructure

-Customer develops business service POJO's

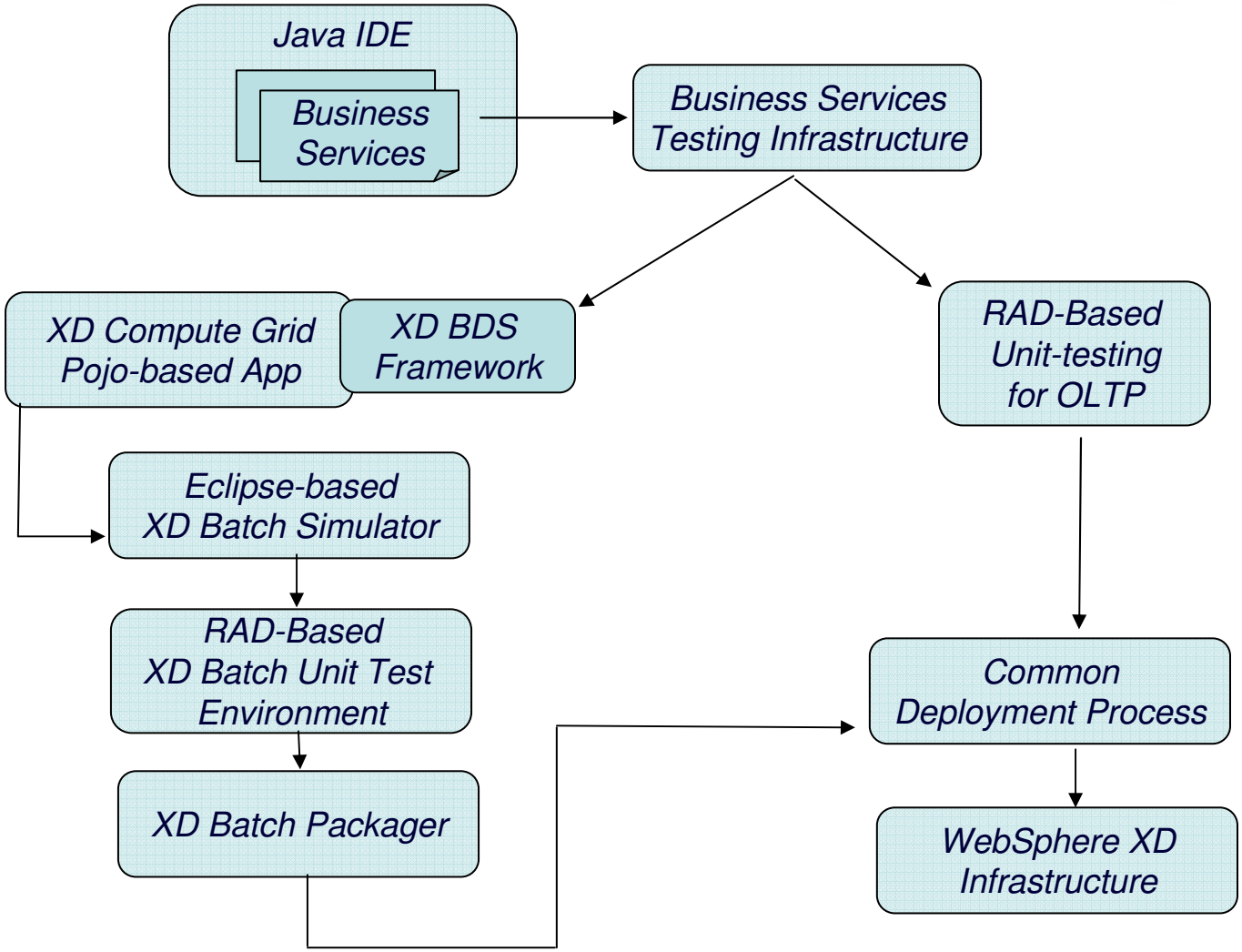
-Applications are assembled via Spring

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

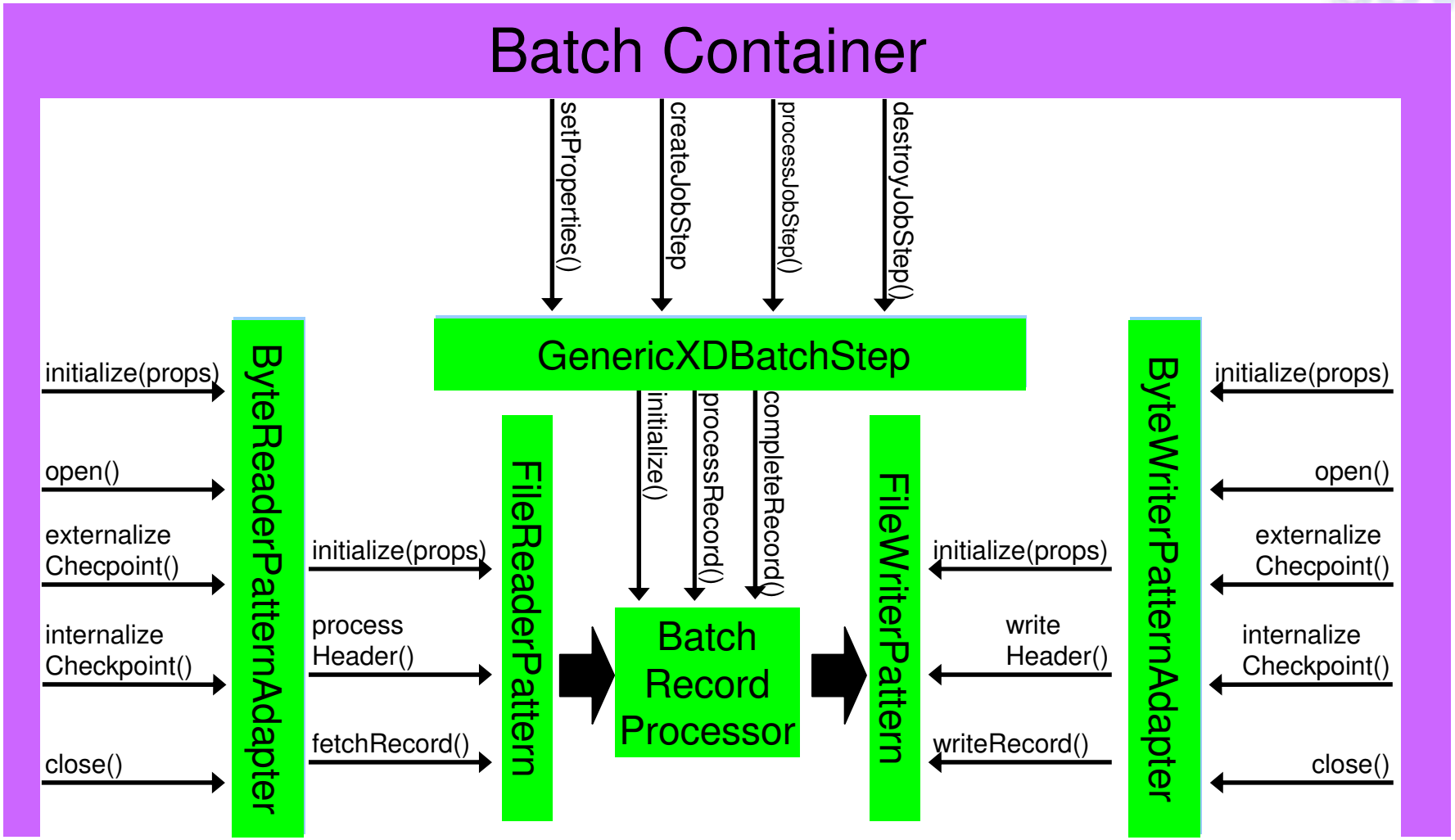
-XD Batch Simulator for development

-XD Batch Unit test environment for unit testing

-XD batch packager for .ear creation



The BDS Framework



WebSphere XD Compute Grid
BDS Framework Overview

- BDS Framework implements XD batch programming model for common use-cases:
  - Accessing MVS Datasets, Databases, files, JDBC Batching
  - Provides all of the restart logic specific to XD Batch programming model
- Customer's focus on business logic by implementing light-weight pattern interfaces; doesn't need to learn or understand the details of the XD Batch programming model
- Enables XD Batch experts to implement best-practices patterns under the covers
- XD BDS Framework owned and maintained by IBM; will be reused across customer implementations to provide stable integration point for business logic.

```
package com.ibm.websphere.batch.devframework.datastreams.patternadapter;

import java.sql.PreparedStatement;

public interface JDBCWriterPattern {

    public void initialize(Properties props);
    public String getSQLQuery();
    public PreparedStatement writeRecord(PreparedStatement pstmt, Object record);
}
```

Development Tooling Story for WebSphere XD Compute Grid

- 1. The **Batch Datastream (BDS) Framework.** This is a development toolkit that implements the Compute Grid interfaces for accessing common input and output sources such as files, databases, and so on. 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.

Conclusions….

Summarizing Compute Grid…

- **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.

Grand Strategy…

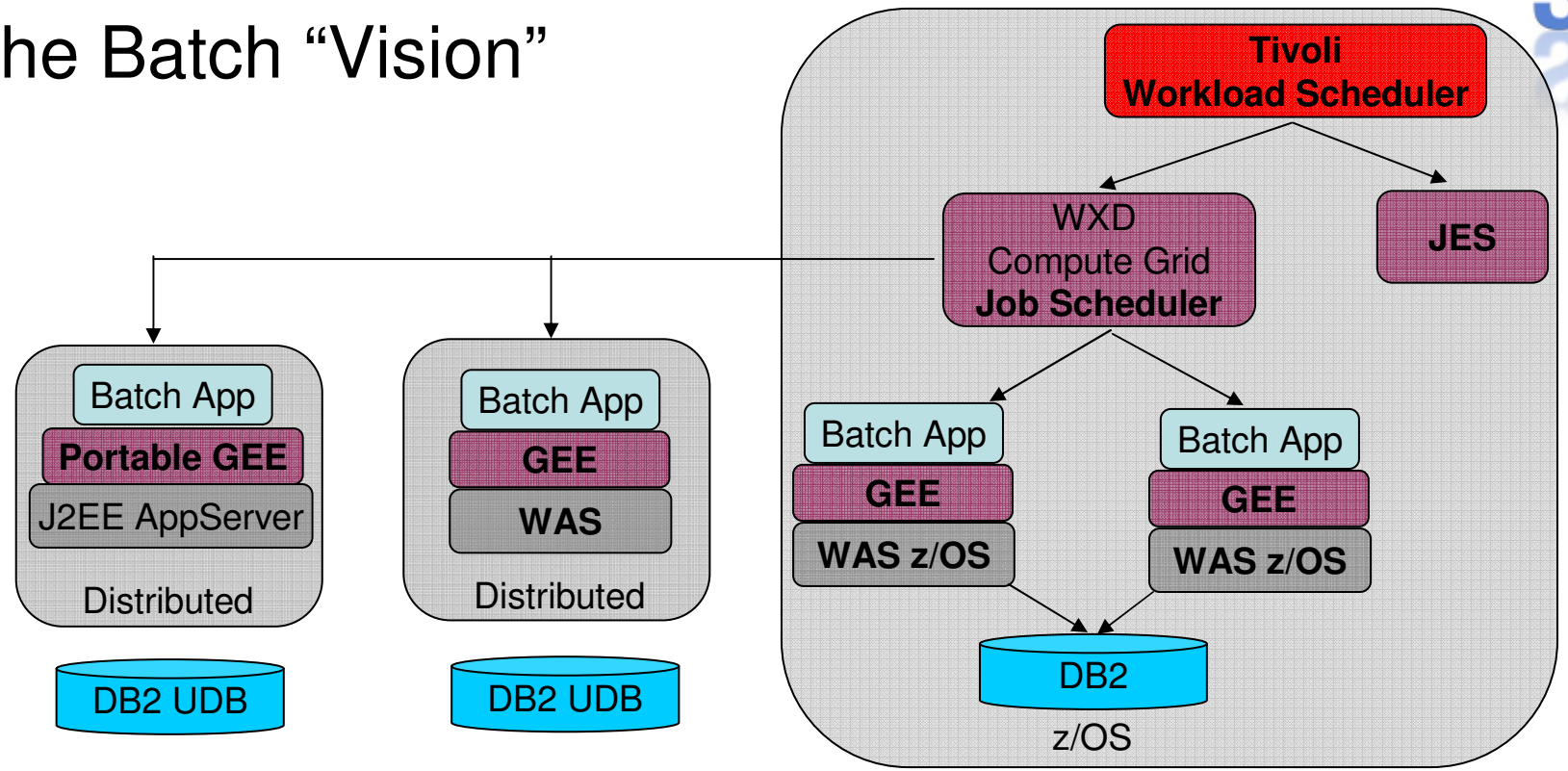
- Two Key Strategic Objectives: ***Ubiquity*** and ***Integration***
- ***Ubiquity:***
  - Batch applications should transcend J2EE vendor and platform
  - Application Placement should be dictated by the ***location of its data***.
  - Compute Grid Batch Containers should run ***everywhere***

- ***Integration:***
  - Existing infrastructure and operational procedures should be embraced and leveraged
  - Differentiate from our competitors through value-added integration
  - Integrate with:
    - The operating system (z/OS specifically)
    - The enterprise scheduler
    - The JVM
    - The WebSphere Product Family
    - Etc…..

Achieving our Vision…

- Step 1: Deliver a ***credible*** product that is ***consumable***
  - *SwissRe, customer partnerships, and Our **free** development tools*
  - *Starter packs en-route, always seeking ways to improve*
- Step 2: Add credibility through ***references*** & ***production customers***
  - *Compute Grid Architecture Board… made by customers, for customers*
  - *Several customers in pre-production… goal of 5 Impact '09 customer pitches*
- Step 3: Build an eco-system via ***business partners*** and ***ISV's***
  - *In Progress*
- Step 4: ***Differentiate*** through ***Integration***
  - *WebSphere Virtual Enterprise, z/OS, WebSphere z/OS, TWS*
  - *Always seeking ways to further integrate*
- Step 5: Take over the world

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

References

- WebSphere Extended Deployment Compute Grid ideal for handling mission-critical batch workloads
  http://www.ibm.com/developerworks/websphere/techjournal/0804_antani/0804_antani.html
- Enterprise Java Batch with Compute Grid WebCast
  http://www-306.ibm.com/software/os/systemz/telecon/nov15/
- WebSphere XD Technical Overview Podcast
  http://www.ibm.com/developerworks/podcast/dysmf/dysmf-2007-ep5txt.html?ca=dwpodcastall
- Java Batch Programming with XD Compute Grid
  http://www.ibm.com/developerworks/websphere/techjournal/0801_vignola/0801_vignola.html
- 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
- Compute Grid Trial Download
  http://www.ibm.com/developerworks/downloads/ws/wscg/learn.html?S_TACT=105AGX10&S_CMP=ART
- Compute Grid Wiki (product documentation)
  http://www.ibm.com/developerworks/wikis/display/xdcomputegrid/Home?S_TACT=105AGX10&S_CMP=ART

Smart SOA

2008 IMPACT

Questions & Answers

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.
Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM trademarks, see www.ibm.com/legal/copytrade.shtml
AIX, CICS, CICSPlex, DB2, DB2 Universal Database, i5/OS, IBM, the IBM logo, IMS, iSeries, Lotus, MQSeries, OMEGAMON, OS/390, Parallel Sysplex, pureXML, Rational, RACF, Redbooks, Sametime, Smart SOA, System i, System i5, System z , Tivoli, WebSphere, zSeries and z/OS.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.
Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
UNIX is a registered trademark of The Open Group in the United States and other countries.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Back up and reference

WebSphere Virtual Enterprise Details…

WebSphere XD

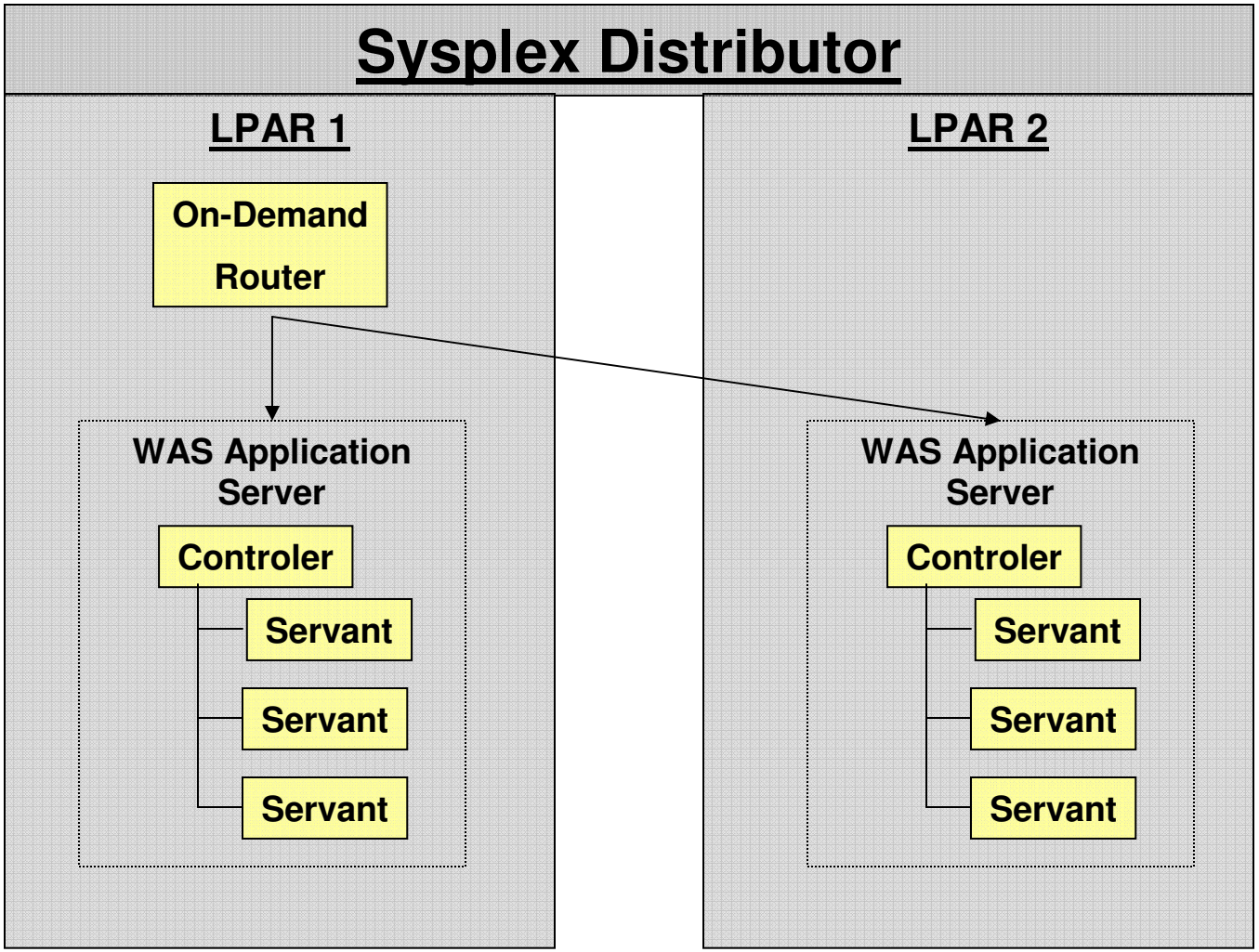
What is vertical and horizontal scaling?

Scaling Vertically

Scaling Horizontally

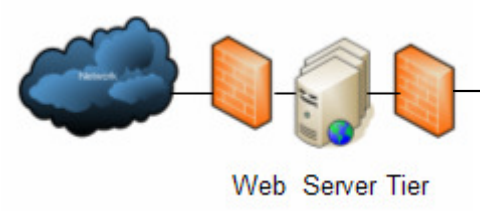
-Vertical implies scaling within the same 'box'.

- Horizontal implies scaling across 'boxes'



On Demand Router

- Classification
- Workload Balancing
- Capacity management
- Version-aware Application routing
- Storm drain avoidance
- Application lazy start



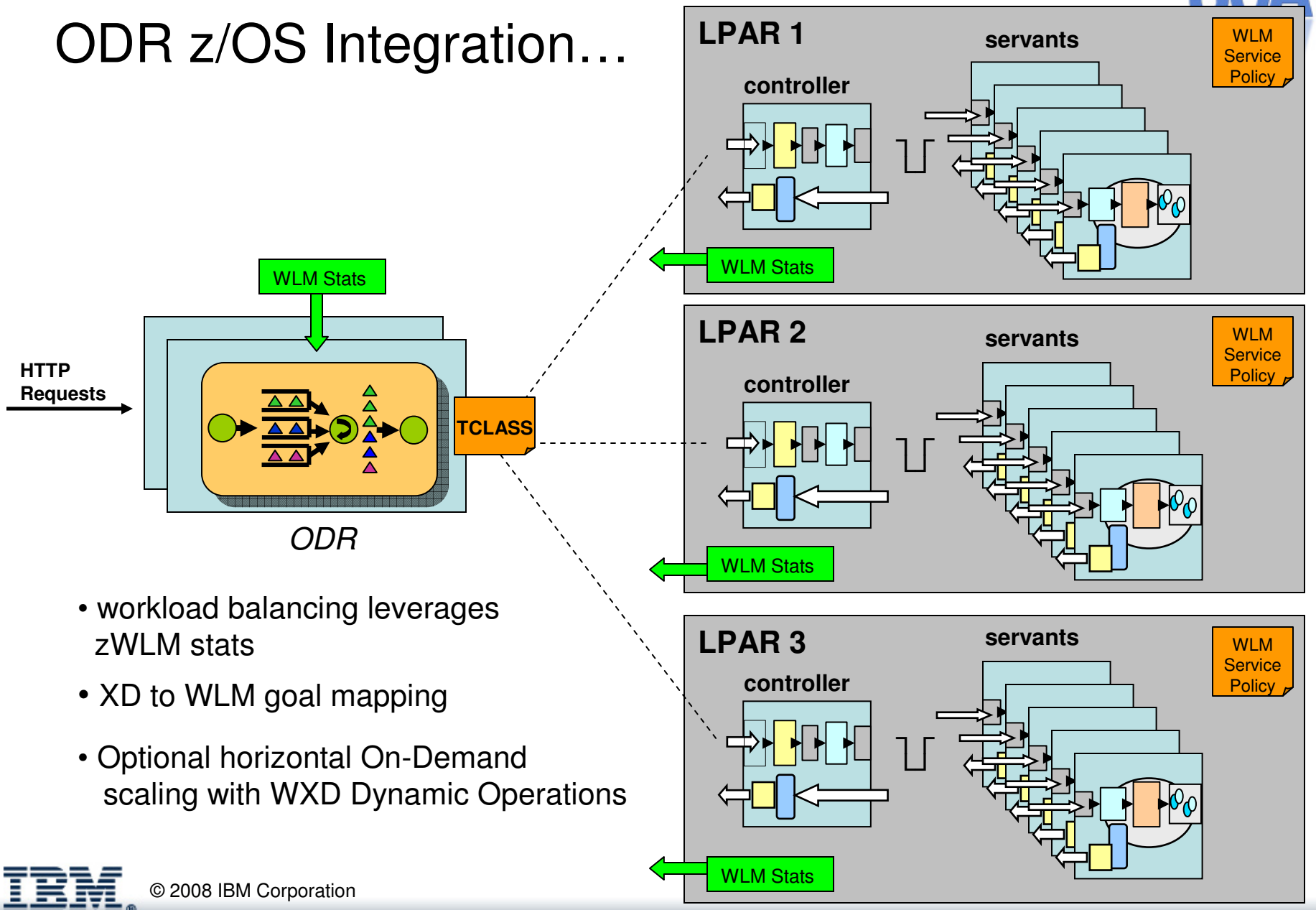
On Demand Router

Application Servers

Application Servers

Application Servers

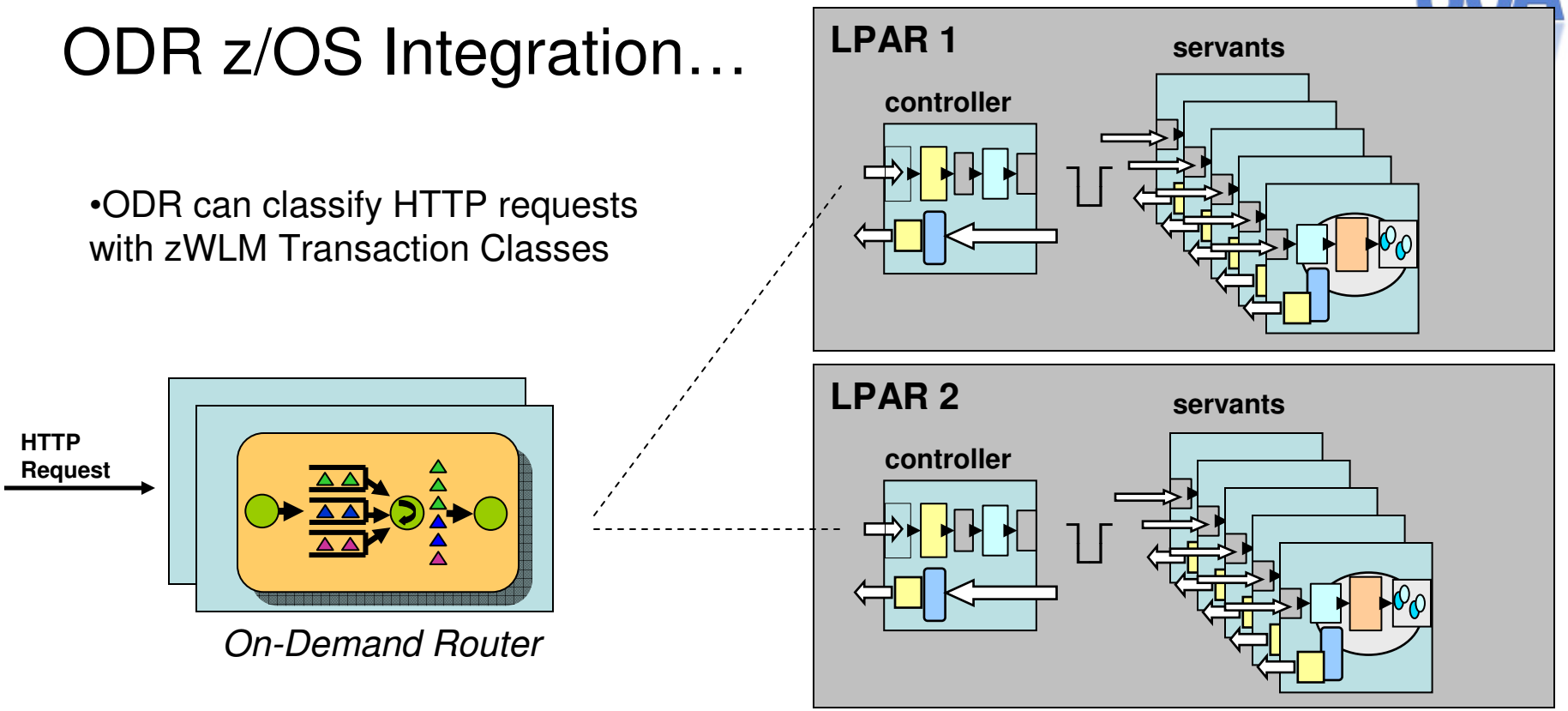
ODR z/OS Integration…



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

ODR z/OS Integration…

•ODR can classify HTTP requests with zWLM Transaction Classes



- Classification rules can be created to bind zWLM Transaction Classes to the HTTP request
- WAS z/OS Control Region will extract that TCLass and push it down to zWLM
- zWLM will route the work to the appropriate Servant Region

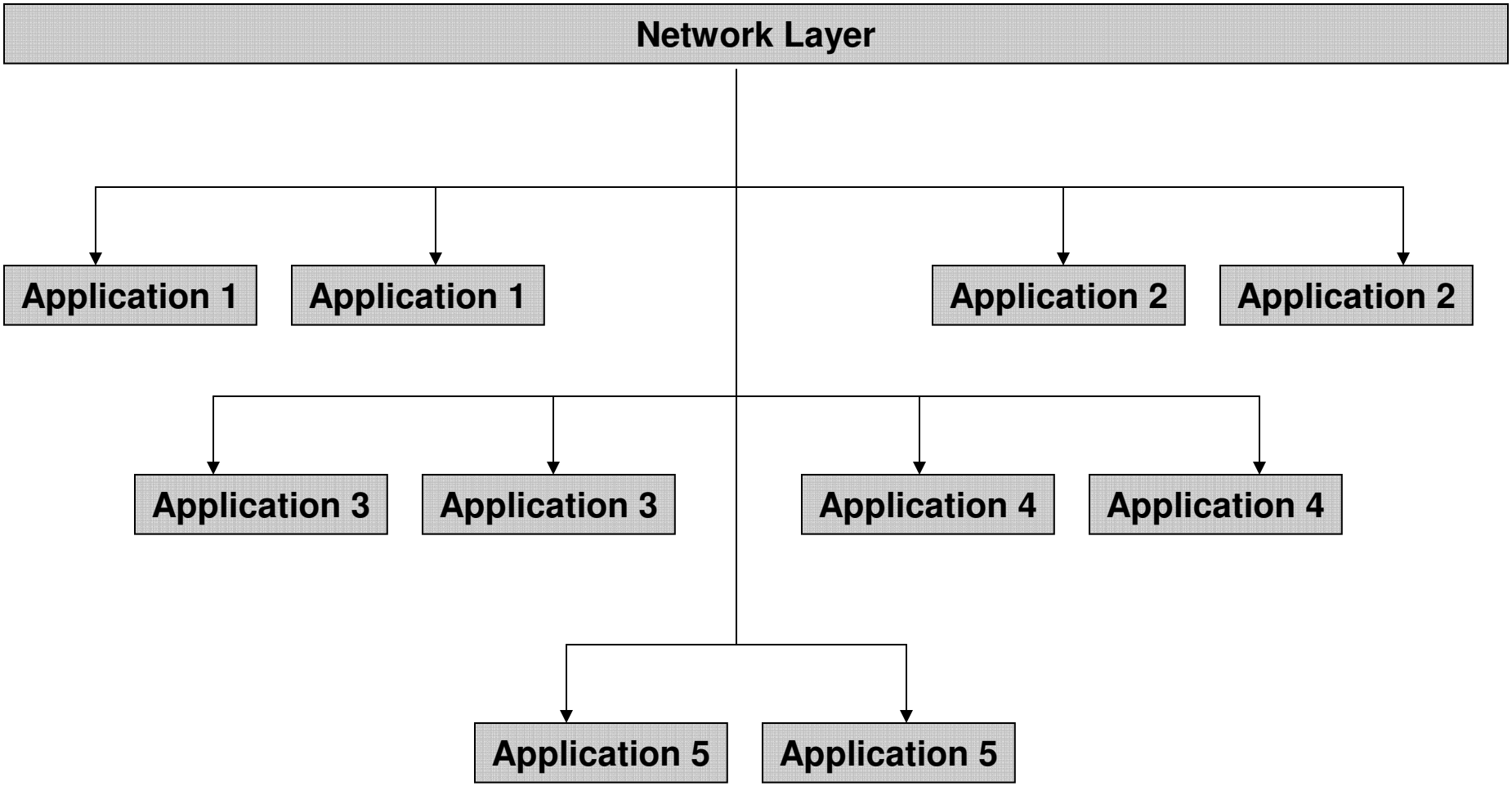
Dynamic Operations Overview

- Virtualized, policy-based, dynamic workload management
- Dynamic application placement
  - Enables starting and stopping server instances based on application load and user-defined goals
- On-Demand Router
  - Enhanced version of the Proxy Server
  - Controls request prioritization, flow, and routing in an Extended Deployment (XD) environment

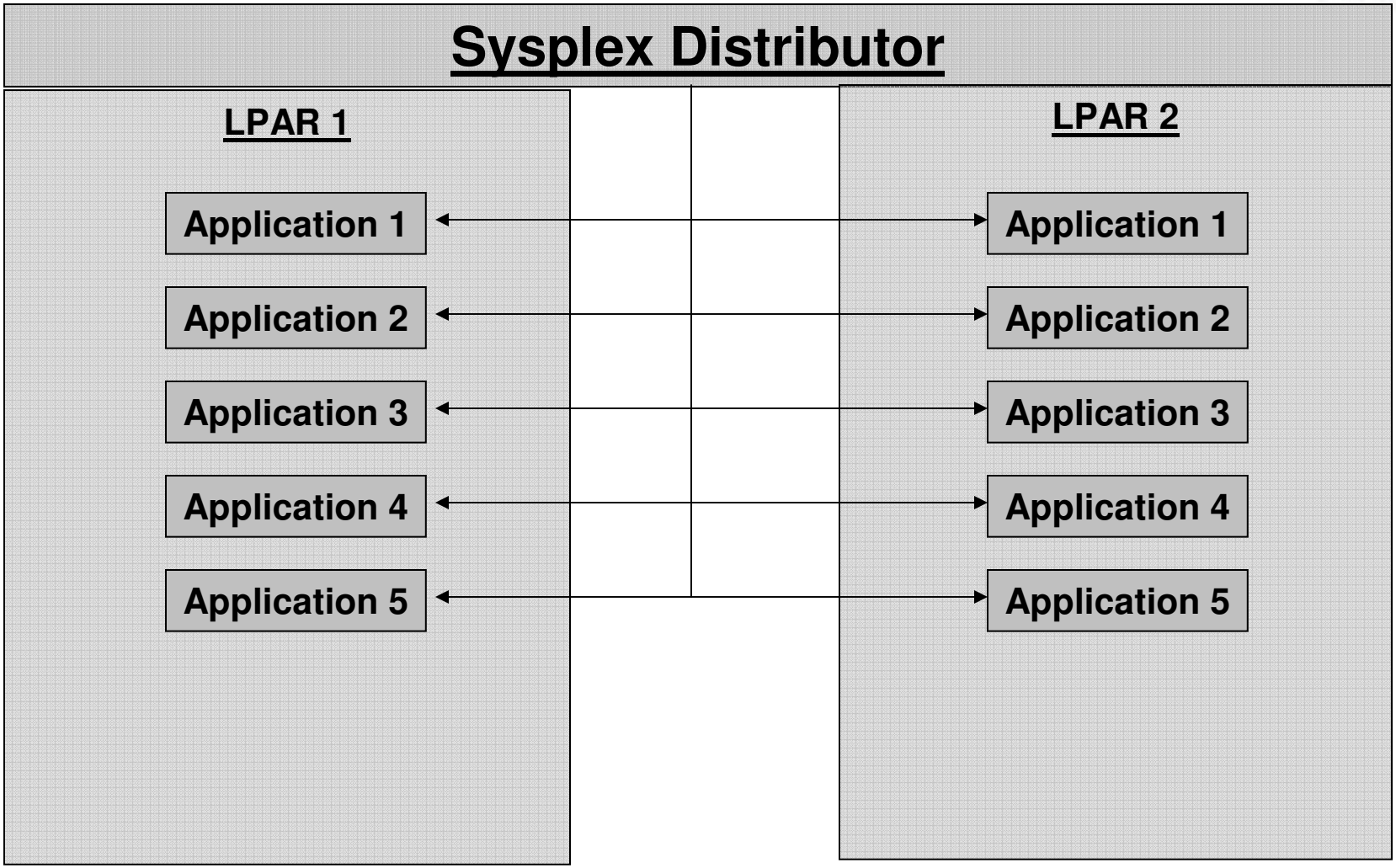
Scenario: Tying z/OS resource management with XD

- Customer has 5 applications
- Each application must be run in its own server
- Each application must be highly-available

Solving this problem with WebSphere Distributed



Solving this problem with WebSphere z/OS

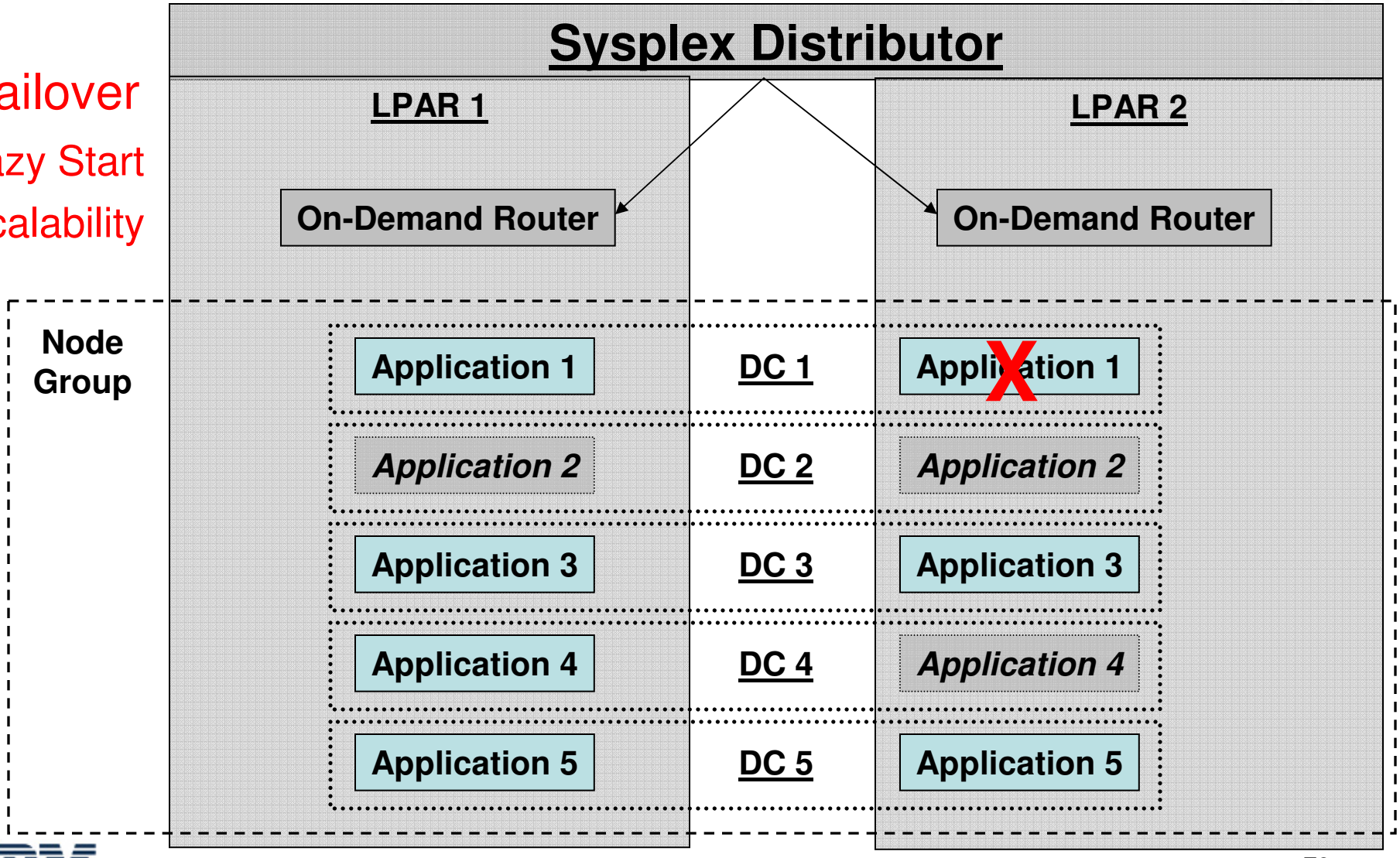


Solving this problem with WebSphere XD z/OS

Failover

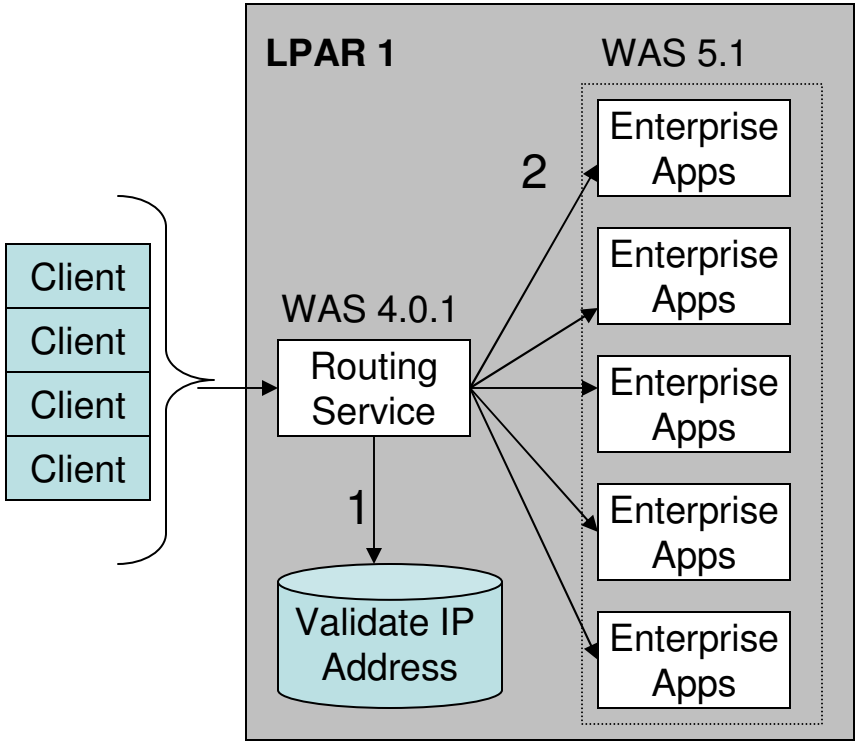
Lazy Start

Scalability



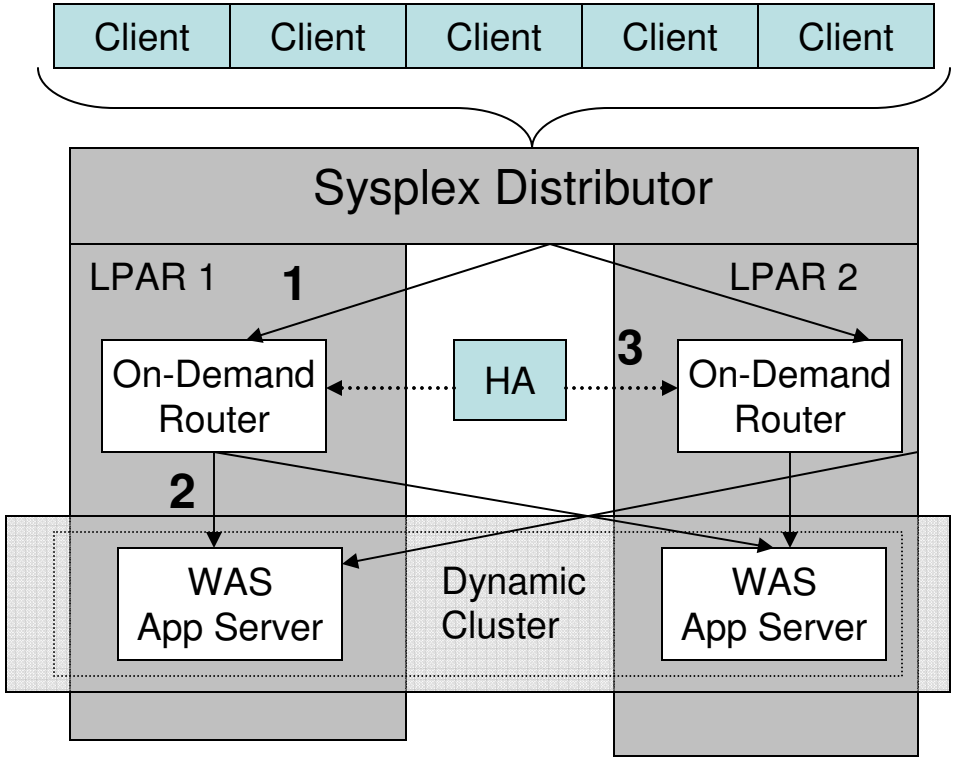
Routing, High Availability & Application Deployment

Infrastructure before WebSphere XD



- Home-grown routing
- Manual application upgrades
- Brittle architecture

Infrastructure after WebSphere XD

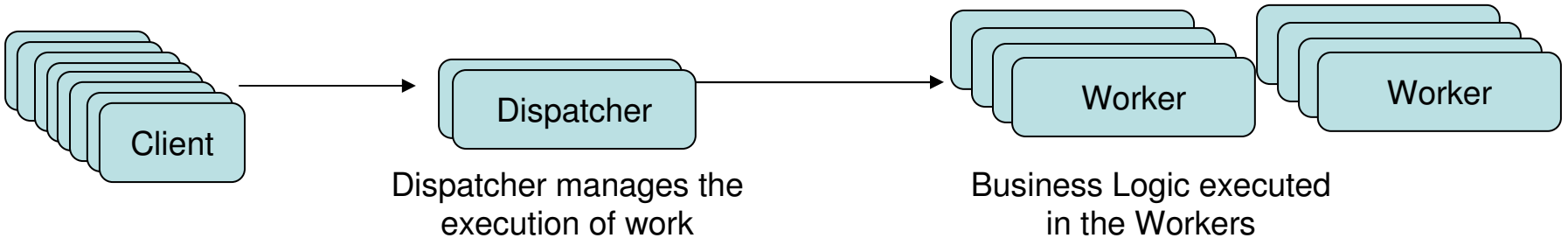


- ✓ Highly Available
- ✓ Streamlined application deployment

Compute Grid Influences…

XD Compute Grid and HPC

Scalability via the Dispatcher/Worker Pattern

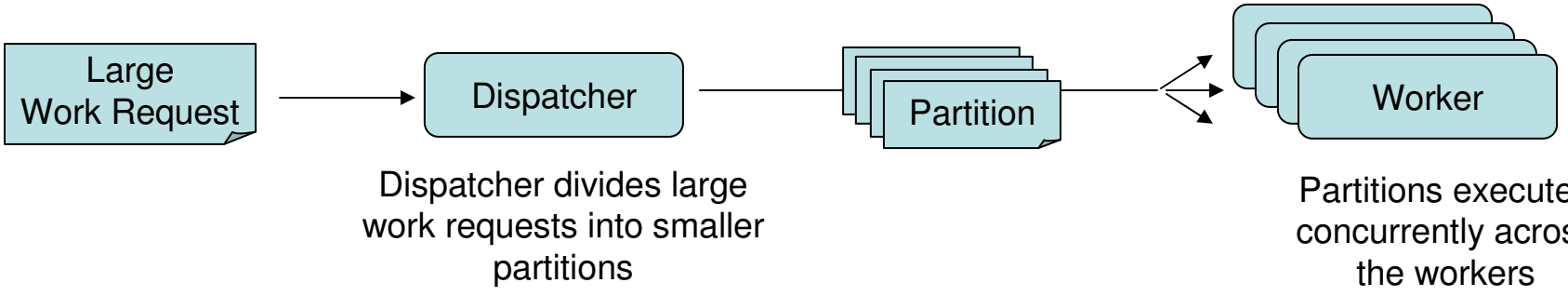


-Workers can scale dynamically to meet the current workload demands

-Dispatcher manages workload execution across the collection of workers

XD Compute Grid and HPC

Performance with Divide & Conquer



- When sequentially processing a large work request:

Elapsed Time = f(Work Request Size); large requests processed sequentially will have longer elapsed times

- When applying Divide & Conquer to process a large work request:

Elapsed Time =f(partition size); partitions are executed concurrently across the collection of workers

- Ideally 50 smaller partitions is 50x faster than 1 large work request.

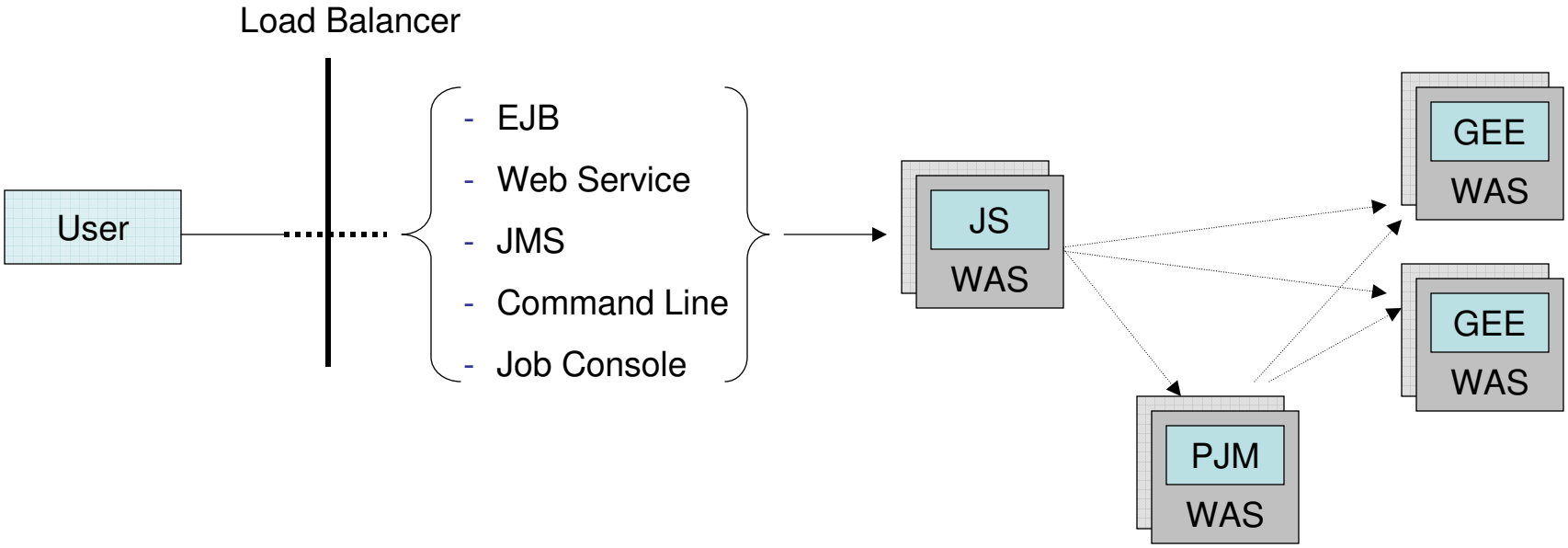
XD Compute Grid Components

- 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

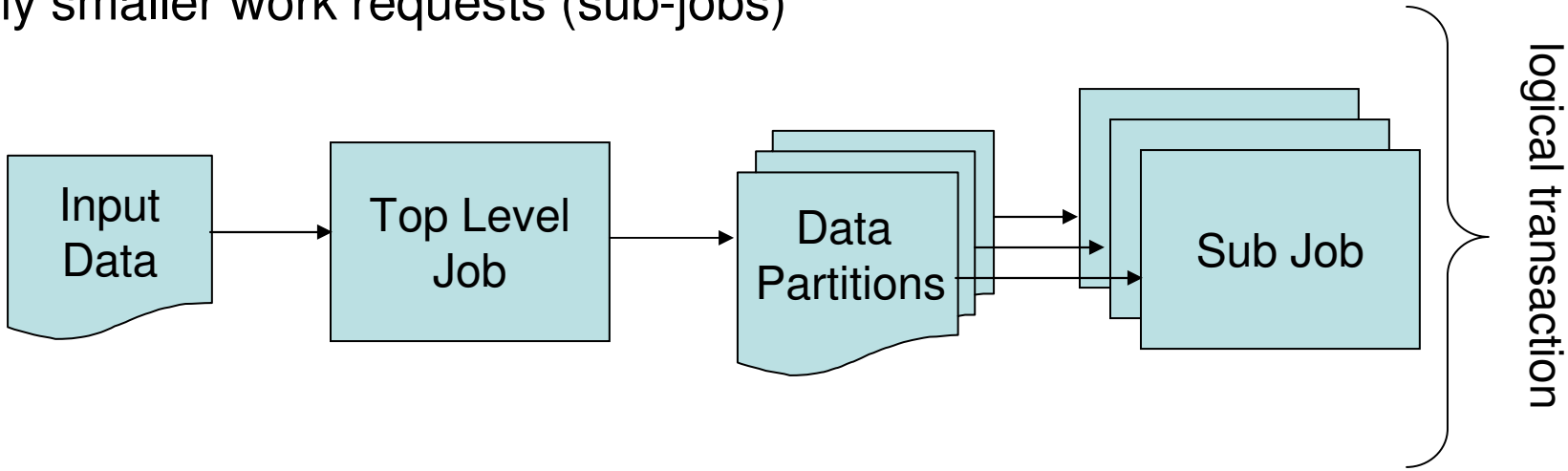
- Grid Endpoints (**GEE**)
  - Executes the actual business logic of the batch job

XD Compute Grid Components



Understanding the Model

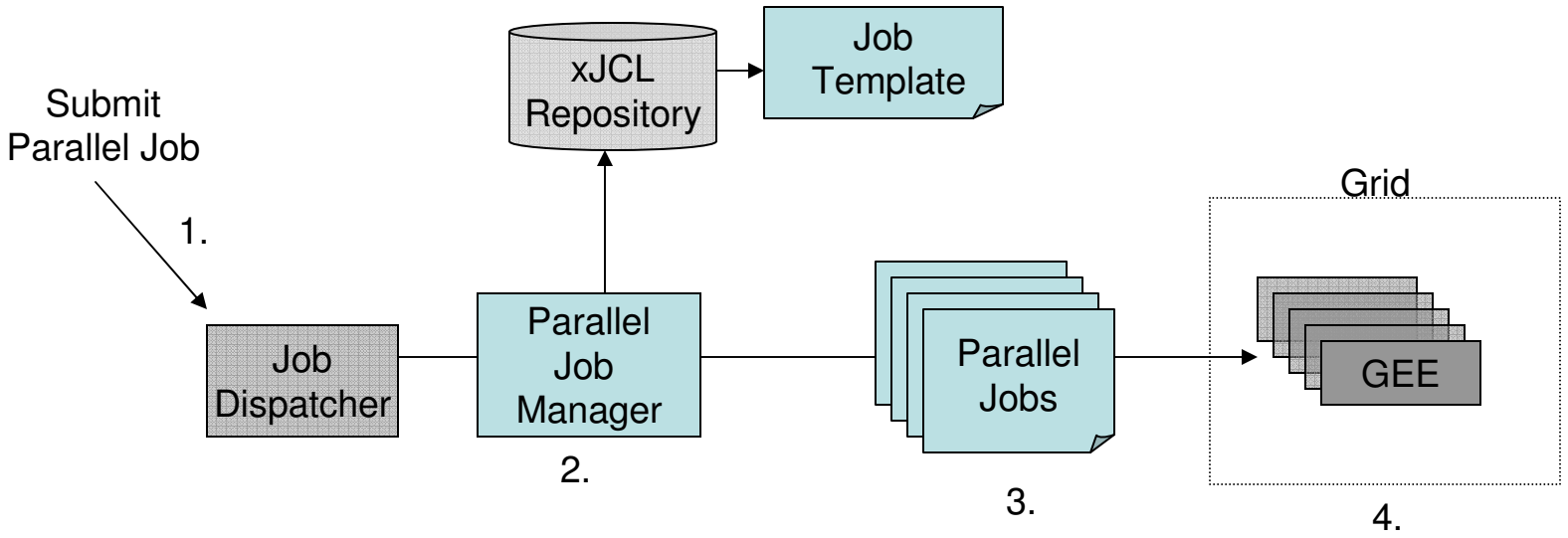
- Parallel Job Manager (PJM) decomposes a large work request into many smaller work requests (sub-jobs)



- PJM then provides operational control over the sub-jobs executing across the job endpoints – note sub-jobs are clones

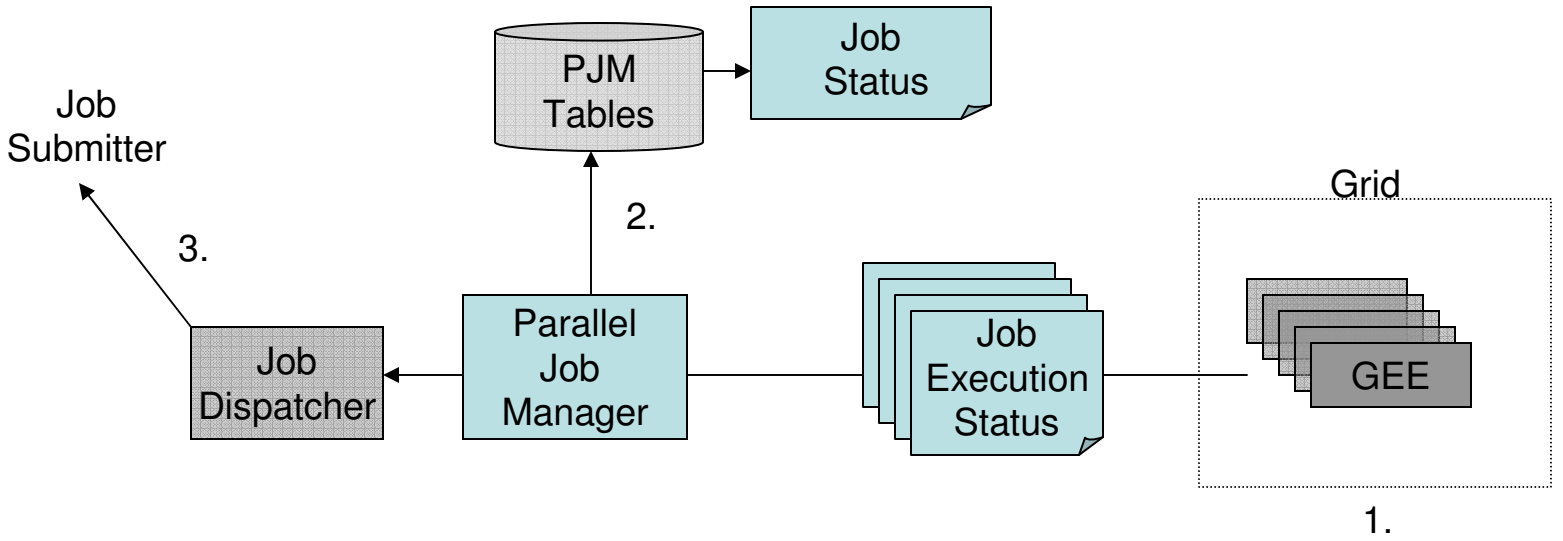
- Administrator only manages the top-level (logical) job; PJM, under the covers, manages the sub-jobs.

Submitting a job to the Parallel Job Manager



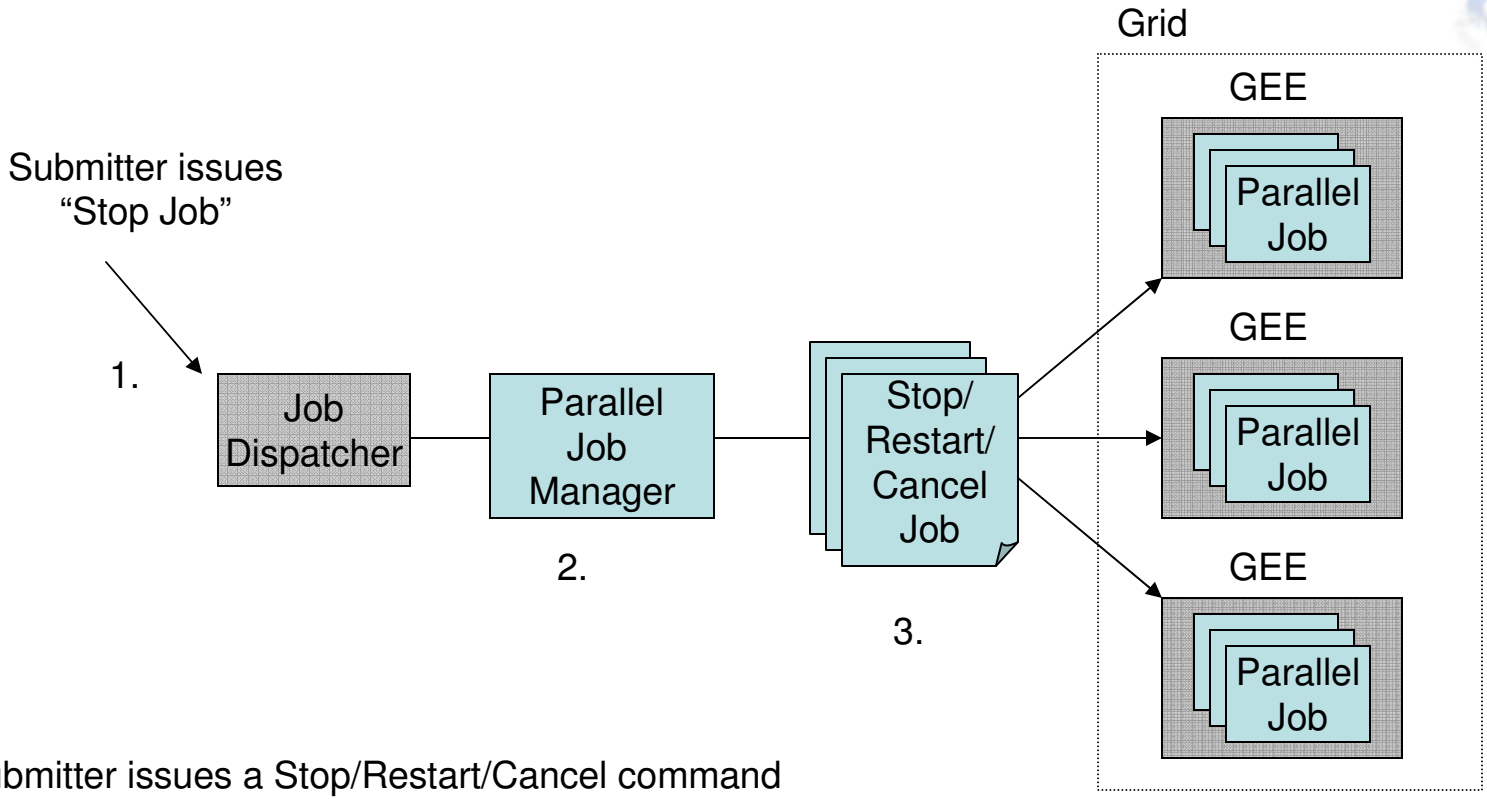
1. Large, single job is submitted to the Job Dispatcher of XD Compute Grid
2. The Parallel Job Manager (PJM), with the option of using job partition templates stored in a repository, breaks the single batch job into many smaller partitions.
3. The PJM dispatches those chunks across the cluster of Grid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

Monitoring a job to the Parallel Job Manager



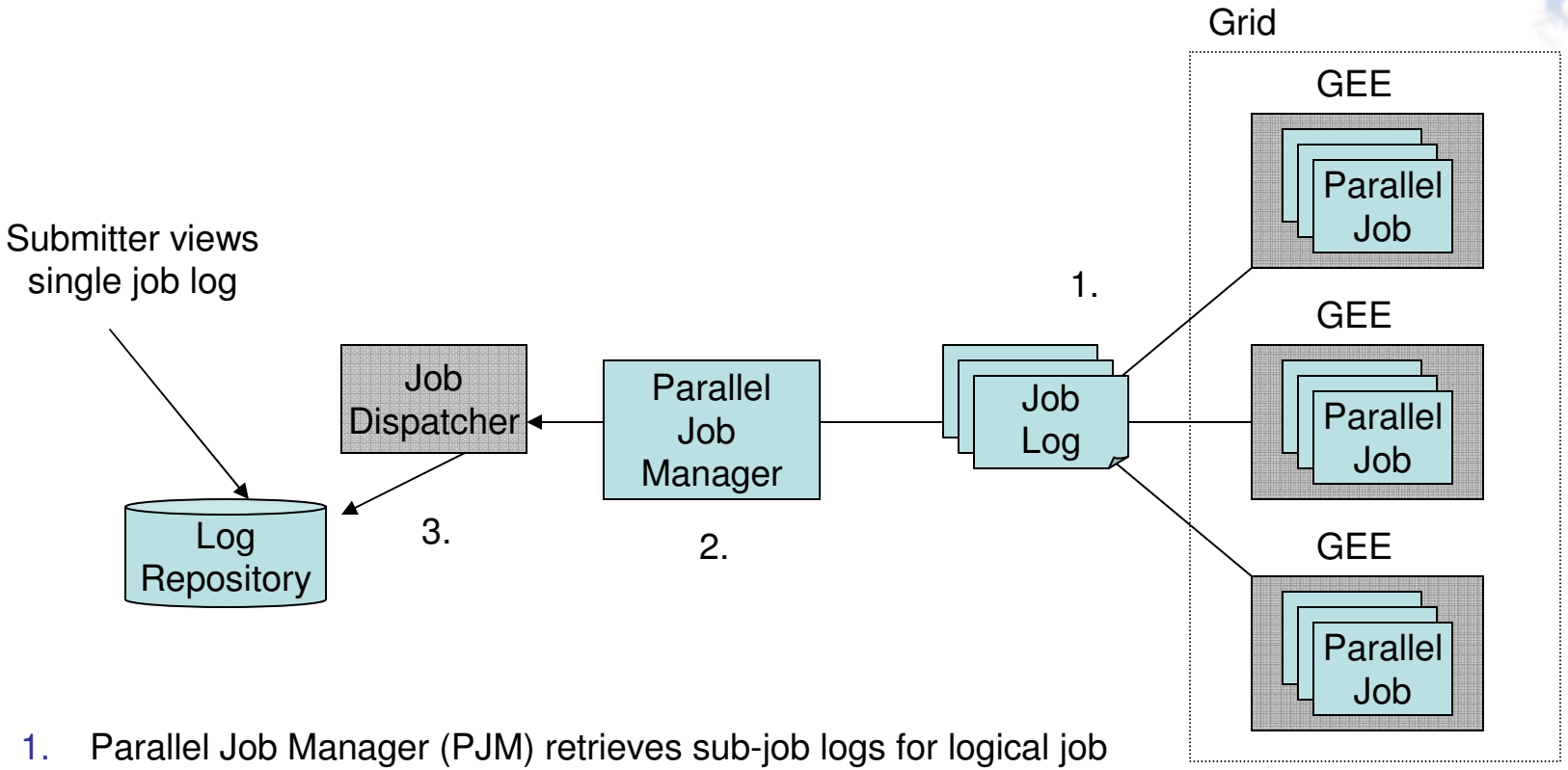
1. Execution status of parallel jobs is reported to the Parallel Job Manager (PJM) from the cluster of GEE's
2. PJM persists the job status in its database tables
3. Job submitter can be notified of status updates and overall progress by the Job Dispatcher.

Managing parallel jobs



1. Submitter issues a Stop/Restart/Cancel command
2. Parallel Job Manager (PJM) determines which jobs must be Stopped/Restarted/Canceled
3. Sub-jobs are issued a Stop/Restart/Cancel command by the Compute Grid infrastructure.

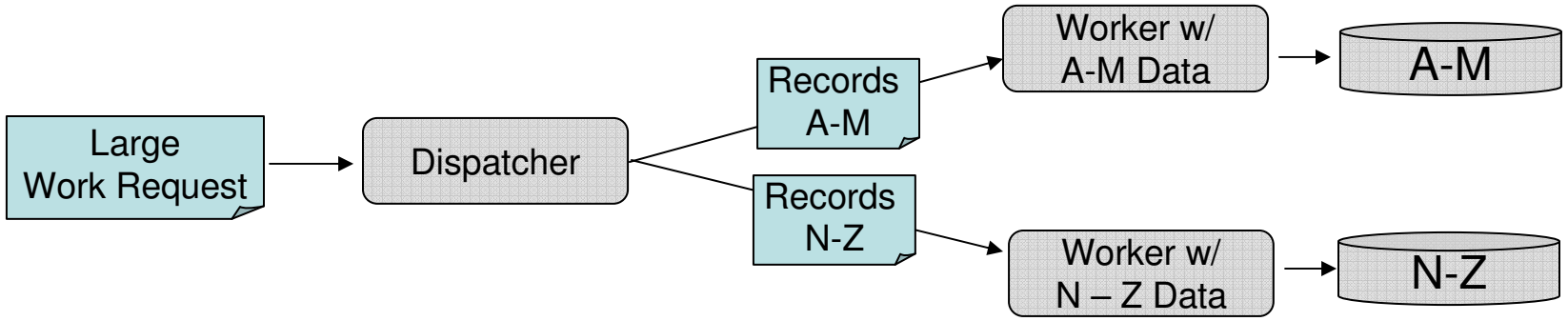
Managing logs for parallel jobs



1. Parallel Job Manager (PJM) retrieves sub-job logs for logical job
2. PJM aggregates the many logs.
3. Long-Running Scheduler stores the job log into the log repository where the submitter can view them.

Compute Grid + XTP = eXtreme Batch

Bringing the data closer to the business logic



-Proximity of the business logic to the data significantly influences performance

-Bring data to the business logic via caching

-Bring business logic to the data via co-location

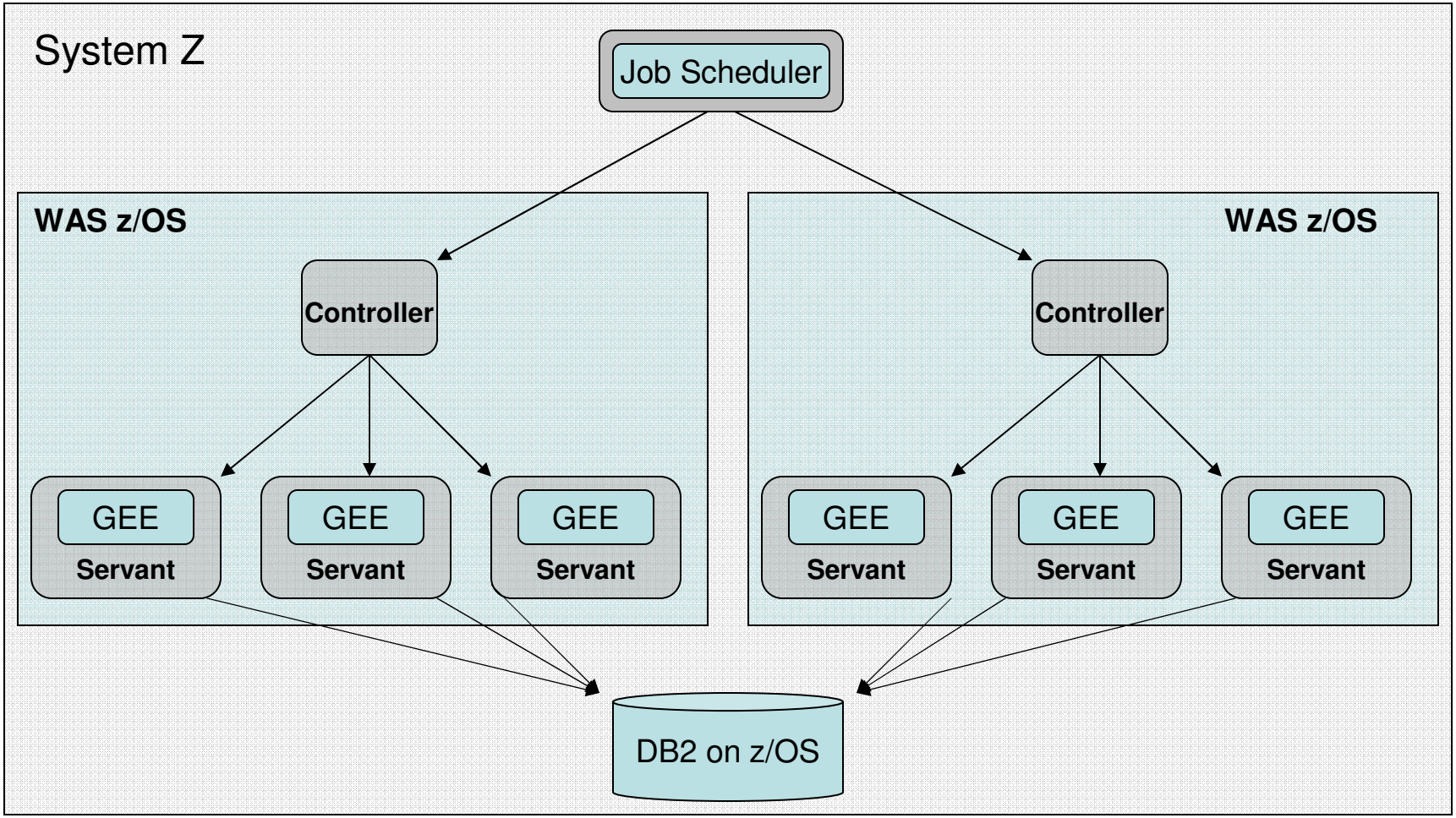
- Increase cache hits and reduce data access through affinity routing

- Data is partitioned across the cluster of workers

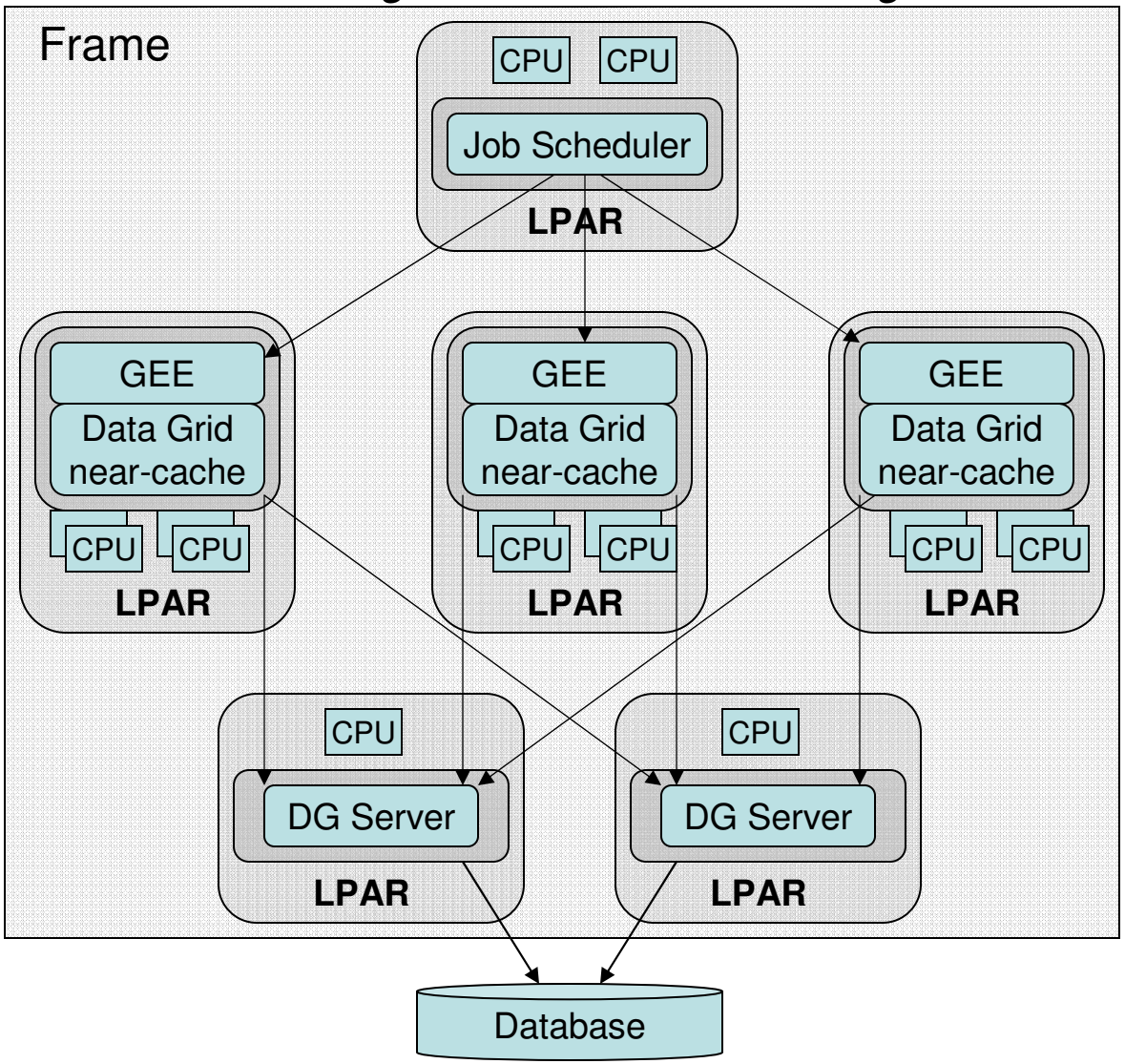
- Work requests are divided into partitions that correspond to the data

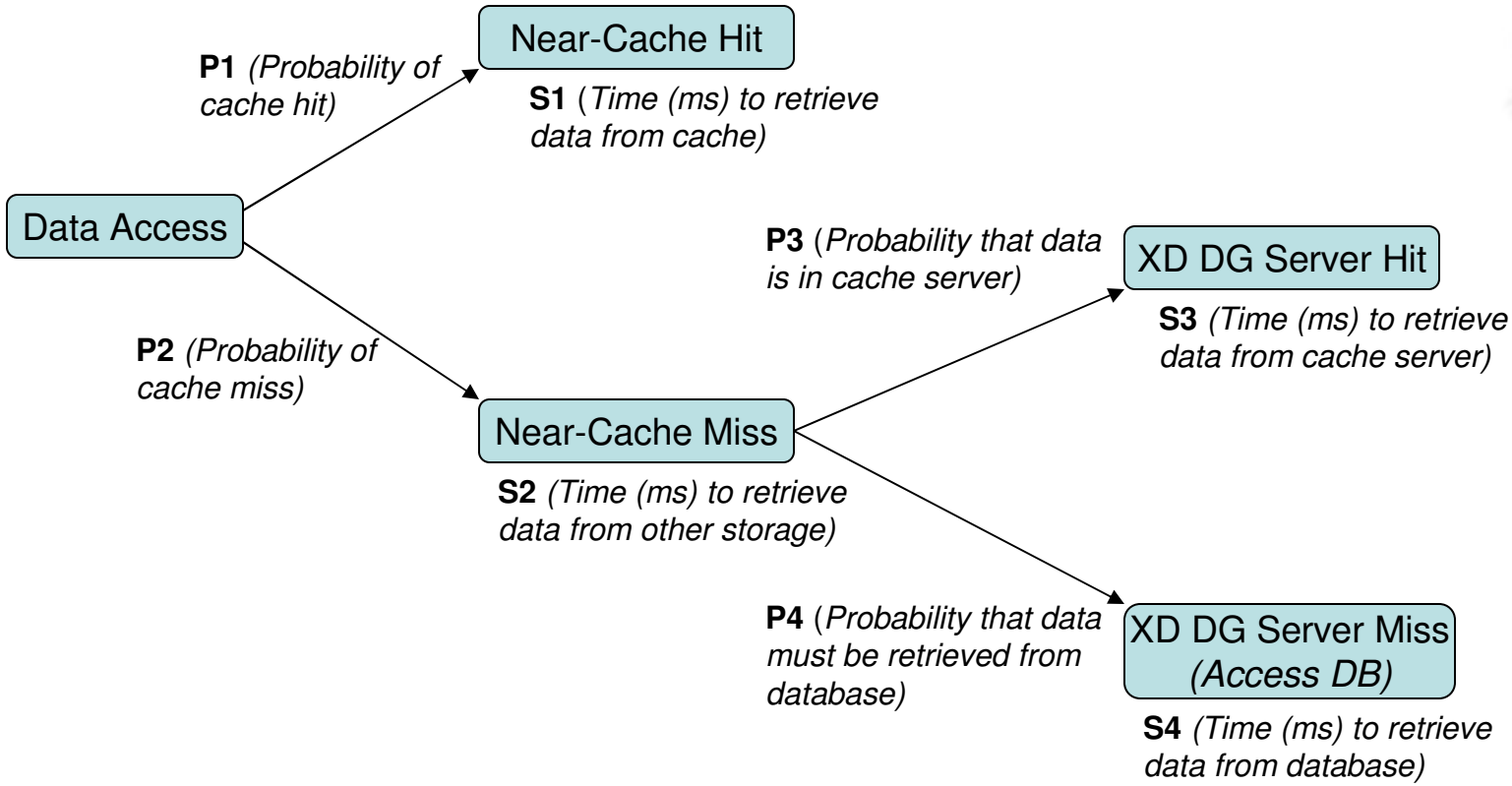
- Work partitions are intelligently routed to the correct work with the data preloaded.

Proximity to the Data- *Co-location of business logic with data*



Proximity to the Data- *Bring data to the business logic with caching*





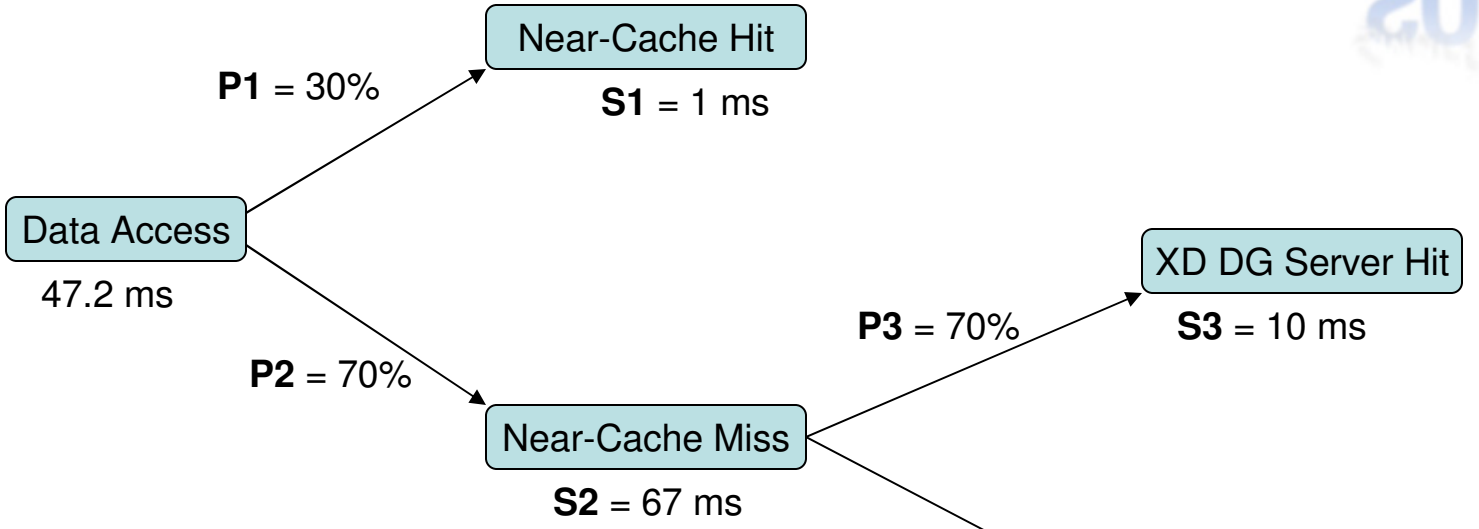
Data Access time (ms) =

(Probability of near- cache hit) * (Time to retrieve data from near-cache) +
(Probability of near-cache miss) * (time to retrieve data from other storage);

Time to retrieve data from other storage (ms) =

(Probability that data is in cache server) * (Time to retrieve data from cache server) +
(Probability that data must be retrieved from database) * (time to retrieve data from database);

Example calculation



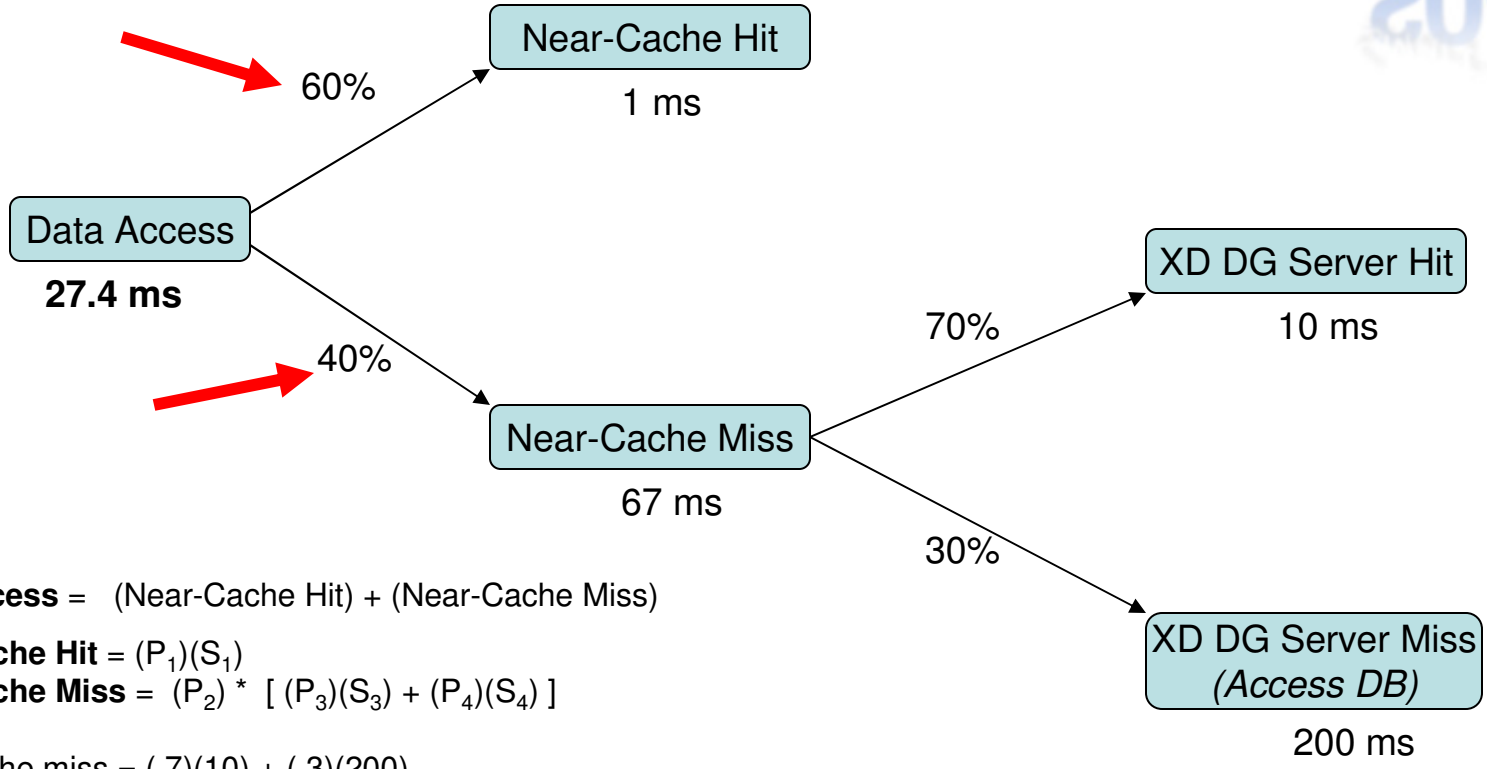
Data Access = (Near-Cache Hit) + (Near-Cache Miss)

Near-Cache Hit = $(P_1)(S_1)$
Near-Cache Miss = (P_2) * [$(P_3)(S_3)$ + $(P_4)(S_4)$]

Near-cache miss = (.7)(10) + (.3)(200)
= 7 + 60 = 67 ms

Data Access = (.3)(1) + (.7)(67)
= .3 + 46.9 = **47.2 ms**

Example calculation- effects of applying affinity routing.



Data Access = (Near-Cache Hit) + (Near-Cache Miss)

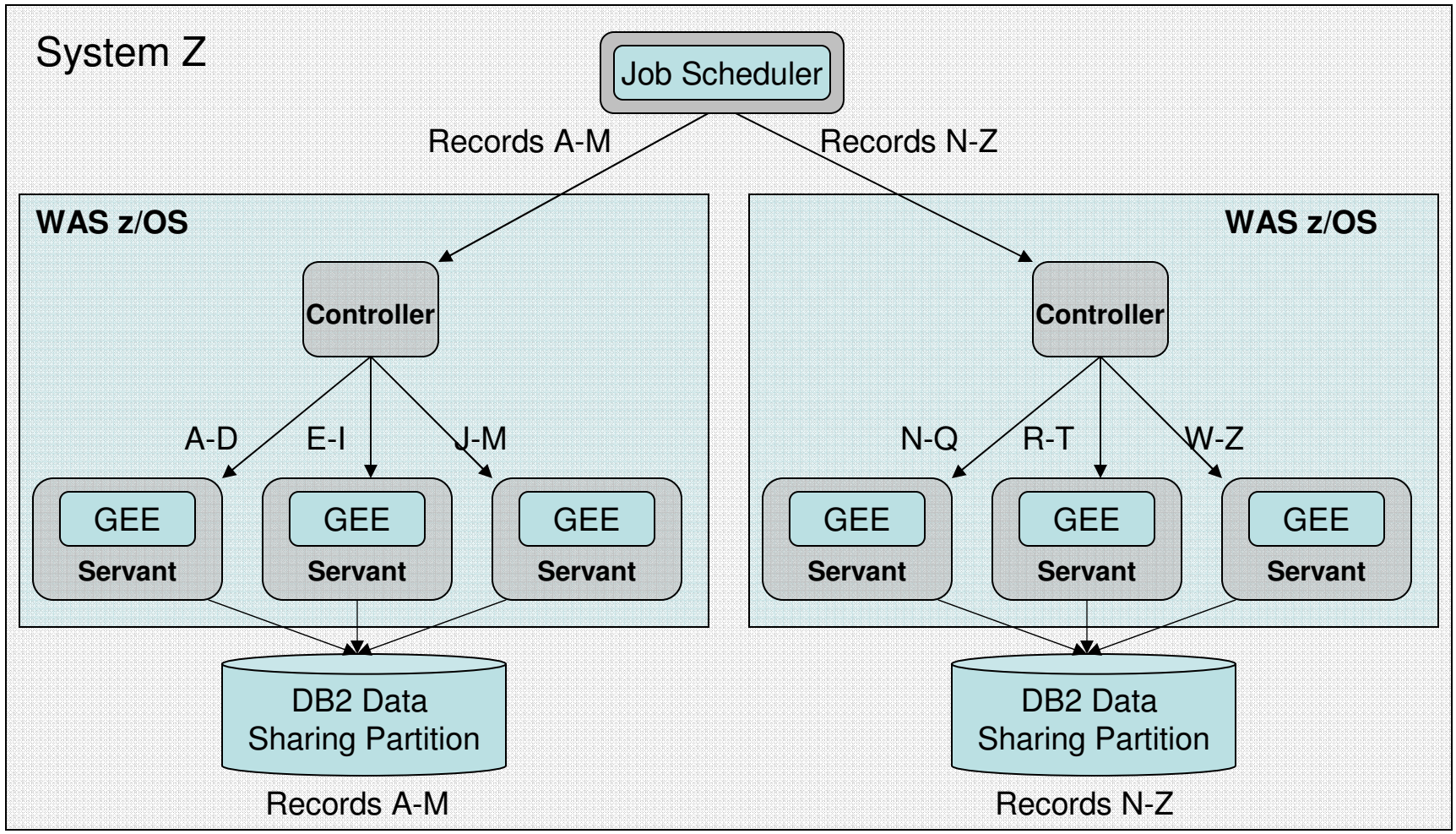
Near-Cache Hit = $(P_1)(S_1)$
Near-Cache Miss = (P_2) * [$(P_3)(S_3)$ + $(P_4)(S_4)$]

Near-cache miss = (.7)(10) + (.3)(200)
= 7 + 60 = 67 ms

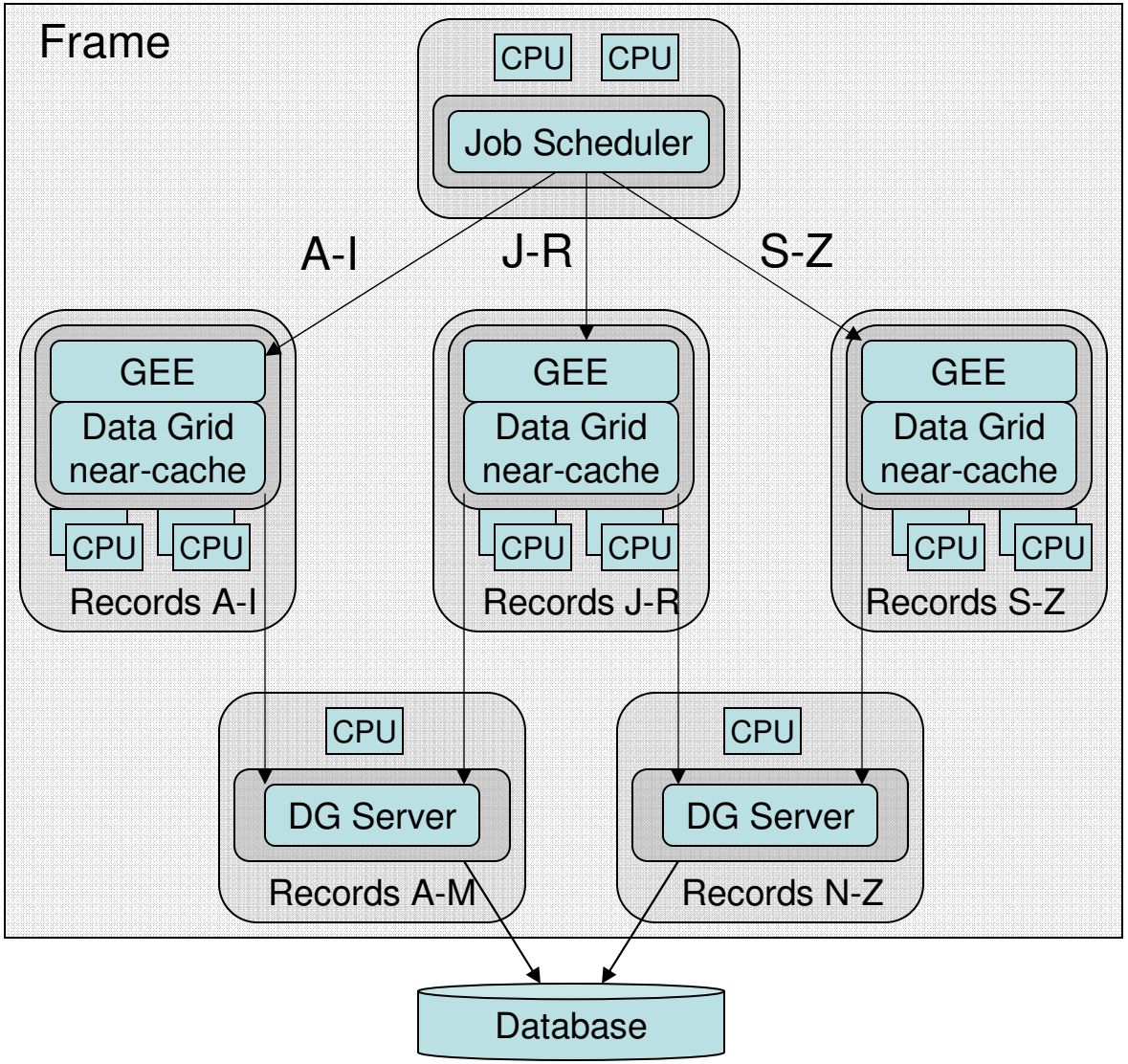
Data Access = (.6)(1) + (.4)(67)
= .6 + 26.8 = **27.4 ms**

(47.2 – 27.4) / 47.2 = **42% improvement in data access time**

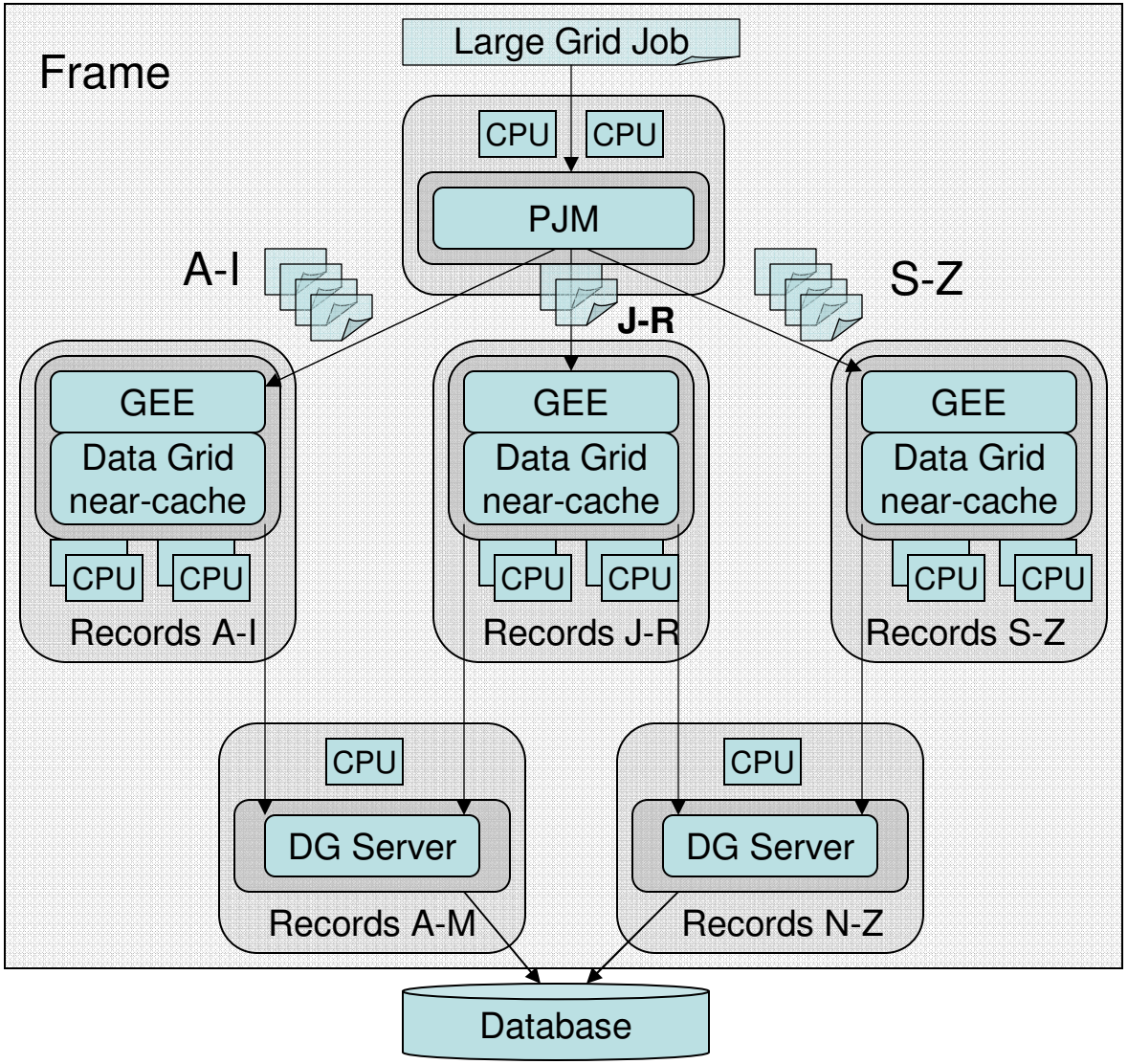
Affinity Routing- *Partitioned data with intelligent routing of work*



Affinity Routing- *Partitioned data with intelligent routing of work*



Divide and Conquer- *Highly Parallel Grid Jobs*



XD Compute Grid and Grid/Utility Computing

-Grid Computing is the coordinated execution of 1000's of jobs across a collection of resources

- Operational Control and Management is essential

- Utility Computing and Goals-Oriented infrastructures provide

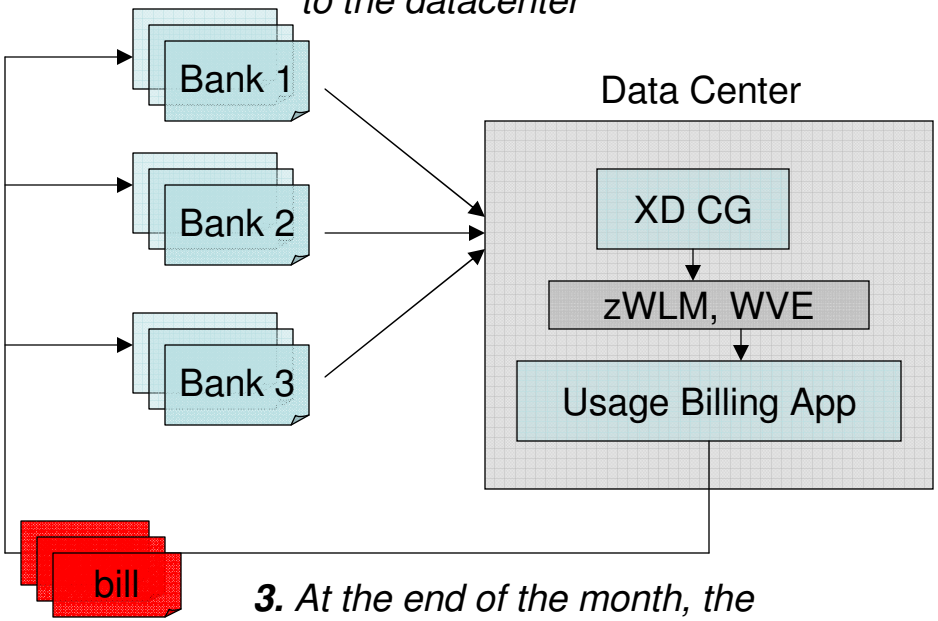
  - Resource utilization metrics for chargeback

  - Virtualization of underlying hardware and software resources

  - Enforcement of service-level agreements within the virtualized infrastructure

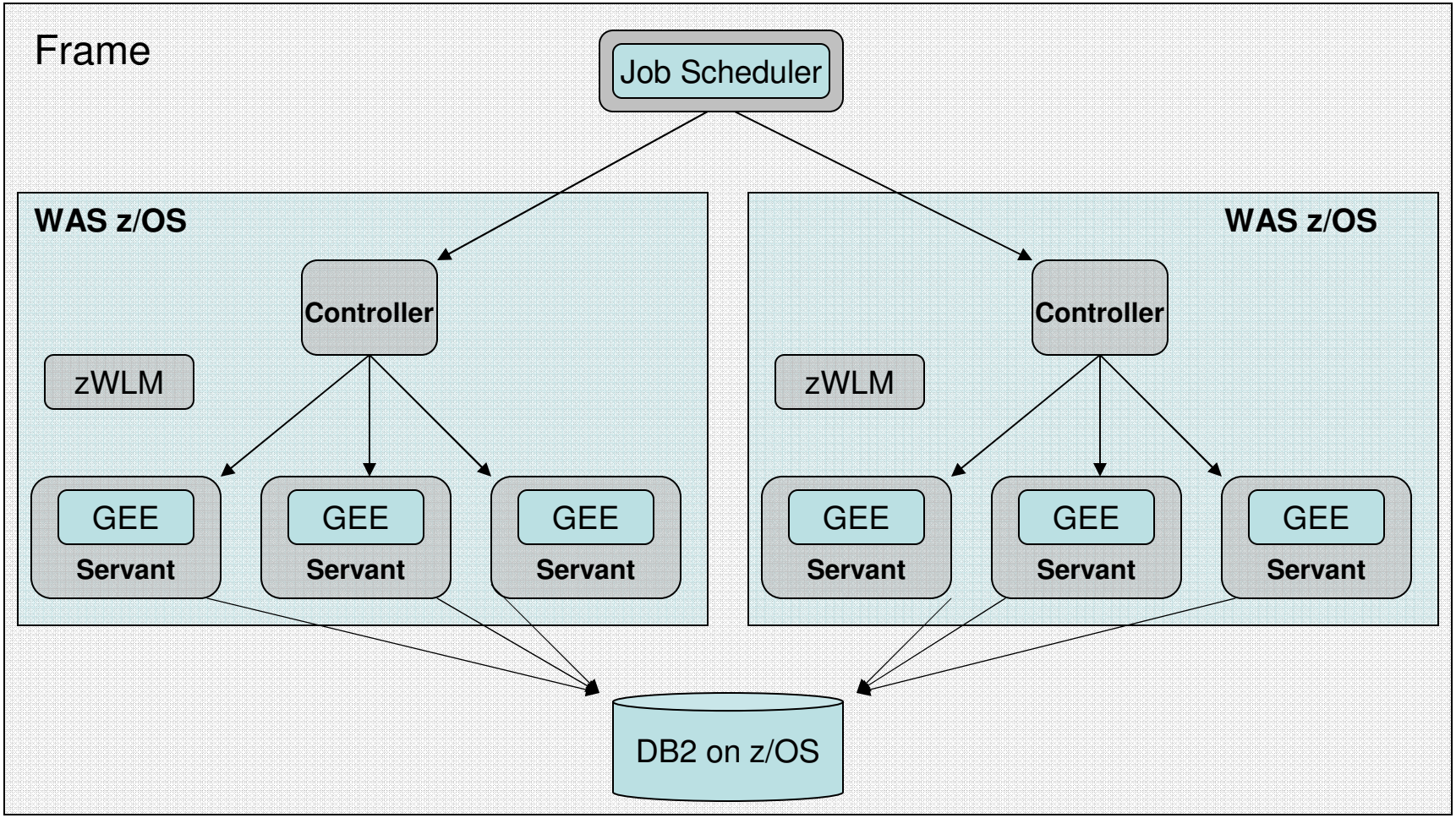
  -Derived from lessons and technologies from the Mainframe

Batch as a service

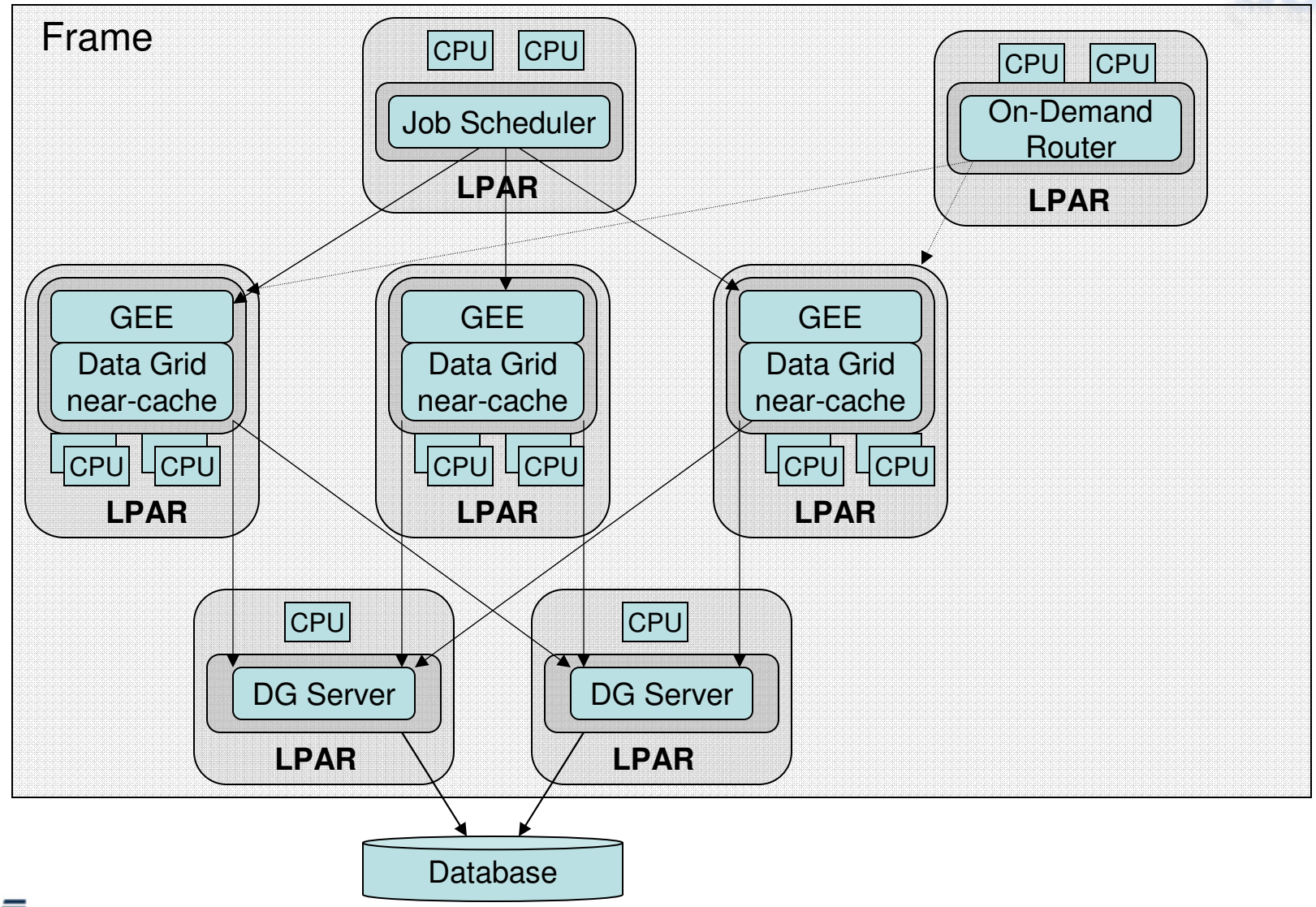


2. *Datacenter executes workloads for each bank, keep tracking of exactly how many resources each bank's jobs used. Achieved on distributed platforms via the On-Demand Router; achieved on z/OS by leveraging the usage accounting facilities of zWLM, RMF, and other system facilities of z/OS*

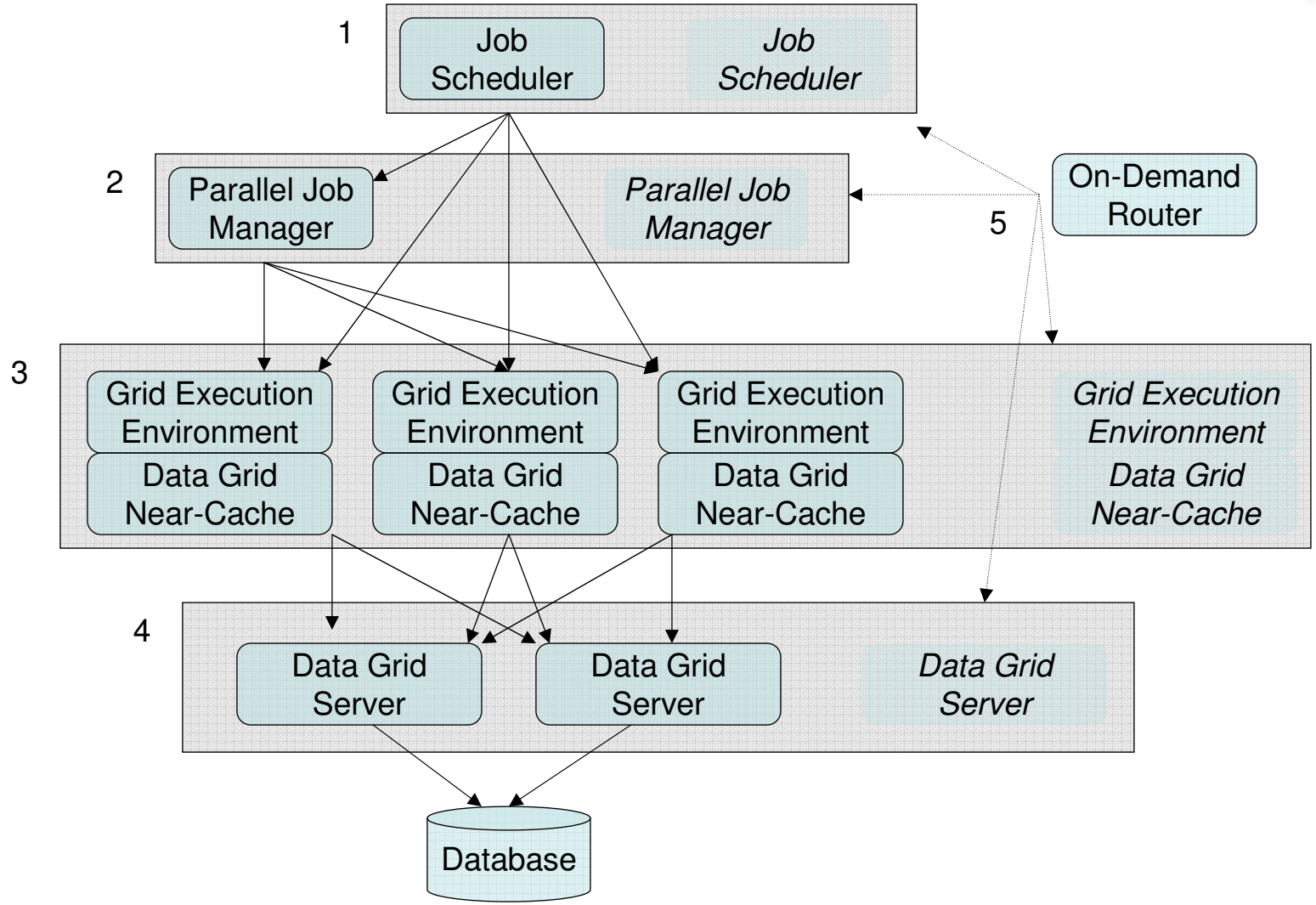
On-Demand Scalability- *With WebSphere z/OS*



On-Demand Scalability- *With WebSphere Virtual Enterprise*



Bringing it all together with “WebSphere XD”



Key Influencers for High Performance Compute Grids

- *Proximity to the Data*
  - Bring the business logic to the data: co-locate on the same platform
  - Bring the data to the business logic: in-memory databases, caching
- *Affinity Routing*
  - Partitioned data with intelligent routing of work
- *Divide and Conquer*
  - Highly parallel execution of workloads across the grid
- *On-Demand Scalability*

Backup

Appendix A- Best Practices

Appendix A – Best Practices

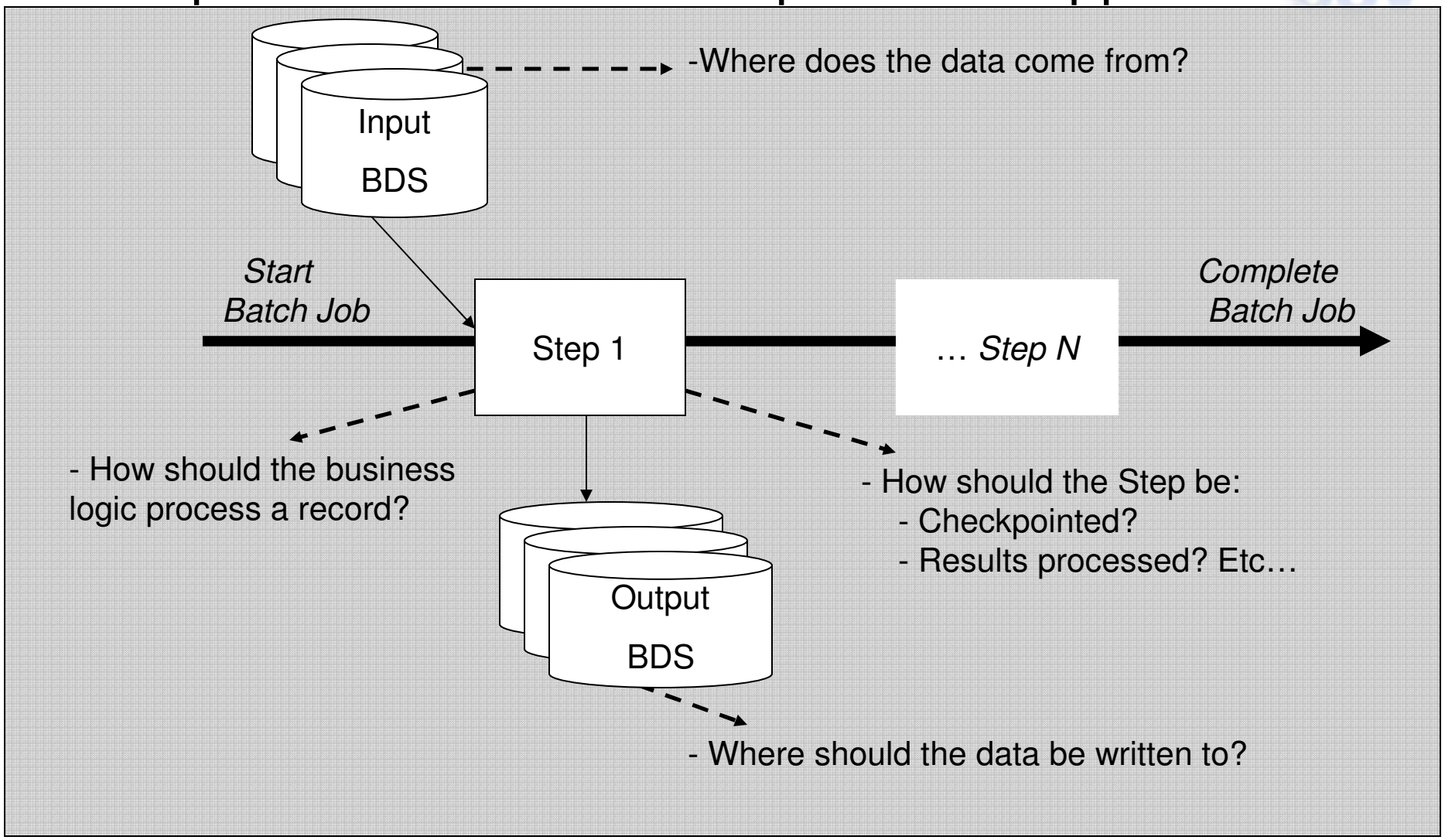
- Application Design
- Parallel Job Manager
- Infrastructure Design
- Misc…

Application Design

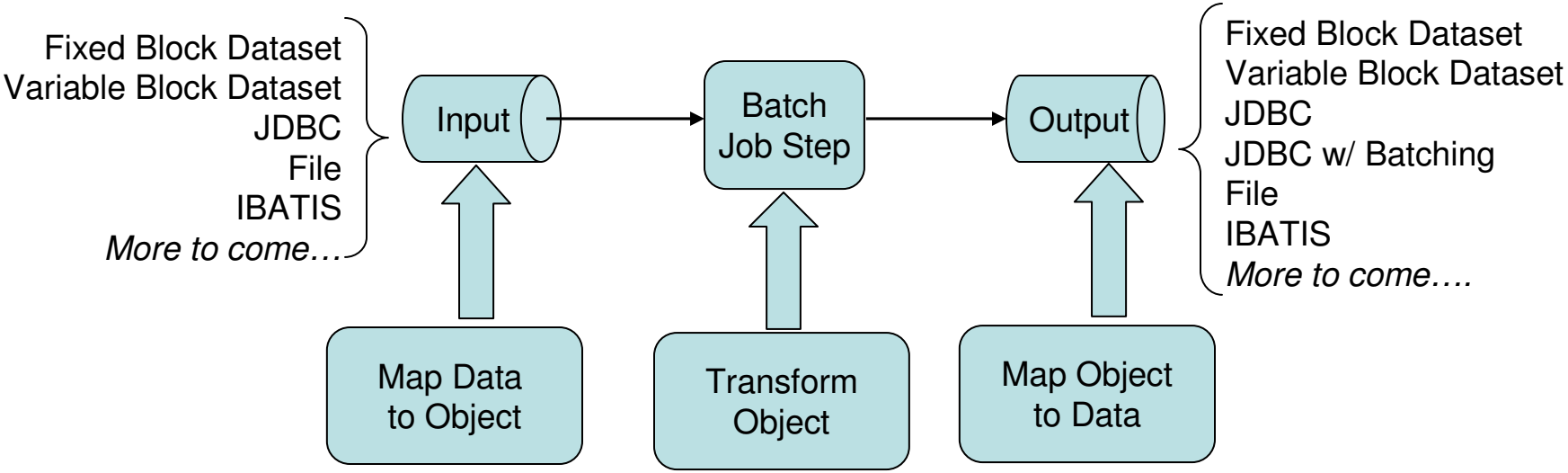
Application Design Considerations

- Strategy Pattern for well structured batch applications
  - Use the BDS Framework!!!
  - Think of batch jobs as a record-oriented Input-Process-Output task
  - Strategy Pattern allows flexible Input, Process, and Output objects *(**think "toolbox"** of input BDS, process steps, and output BDS)*
- Designing "services" shared across OLTP and Batch
  - Cross-cutting Functions (Logging, Auditing, Authorization, etc)
  - Record-oriented services logic
    - Service doesn't care where the input record came from (OTLP or Batch)
  - POJO-based "services", not heavy-weight services
  - Be aware of transaction scope for OLTP and Batch. *TxRequiresNew in OLTP + TXRequires in Batch => Deadlock Possible*
- Designing the Data Access Layer (DAL)
  - DAO Factory pattern to ensure options down the road
  - Context-based DAL for OLTP & Batch in same JVM
  - Configuration-based DAL for OLTP & Batch in different JVM's

Components of an XD Compute Grid Application



How to think about batch jobs



-Customer implements pattern interfaces for input/output/step

-Pattern interfaces are *very* lightweight.

-They follow ***typical*** lifecycle activities:

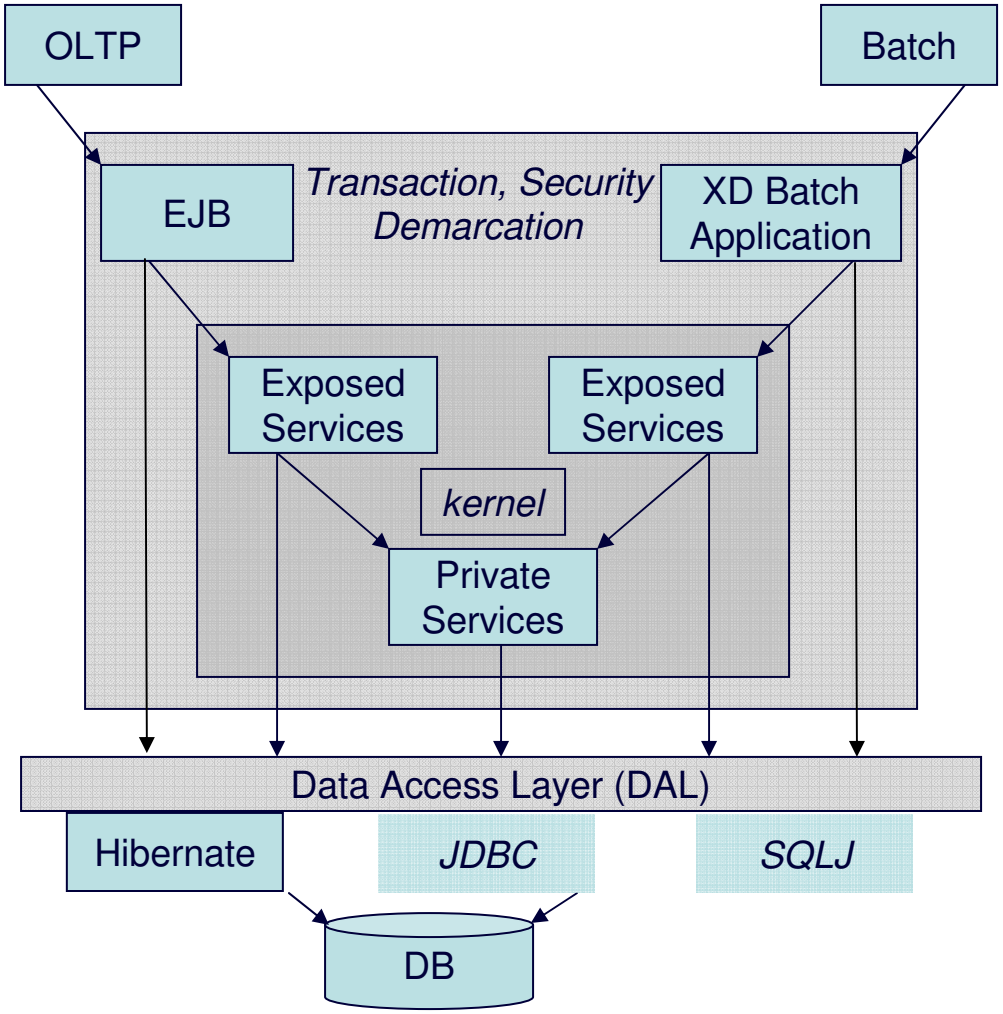
-I/O patterns: initialize, map raw data to single record, map single record to raw data, close

-Step pattern: Initialize, process a single record, destroy.

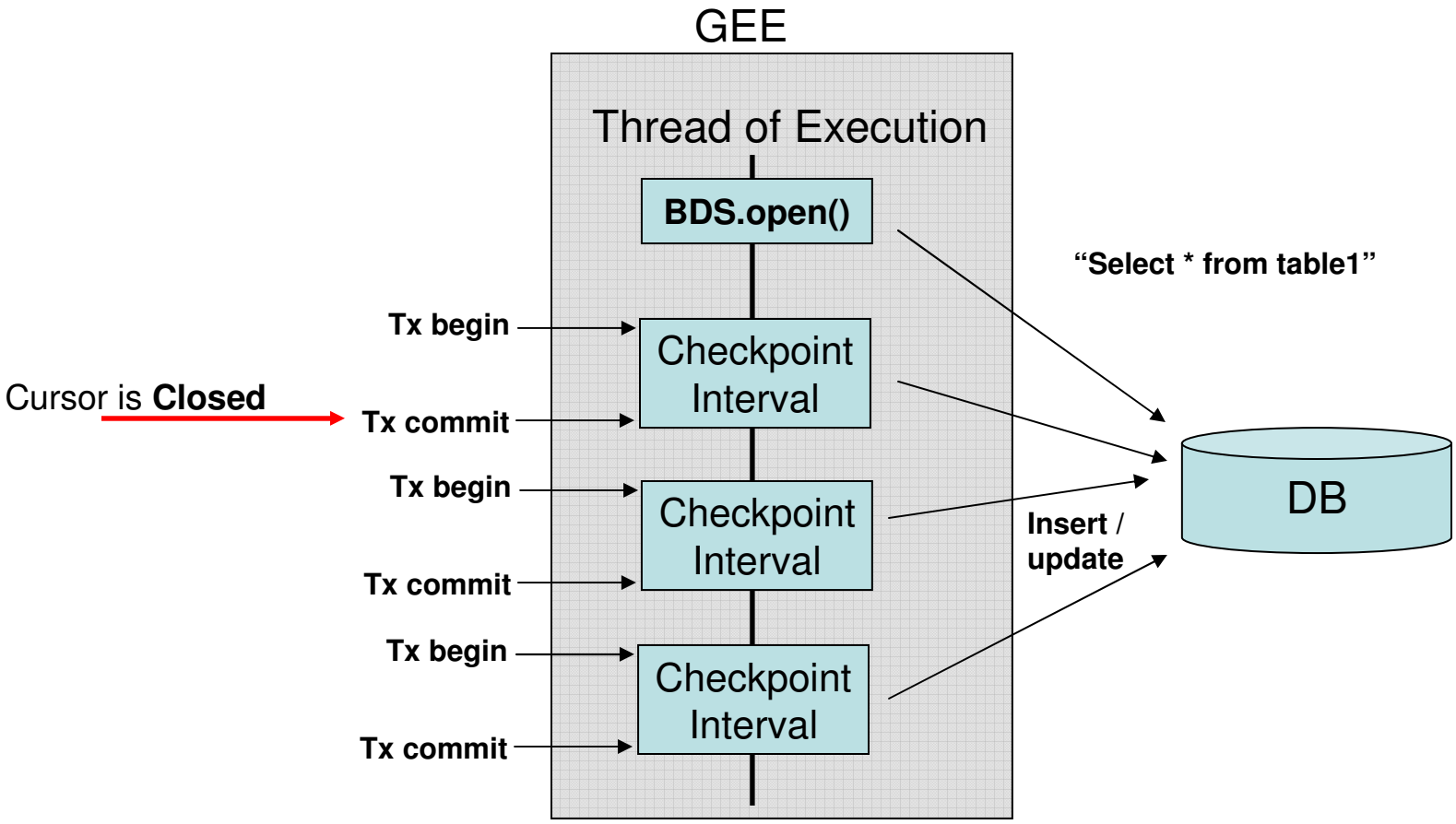
Example Application Architecture for Shared OLTP and Batch Services

-J2EE and XD manage Security, transactions

- *Spring-based application Configuration*
- *Custom authorization service within kernel for business-level rules*
- *Initial data access using Hibernate. Investigating JDBC, SQLJ, etc*



Key “GOTCHA”… Curser Holdability



Cursor Holdability Options

- If XA Datasource
  - Configure Last Participant Support
  - Stateful Session Bean Façade Pattern (via BDS Framework)
  - Non-transactional Data Source (via EJB 3 Feature Pack)
- If Non-XA
  - Configure Cursor Holdability on the Datasource

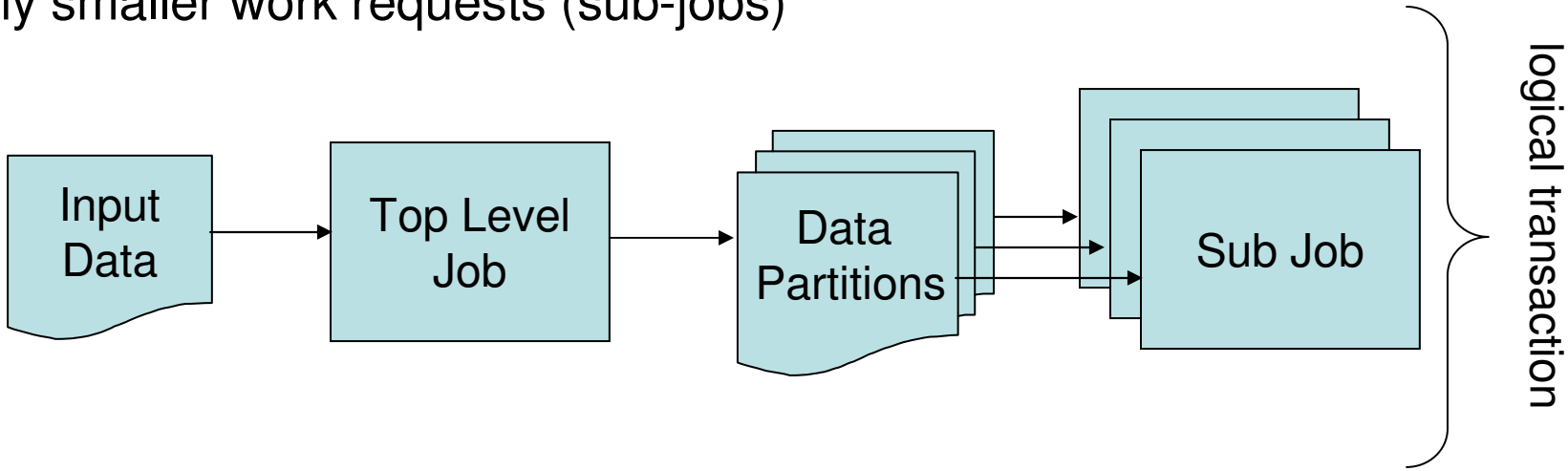
Parallel Job Manager

Parallel Job Manager Best Practices (circa v6.1.0.1)

- Understanding the model
- Regarding data partitions
- SPI techniques
- About logical transactions
- Persisting collector/analyzer state data

Understanding the Model

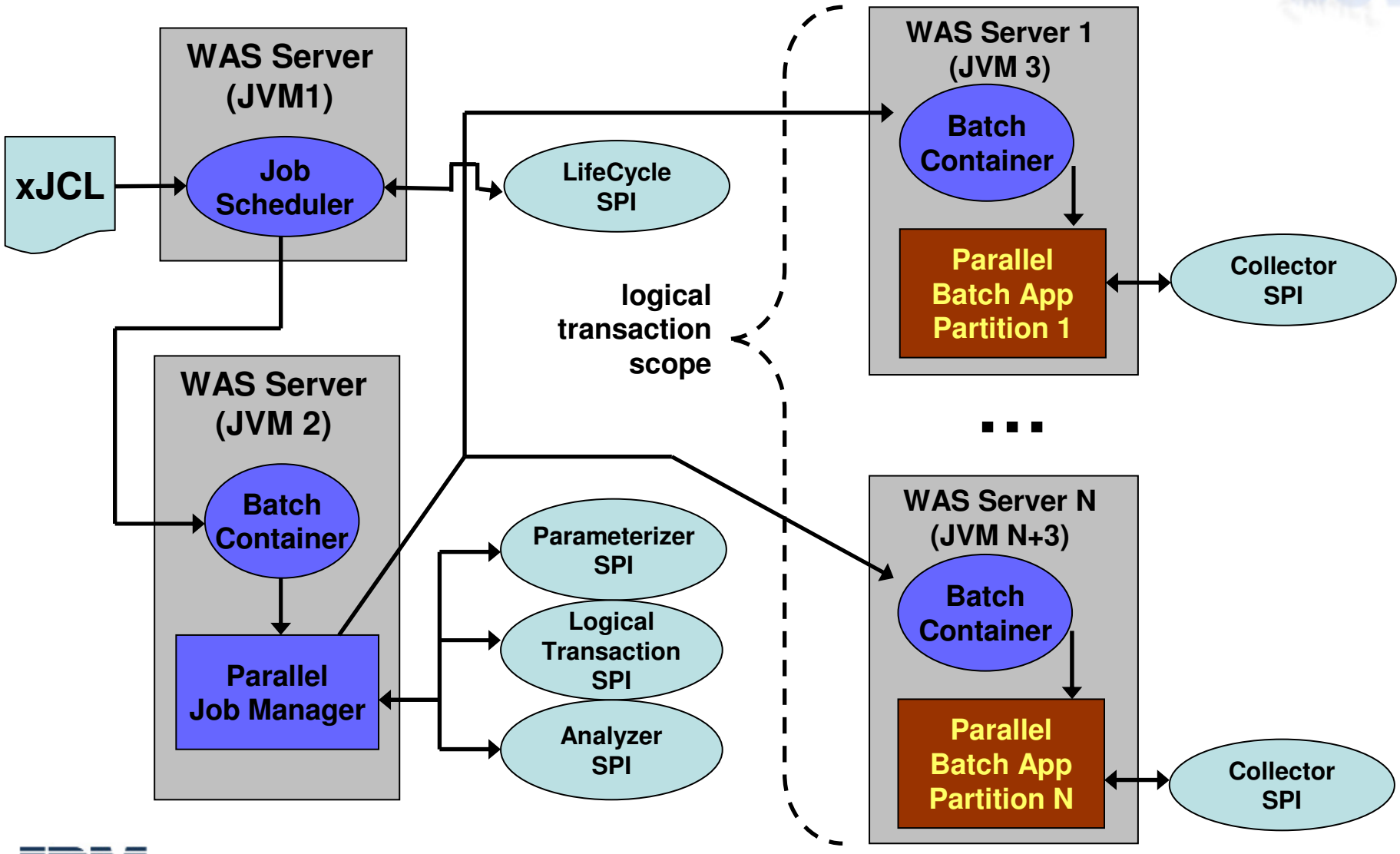
- Parallel Job Manager (PJM) decomposes a large work request into many smaller work requests (sub-jobs)



- PJM then provides operational control over the sub-jobs executing across the job endpoints – note sub-jobs are clones

- Administrator only manages the top-level (logical) job; PJM, under the covers, manages the sub-jobs.

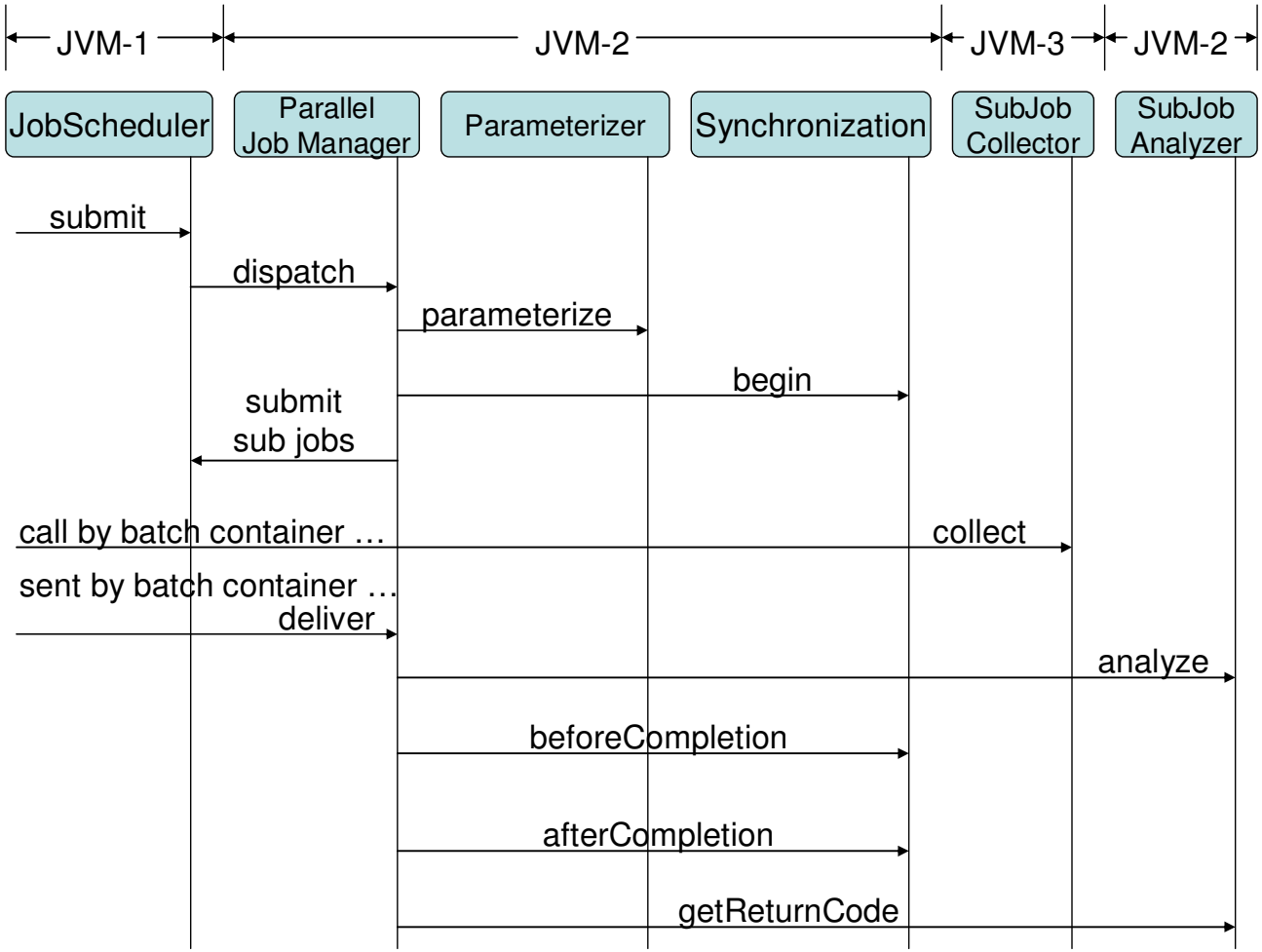
Understanding the Model (the SPIs)



Understanding the model (the SPIs…)

- Parameterizer
  - Called by PJM
  - Specifies partitioning scheme (number of subjobs, unique parameters for each subjob instance)
- LogicalTransaction
  - Called by PJM
  - Demarcates logical transaction (begin, commit, etc)
  - Logical in nature – no actual resource registration – this is not JTA
- SubJobCollector
  - Called by Batch Container in app server where a subjob executes
  - Called before each checkpoint
  - Allows an Externalizable to be sent to the SubJobAnalyzer
- SubJobAnalyzer
  - Called by PJM
  - Called each time a SubJobCollector externalizable arrives
  - Called each time a sub-job ends (receives sub-job return code)
- LifeCycle
  - Called by Job Scheduler
  - Called each time a job (any job) changes state (i.e. submitted, executing, ended)

Understanding the model - OID

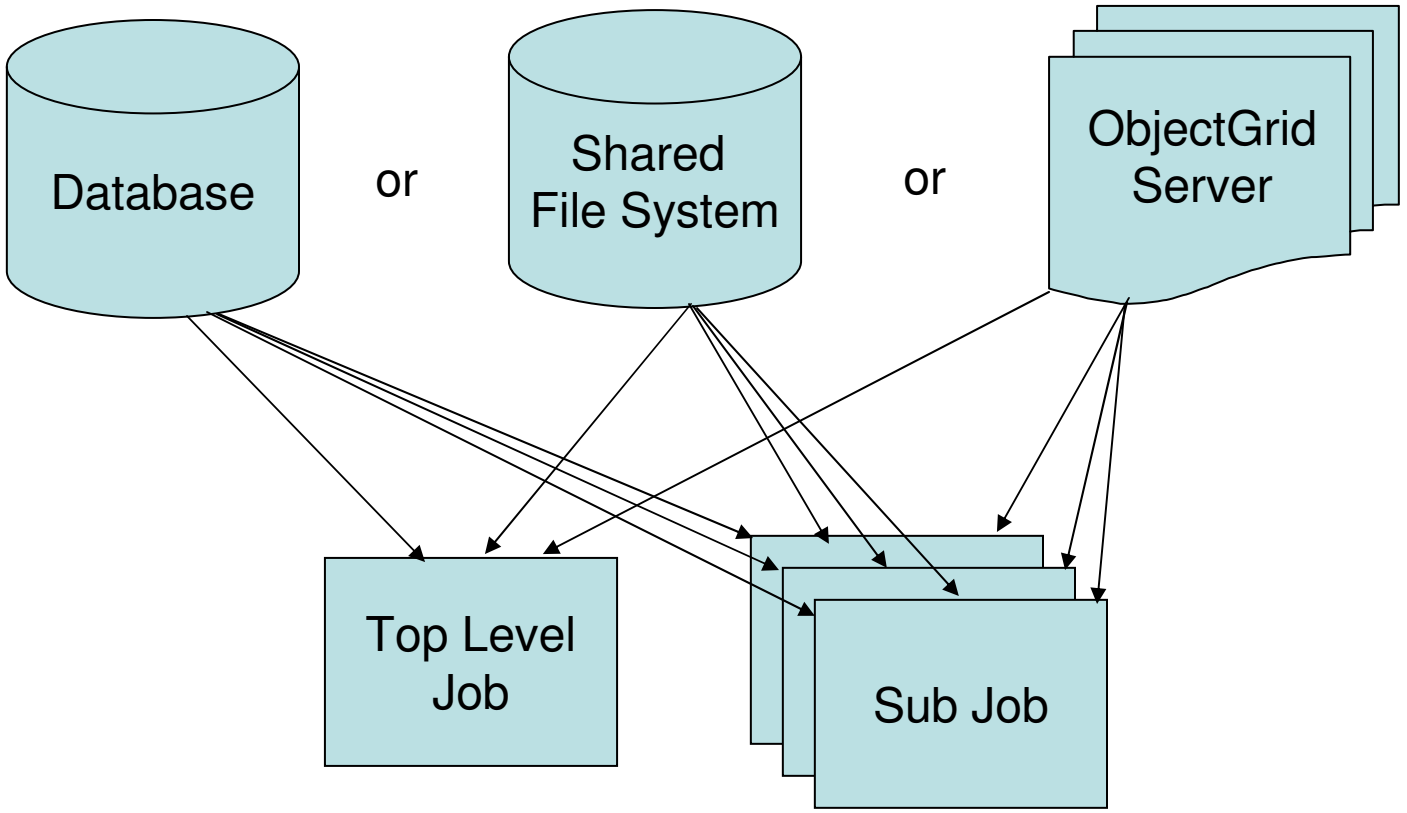


Regarding Data Partitions

- Input records must be partitionable on some natural key
  - i.e. record number, customer number, ObjectGrid partition id, etc
- It is the job of the Parameterizer to implement partition scheme – Compute Grid does not know customer's data!
- Input records (or at least each partition) must be processable independent of other records (or partitions)
- Compute Grid does not provide data transport! Data must be shareable among app servers executing the subjobs – i.e. shared database, shared file system, ObjectGrid server, etc
- It would be possible to implement a top-level job where first job step uses file transfer APIs to stage data partitions on endpoints

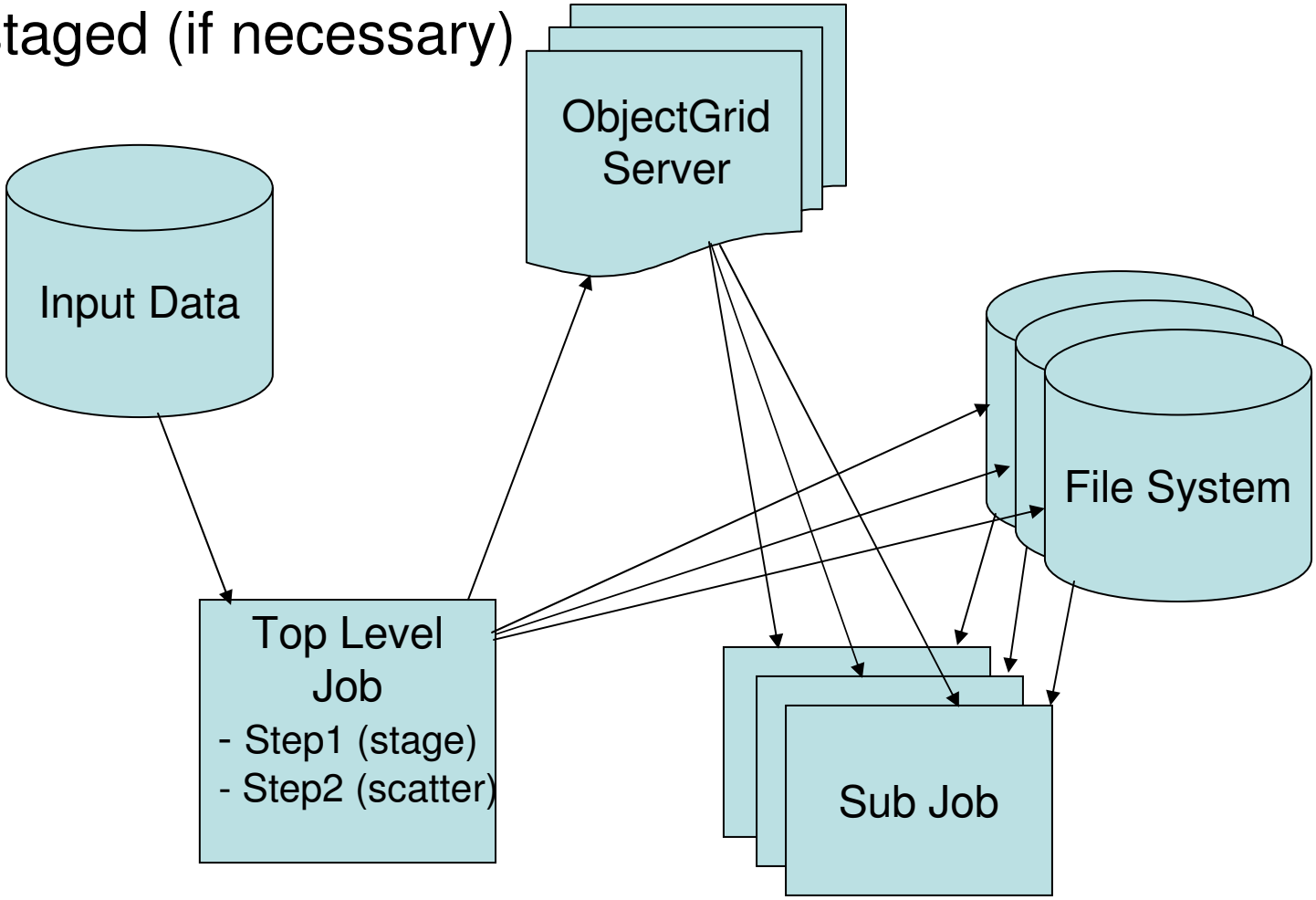
Regarding Data Partitions …

- Data is shared (usually best)



Regarding Data Partitions …

- Data is staged (if necessary)



SPI Techniques

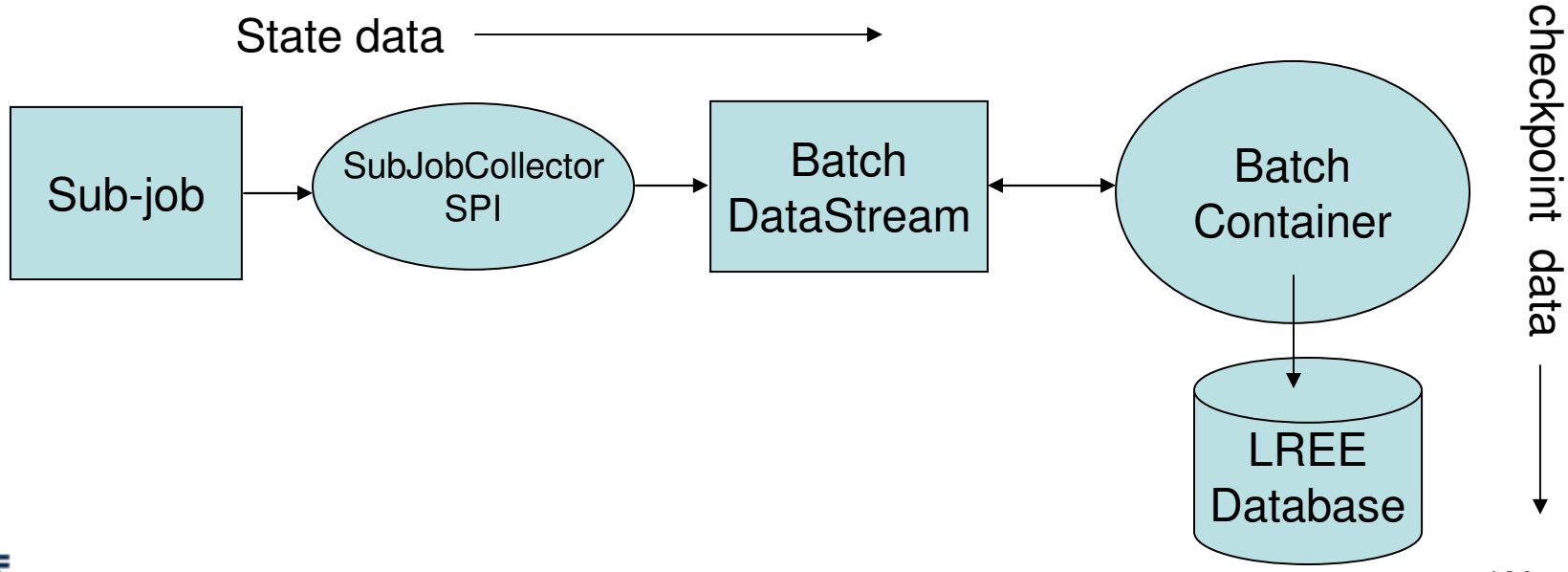
- Remember there is a single Parameterizer SPI for all profiles tied to a WAS instance (sorry ☹)
  - Best practice – use ISSW “SPI Router” from CitiDirect work (contact Patrick Nogay for now)
  - This allows each parallel job to specify its own SPI classes (nice!)
- Parameterizer
  - Using “SPI Router” pattern (granularity can then be per job)
  - xJCL properties specified in top level job are passed to parameterizer (so is job name)
    - This info can be used to further influence Parameterizer’s decision
- Collector/Analyzer
  - Use to communicate sub-job state to top level job – e.g. error counts, etc

About logical transactions

- These are not JTA transactions ! (so application code must handle rollback – compensation model)
- Begin invoked before sub-jobs submitted
- Commit issued after all sub-jobs complete
- Rollback occurs if
  - any sub-job fails
  - any sub-ojb enters restartable state
  - SubJobAnalyzer throws Rollback exception
- Rollback strategies (since transaction is only logical)
  - Use ‘visible” flag in record – mark visible=true upon commit
  - Use staging area – move data from staging to final destination upon commit

Persisting collector state data

- To make collector data part of checkpoint
  - Use “dummy” batch data stream
  - Can be done on either sub-job nodes, top-level job node or both
  - Remember that checkpoint token is 2970 bytes max !



Infrastructure Design

Infrastructure Design Considerations

- High Availability practices
  - Job Scheduler can be made highly available (as of 6.1)
  - Cluster GEE's
- Disaster Recovery practices
  - Today, Active/Inactive approach
  - Tomorrow, Active/Active approach
- Security
  - Job Submitter and Compute Grid Admin roles
  - Options for using Job Submitter identity or Server's identity ***(Performance degradation today!)***
- Connecting Compute Grid to the Enterprise Scheduler
  - JMS Client connector bridges enterprise scheduler to Job Scheduler
  - JMS best practices for securing, tuning, etc apply

High Availability

Topology Questions…

- First, is the Parallel Job Manager (PJM) needed, will you run highly-parallel jobs?
- What are the *high availability* requirements for the JS, PJM, and GEE?
  - Five 9's? Continuous?
- What are the s*calability* requirements for the JS, PJM, GEE?
  - Workloads are predictable and system resources are static?
  - Workloads can fluctuate and system resources are needed on-demand?
- What are the performance requirements for the batch jobs themselves?
  - They must complete within some constrained time window?
- What will the workload be on the system?
  - How many concurrent jobs? How many highly-parallel jobs? Submission rate of jobs?

Topology Considerations…

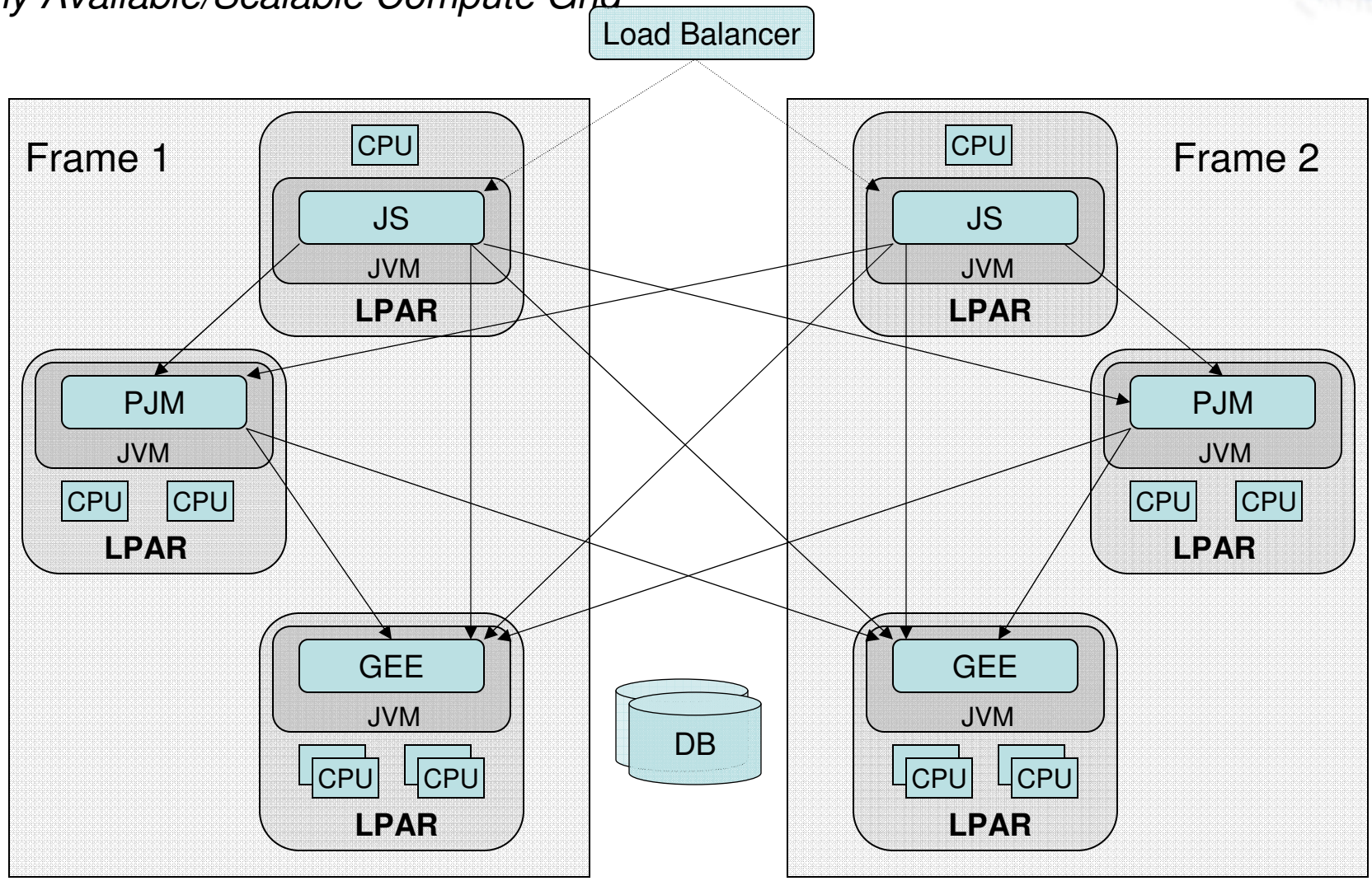
- If the Job Scheduler (JS) does not have system resources available when under load, managing jobs, monitoring jobs, and using the JMC will be impacted.
- If the PJM does not have system resources available when under load, managing highly parallel jobs and monitoring the job partitions will be impacted.
- If the GEE does not have system resources available when under load, the execution time of the business logic will be impacted.
- The most available and scalable production environment will have:
  - Redundant JS. JS clustered across two datacenters.
  - Redundant PJM. PJM clustered across two datacenters.
  - n GEE's, where n is f(workload goals). Clustered across two datacenters

Cost Considerations…

- GEE will most likely require the most CPU resources. The total number of CPU's needed is dependent on:
  - the workload goals
  - max number of concurrent jobs in the system.
- PJM will require fewer CPU's than the GEE. The total number of CPU's needed is dependent on:
  - Rate at which highly-parallel jobs are submitted
  - Max number of concurrent parallel partitions running in the system.
- Job Scheduler will require fewer CPU resources than the GEE, and perhaps the PJM too. The total number of CPU's needed is dependent on:
  - Rate at which jobs will be submitted
  - Max number of concurrent jobs in the system

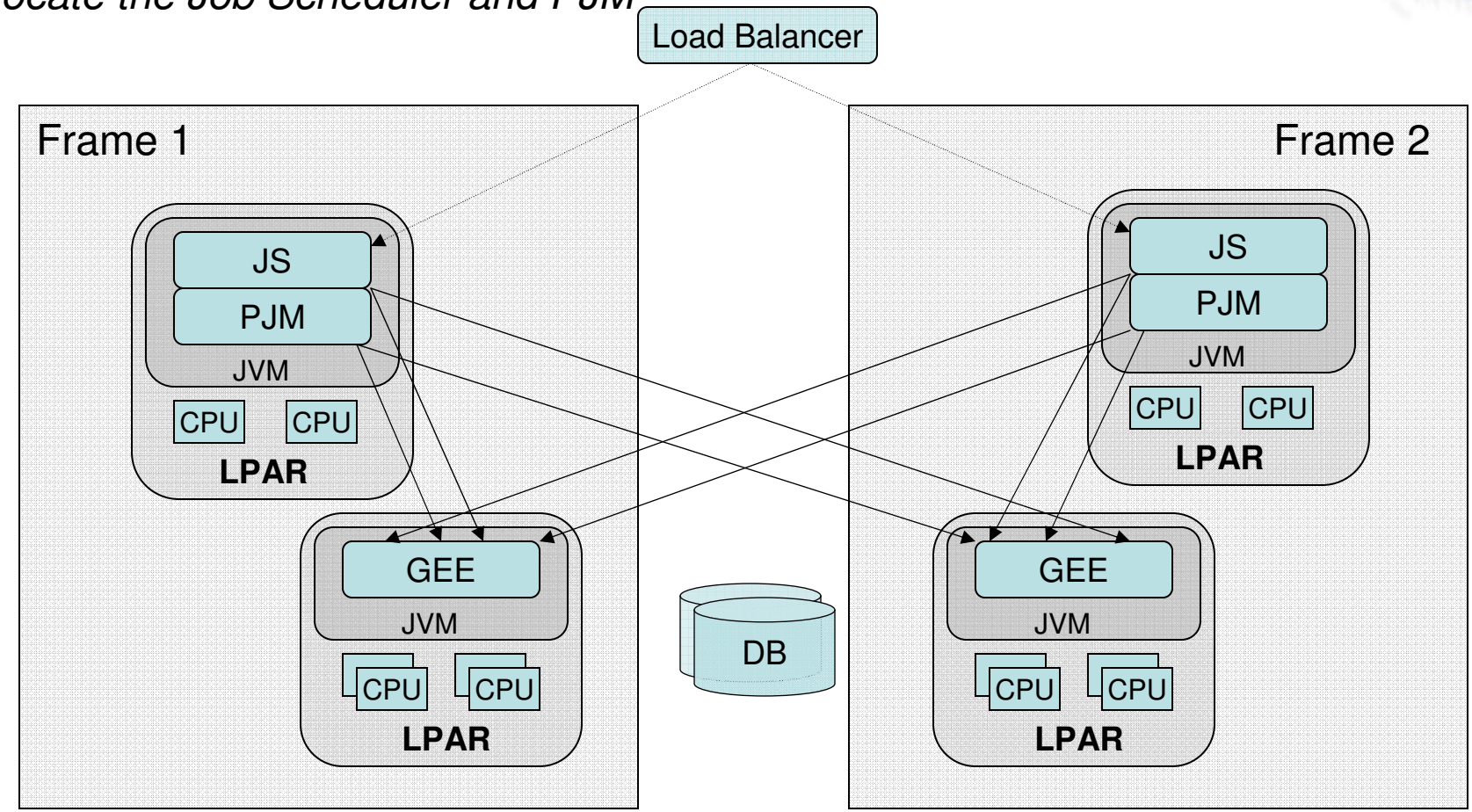
Example Production Topology-

Highly Available/Scalable Compute Grid



Example Production Topology-

Co-locate the Job Scheduler and PJM

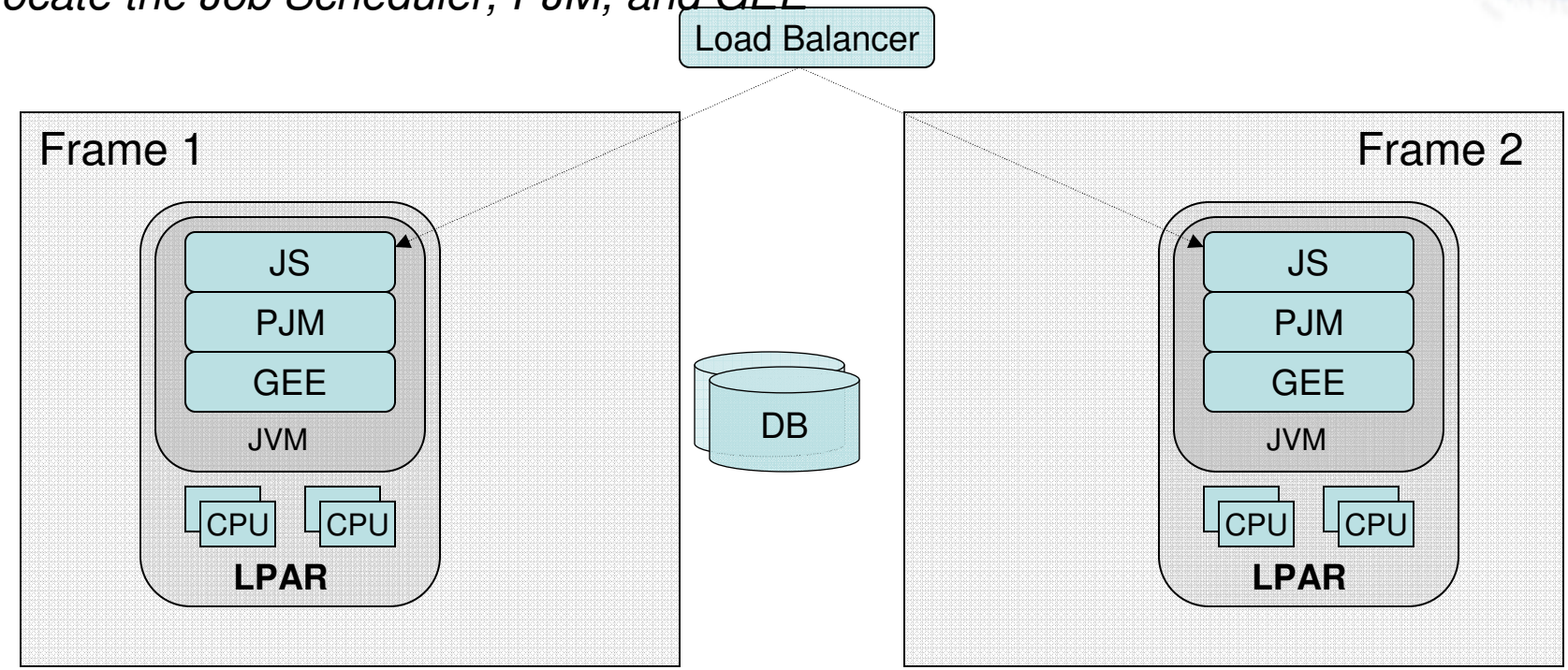


Pro: Faster interaction between JS and PJM due to co-location and ejb-local-home optimizations

Con: Possibility of starving JS or PJM due to workload fluctuations

Example Production Topology-

Co-locate the Job Scheduler, PJM, and GEE



Con: Possibility of starving JS, PJM, and GEE due to workload fluctuations

Con: Not scalable

High Availability – Summary & Key Considerations

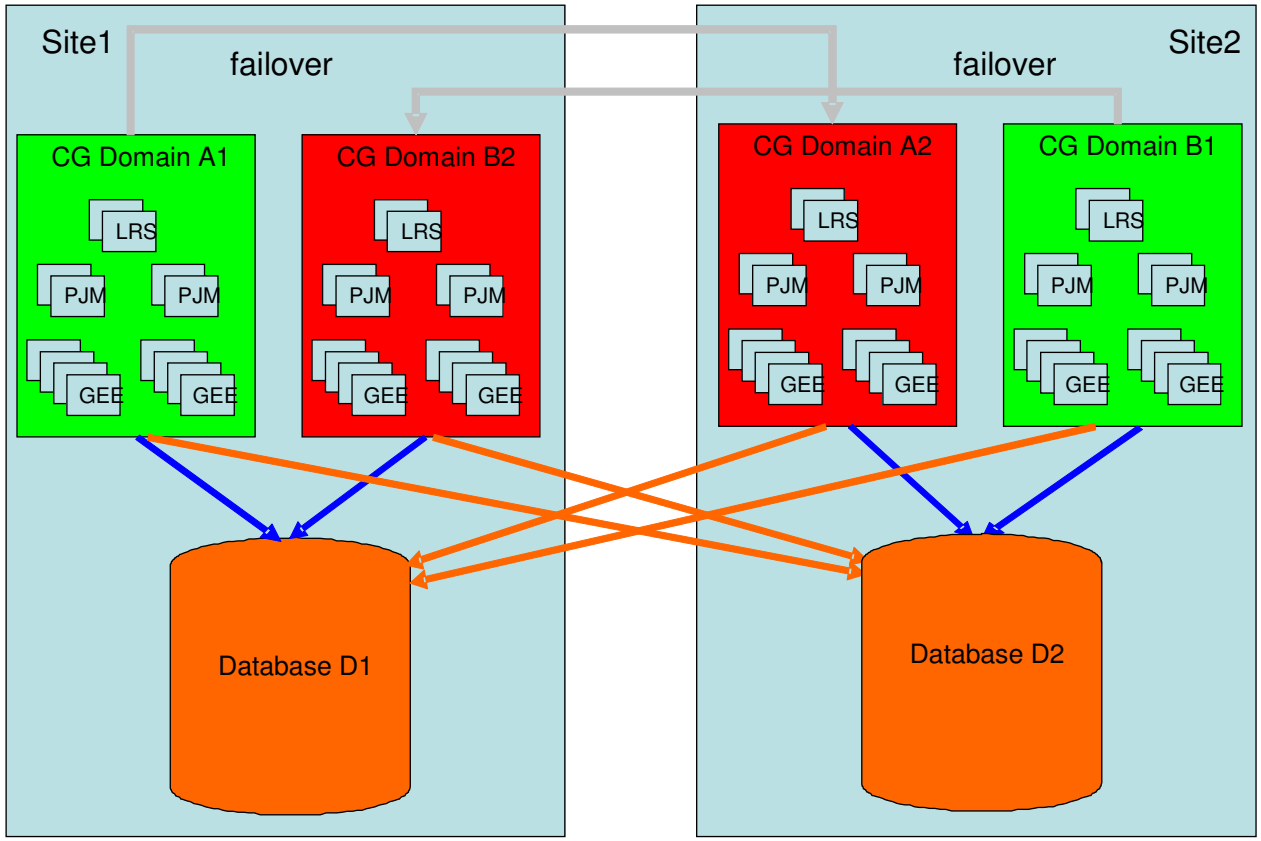
- Clustered Job Scheduler
  - Configure Job Schedulers on clusters
  - Multiple active Job Schedulers (since XD 6.1)
  - Jobs can be managed by any scheduler in your cluster
- Clustered Endpoints
  - Batch applications hosted on clusters
- Network Database
- Shared File System

Disaster Recovery

Disaster Recovery

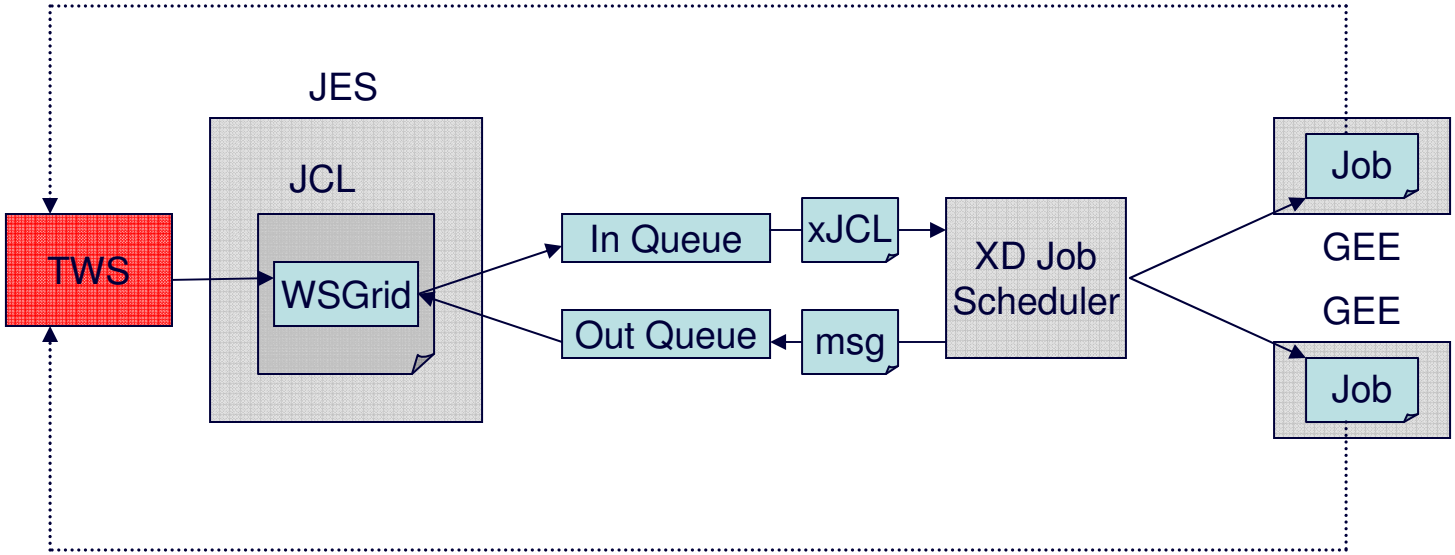
- DR Topology
  - Build separate cells for geographically dispersed sites
  - Limit Compute Grid scheduling domains to endpoints within a cell
  - Use Active/Inactive DR domains
    - Jobs cannot be processed on primary and back domains simultaneously
  - Active/Active DR Topology is through a pair of Active/Inactive DR domains
    - Host backup (inactive) domain on a remote site
- DR Activation Process
  - Use CG provided DR scripts to prepare the inactive domain for takeover
  - Complete takeover by activating the inactive domain

Active/Active Multi-site Disaster Recovery Topology



Enterprise Scheduler Integration

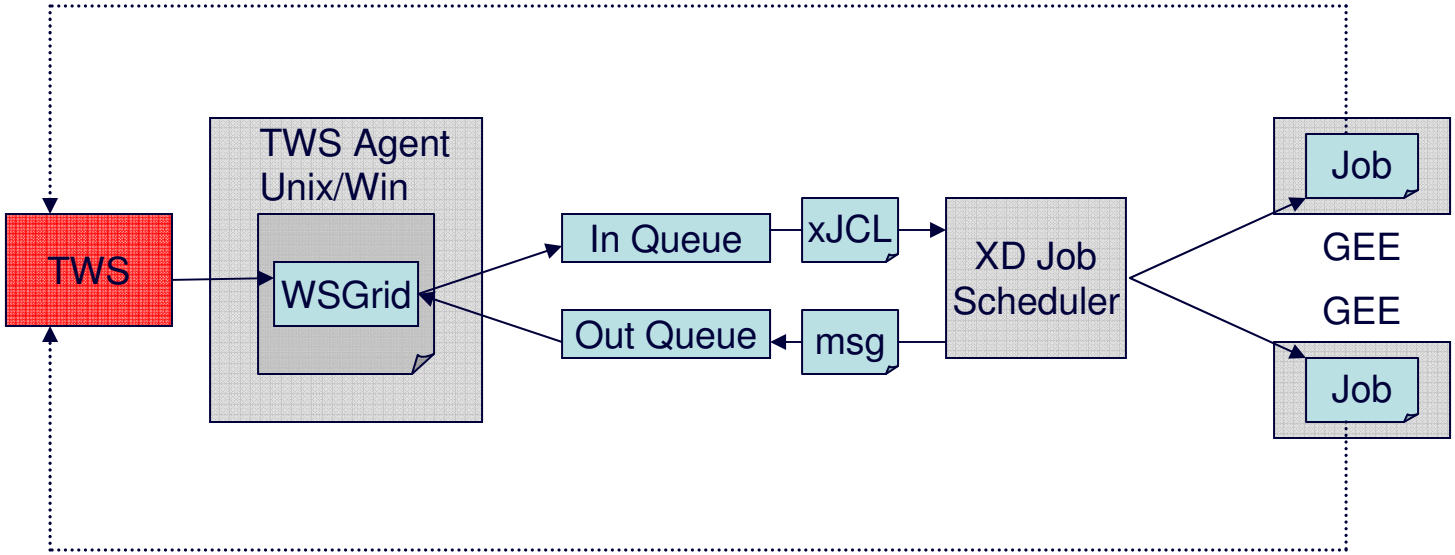
Enterprise Schedulers and XD Compute Grid on z/OS



Dynamic Scheduling

- *TWS is the central enterprise scheduler.*
- *Jobs and commands are submitted from TWS to XD via WSGRID*
- *Jobs can dynamically schedule to TWS via its EJB interface*

Enterprise Schedulers and XD Compute Grid on Distributed

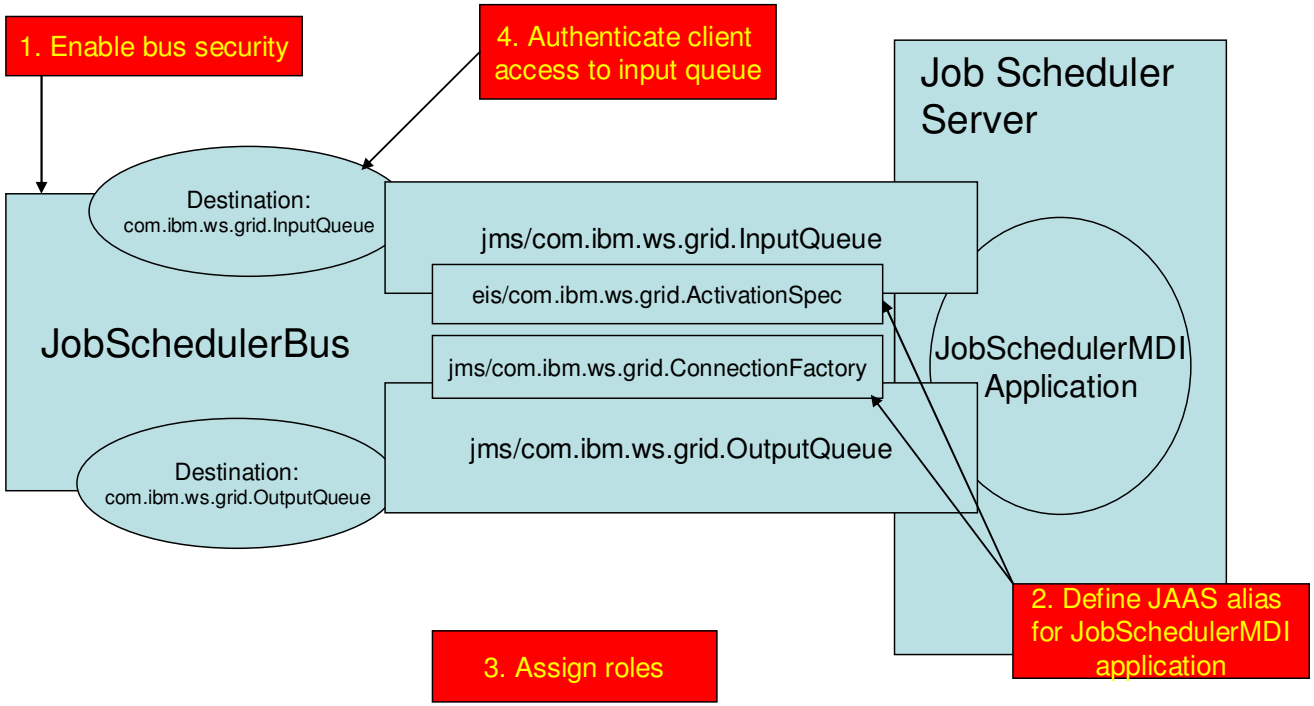


Dynamic Scheduling

- *TWS is the central enterprise scheduler.*
- *Jobs and commands are submitted from TWS to XD via WSGRID*
- *Jobs can dynamically schedule to TWS via its EJB interface*

Securing WSGrid

Securing Job Scheduler Message-Driven Interface



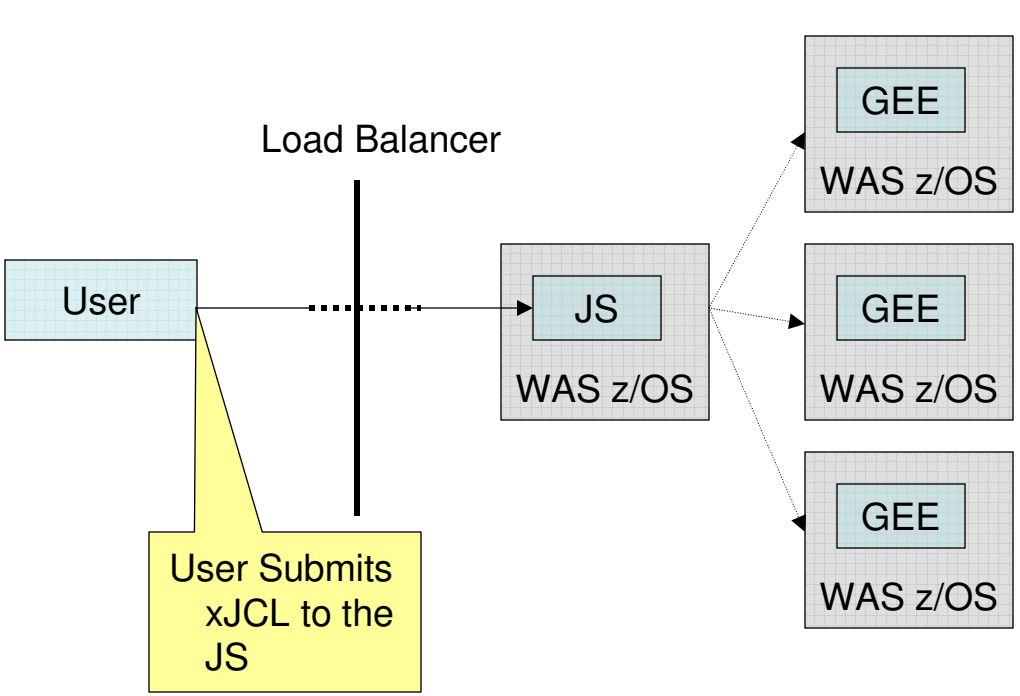
Misc…

Appendix B-

Submitting and Executing Batch Jobs w/ Compute Grid

Submitting and Executing Batch Jobs to XD Compute Grid

Submitting XD Batch Jobs



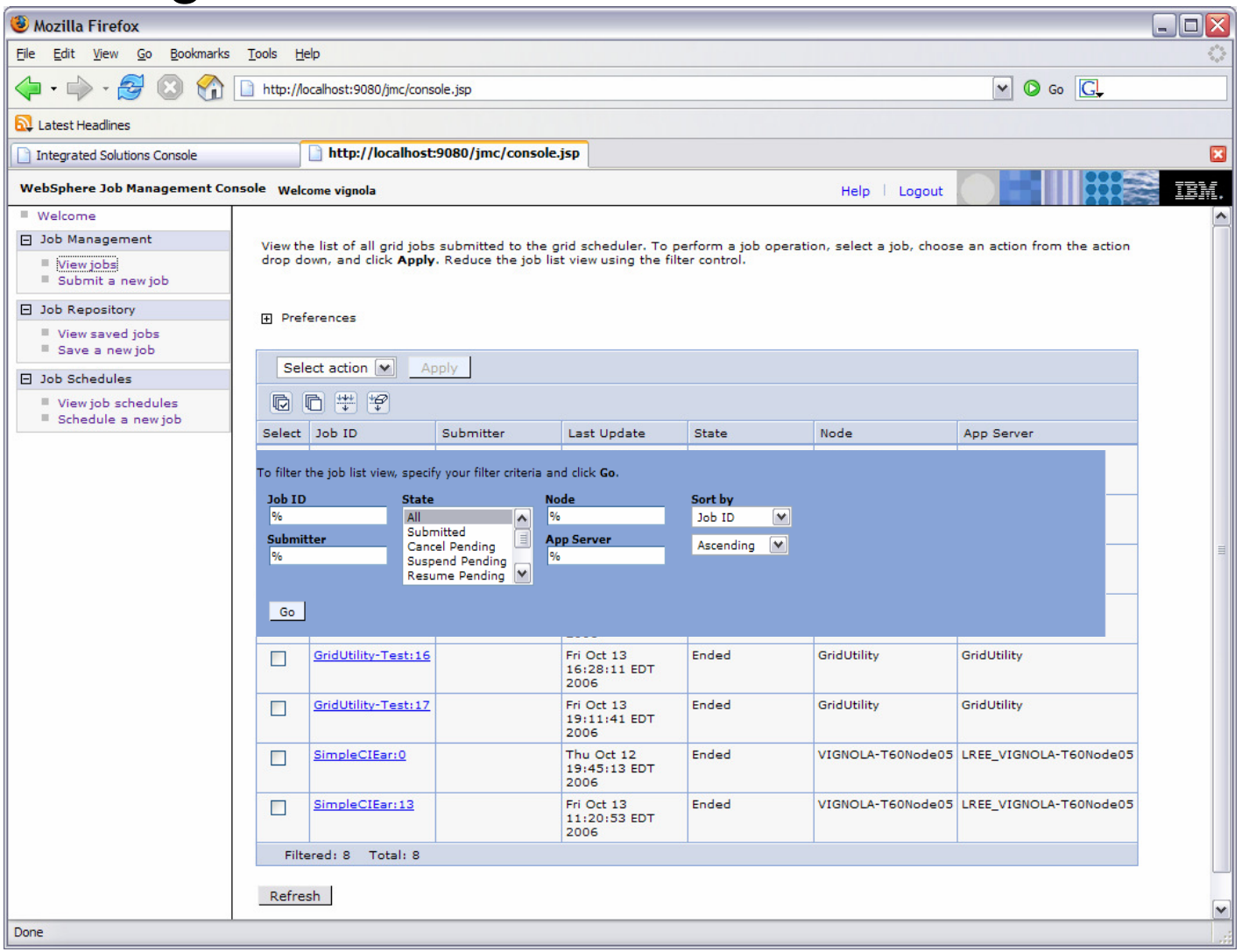
Sample xJCL

```
<job name="SampleJob">
   <jndi-name>batch.samples.sample1</jndi-name>
   <checkpoint-algorithm name="timebased">
        <classname>checkpoints.timebased</classname>
        <props>
              <name="interval" value="15"/>
         </props>
   </checkpoint-algorithm>
   <job-step name="Step1">
     <jndi-name>batch.samples.step1 </jndi-name>
     <checkpoint-algorithm-ref name="timebased"/>
     <batch-data-streams>
          <bds>
              <logical-name> myInput </logical-name>
              <impl-class> bds.sample1 </impl-class>
               <props>
                    <prop name="FILENAME"
                              value="/tmp/input"/>
               </props>
          </bds>
     </batch-data-streams>
    </job-step>
</job>
```

Job Management Console in **XD v6.1**

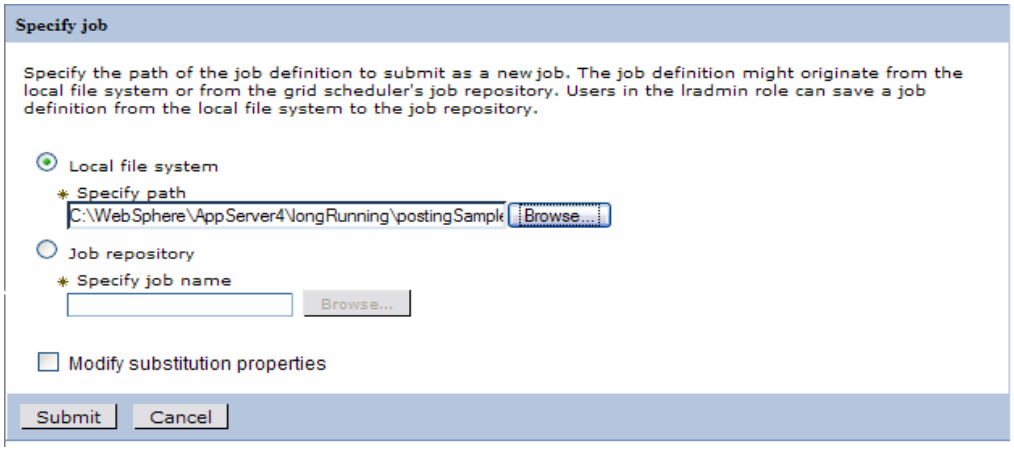
- Web Interface to Scheduler
  - Hosted in same server (cluster) that hosts scheduler function
  - Replaces job management function formerly found in admin console
- Provides essential job management functions
  - job submission
  - job operations
    - cancel, stop
    - suspend, resume
    - restart, purge
  - job repository management
    - save, delete job definitions
  - job schedule management
    - create, delete job schedules
- Security
  - userid/password login
  - lrsubmitter, lrAdmin roles

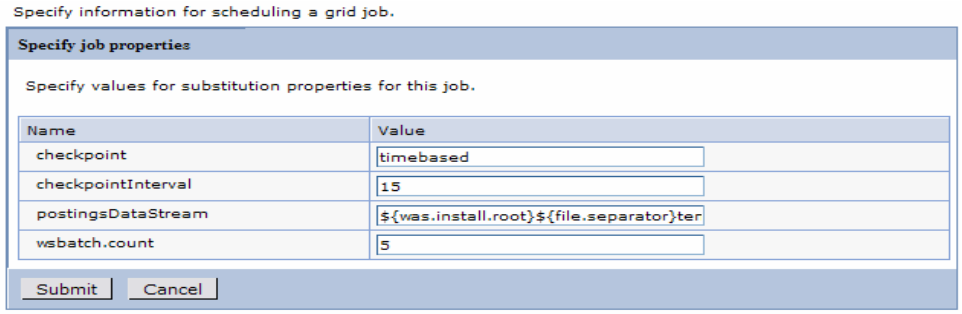
Job Management Console v6.1 – View Jobs



Job Management Console v6.1 - Submit Job

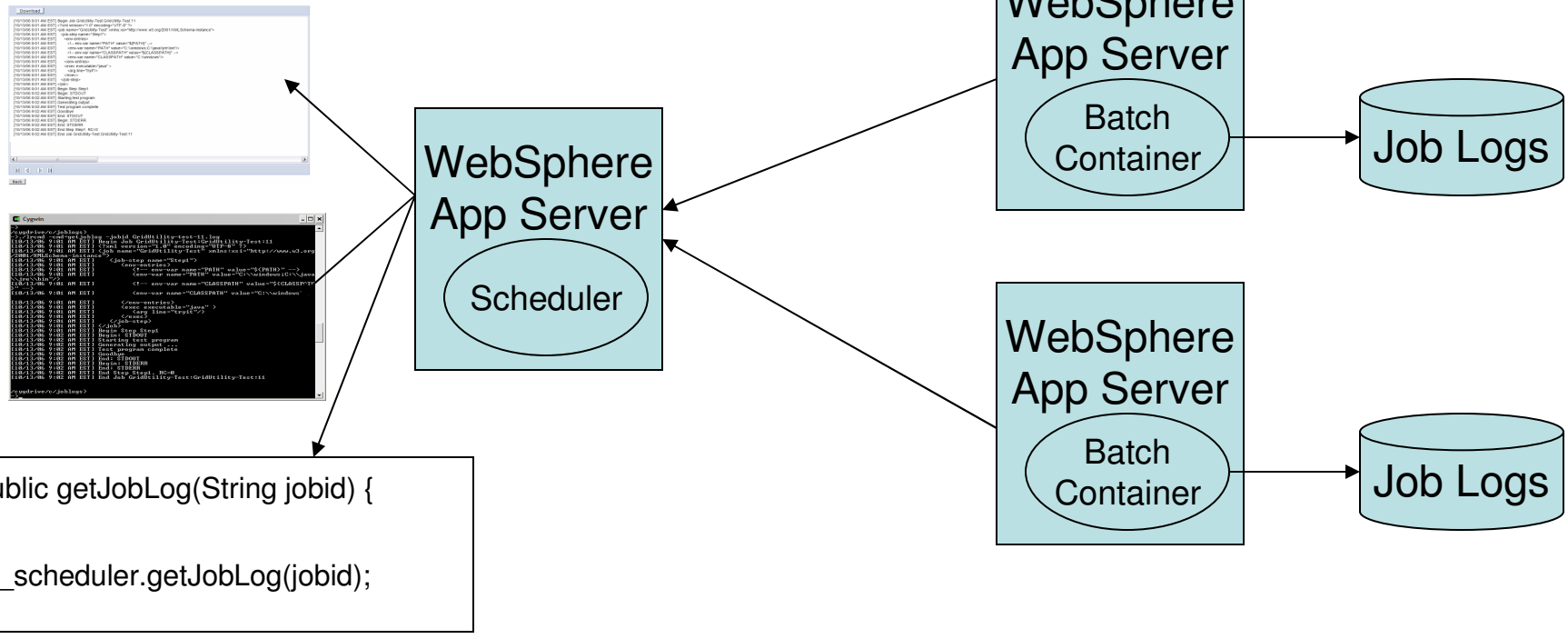
- simple one-click job submission
- job definition source from file system or repository
- optionally review/modify substitution properties
  - property edit page to review/modify substitution property values





Job Logs in v6.1

- Stored in file system local to batch container
- Remote access via all scheduler APIs (console, lrcmd, APIs)



Job Logs in v6.1 …

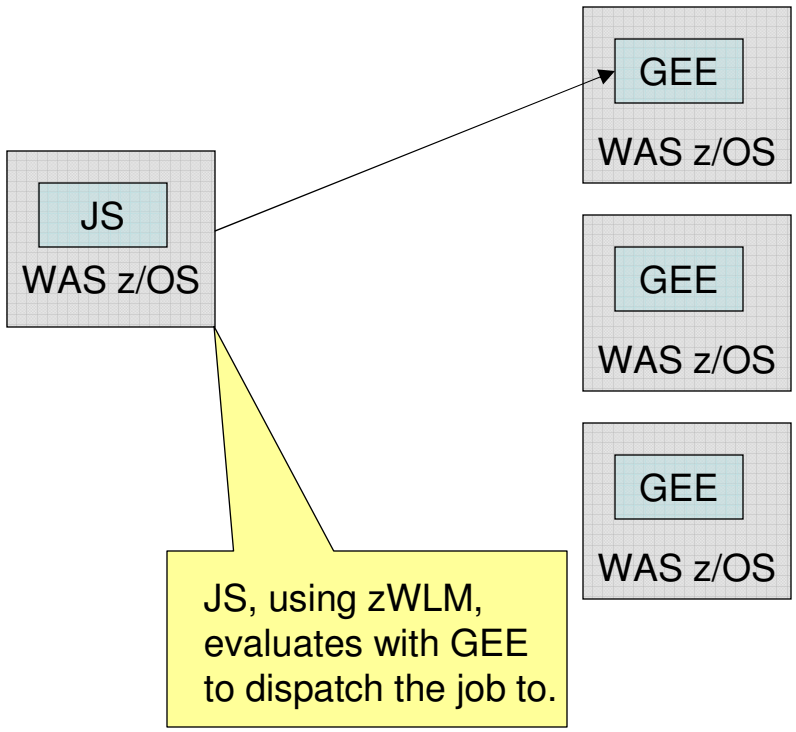
This panel shows the log of grid job GridUtility-Test:11. To save it on the local file system, click **Download**.

Download

```
[10/13/06 9:01 AM EST] Begin Job GridUtility-Test:GridUtility-Test:11
[10/13/06 9:01 AM EST] <?xml version="1.0" encoding="UTF-8" ?>
[10/13/06 9:01 AM EST] <job name="GridUtility-Test" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
[10/13/06 9:01 AM EST]    <job-step name="Step1">
[10/13/06 9:01 AM EST]      <env-entries>
[10/13/06 9:01 AM EST]        <!-- env-var name="PATH" value="${PATH}" -->
[10/13/06 9:01 AM EST]        <env-var name="PATH" value="C:\\windows;C:\\java\\jre\\bin"/>
[10/13/06 9:01 AM EST]        <!-- env-var name="CLASSPATH" value="${CLASSPATH}" -->
[10/13/06 9:01 AM EST]        <env-var name="CLASSPATH" value="C:\\windows"/>
[10/13/06 9:01 AM EST]      </env-entries>
[10/13/06 9:01 AM EST]      <exec executable="java" >
[10/13/06 9:01 AM EST]        <arg line="tryit"/>
[10/13/06 9:01 AM EST]      </exec>
[10/13/06 9:01 AM EST]    </job-step>
[10/13/06 9:01 AM EST] </job>
[10/13/06 9:01 AM EST] Begin Step Step1
[10/13/06 9:02 AM EST] Begin: STDOUT
[10/13/06 9:02 AM EST] Starting test program
[10/13/06 9:02 AM EST] Generating output ...
[10/13/06 9:02 AM EST] Test program complete
[10/13/06 9:02 AM EST] Goodbye
[10/13/06 9:02 AM EST] End: STDOUT
[10/13/06 9:02 AM EST] Begin: STDERR
[10/13/06 9:02 AM EST] End: STDERR
[10/13/06 9:02 AM EST] End Step Step1. RC=0
[10/13/06 9:02 AM EST] End Job GridUtility-Test:GridUtility-Test:11
```

Back

Dispatching Batch Job

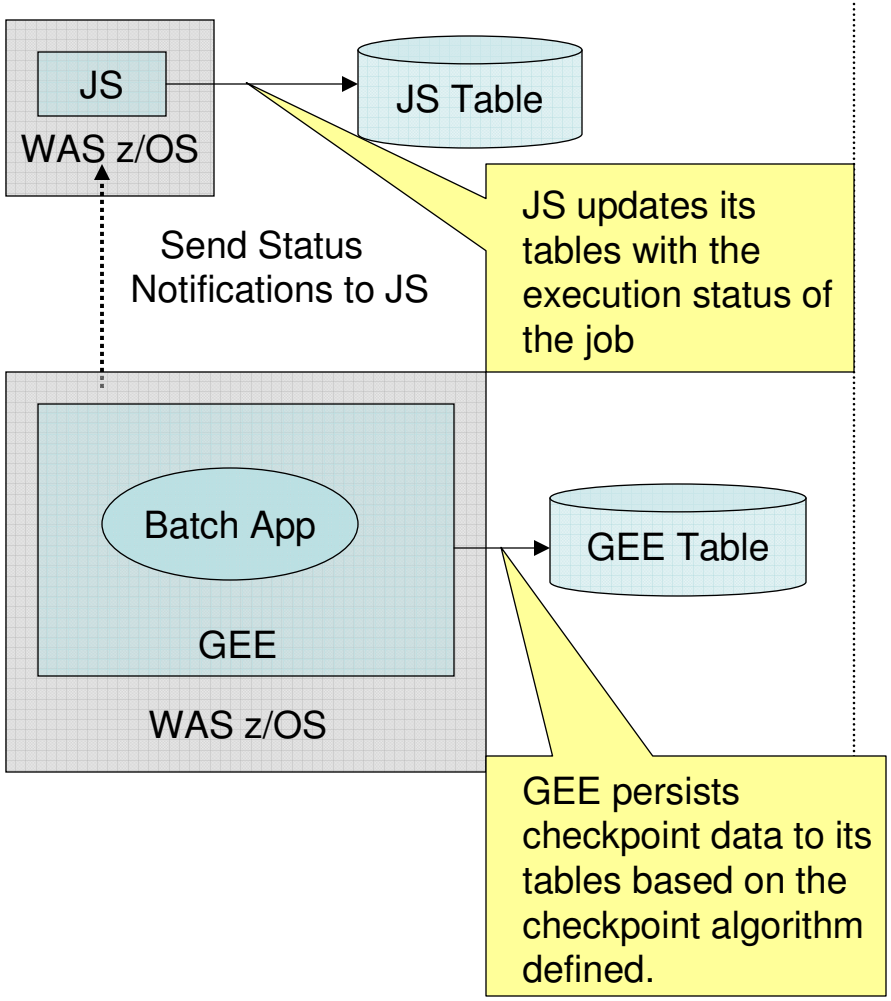


GEE is a J2EE application that runs within a WebSphere z/OS Application Server **Servant Region**. Therefore servant-region behavior is applied to batch jobs- if workload and service policies deem it necessary, new servants can be dynamically started or stopped.

In WebSphere XD 6.1, new SPI's are introduced that allow more control over how a batch job is dispatched: This can be used to:

- Override the chosen GEE target
- force some authorization to take place first
- Assign a specific job class (which maps to some zWLM service policy
- force the job to be scheduled for execution at a later time.
- **completely pluggable and customizable**

Executing Batch Jobs



The GEE executes the batch application 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.

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

Note that in XD 6.1 the JS will provide WS-Notifications so non XD components can register as listeners for status on a particular job. In addition there are updates to the manner in which job logs are managed.

Job Classes in v6.1

- Administrative control over resource consumption
- Defined through Grid Scheduler configuration in WAS admin console
- Named policies that control
  - maximum execution time
  - maximum number of concurrent jobs per endpoint (logical batch container)
  - maximum job log size
  - job log retention (age, space)
  - execution record retention (age, number)
- Assigned via class= keyword in xJCL
- Can be overridden by JobClass SPI

Classification Rules in v6.1

- Administrative rules for service policy assignment
- Defined through Grid Scheduler configuration in WAS admin console
- Rules are cell-wide ordered list
- Evaluated in specified order
- First match assigns service policy
- Rules are boolean expression formed using following operands:
  - job name
  - job class
  - submitter identity, group
  - application type (j2ee, utility)
  - time, date
  - platform (e.g. z/OS)

Usage Accounting in v6.1

- Two options
  - Stored in scheduler database. With MBean interface to do CSV format export to file (importable to Tivoli ITUAM)
  - New SMF Record 120 Subtype
  - May choose either or both on z/OS
- Data gathered per job
  - job name
  - job id
  - time/Date (start/end)
  - submitter
  - accounting number
  - cell/node/server names
  - CPU time

xJCL Substitution Properties in v6.1

- Ant-style substitution - ${<*property-name*>}
- Optional default settings

```
<substitution-props>
    <prop name="wsbatch.count" value="5" />
</substitution-props>
```

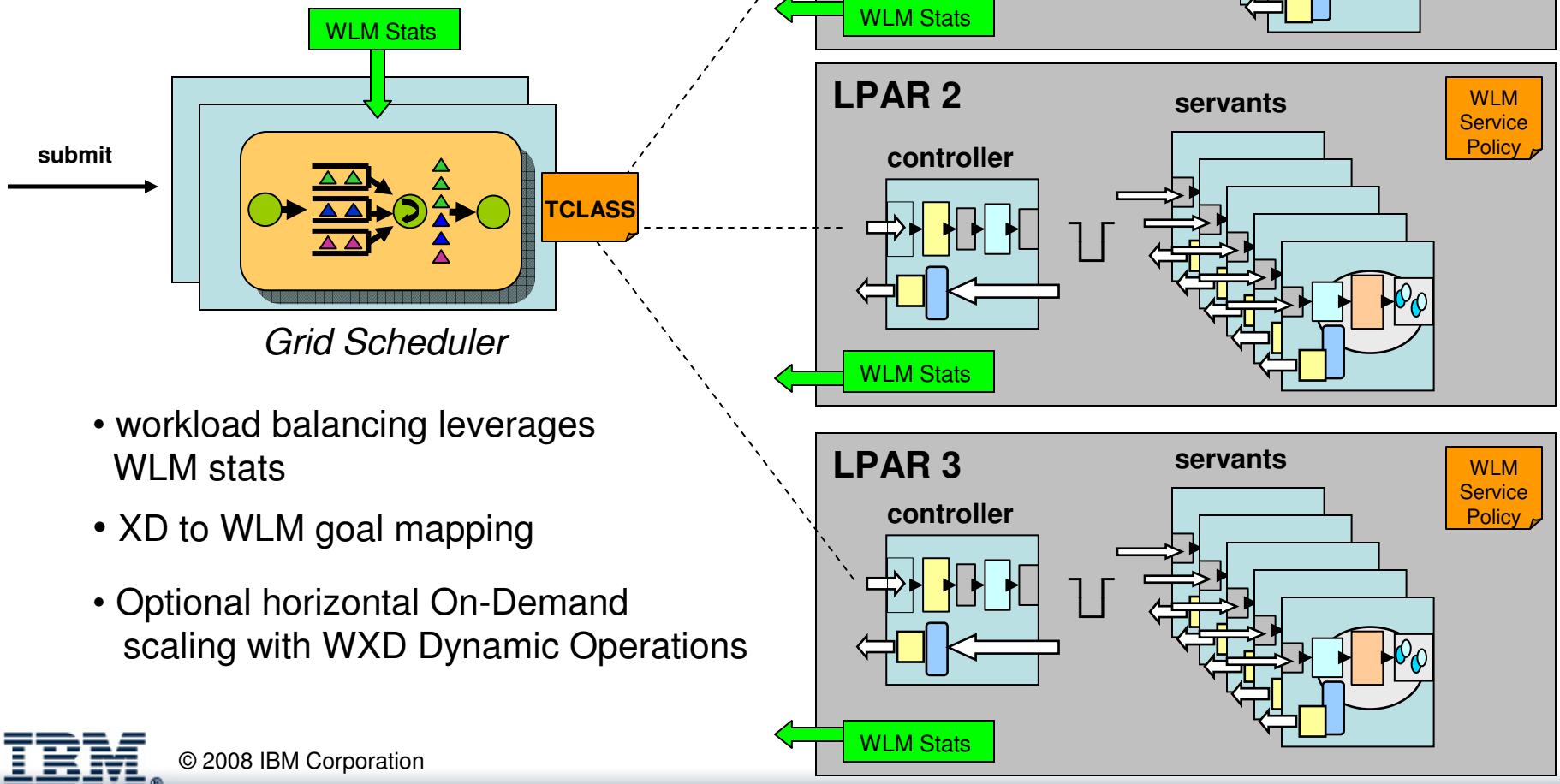
- Substitution property values specifiable on all job submission interfaces:
  - job management console
    - submission
    - scheduling
  - lrcmd
  - scheduler APIs

Compute Grid z/OS Integration

- SMF accounting records for J2EE batch jobs
  - SMF 120 (J2EE) records tailored to jobs
  - Record includes: job id, user, CPU time
- Dynamic Servants for J2EE batch job dispatch
  - XD v6.0.1 uses pre-started servants (min=max, round-robin dispatch)
  - New support will exploit WLM to start new servants to execute J2EE batch jobs on demand
- Service policy classification and delegation
  - New classification critieria, including: jobname, submitter, jobclass
  - leverage XD classification to select z/OS service class by propagating transaction class from Grid Scheduler to z/OS app server for job registration with WLM

Compute Grid z/OS Integration …

- Dynamic servants for vertical On-Demand scaling

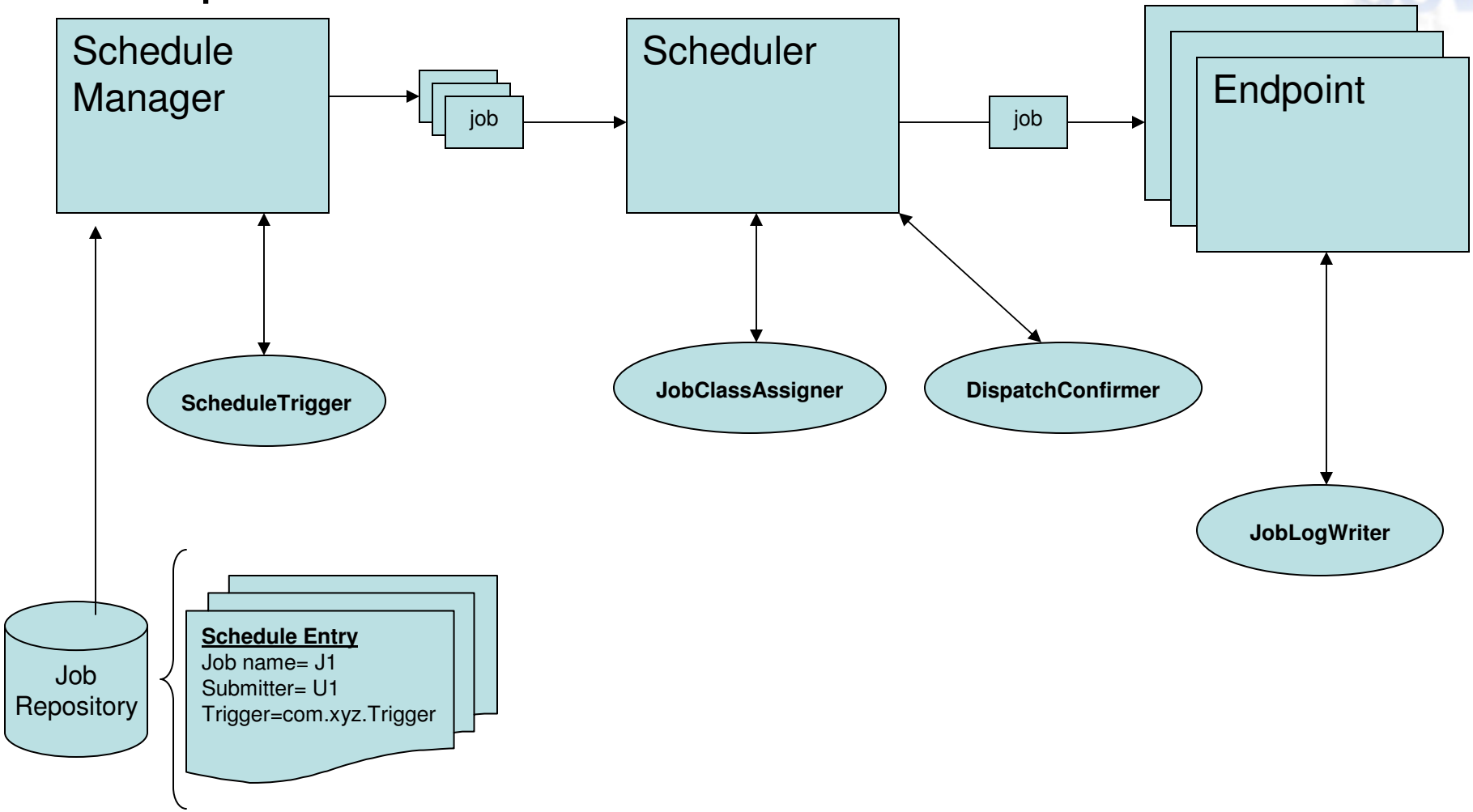


- workload balancing leverages WLM stats
- XD to WLM goal mapping
- Optional horizontal On-Demand scaling with WXD Dynamic Operations

Compute Grid SPIs

- ScheduleTrigger
  - custom rule for submitting a scheduled job
  - allows for submission conditions beyond time/date
- JobClassAssigner
  - custom logic for validating job class usage
  - allows for override of user-specified job class
- DispatchConfirmer
  - custom logic for confirming XD dispatch decision
  - multiple actions possible:
    - cancel job
    - re-queue job for later dispatch
    - specify target endpoint
- JobLogInterceptor
  - allows for modification of job log content
    - Edit
    - Delete
    - System log vs job log only
  - Can replace file-based logs with alternative destination
- JobNotificationListener
  - Receives notifications for key job lifecycle events:
  - job start/end
  - step start/end

Compute Grid SPIs …



Appendix C – Some Compute Grid Use-cases

Overview

- XD Compute Grid Use-cases
  - **Batch Modernization**
  - Highly parallel batch jobs
  - Dynamic OLTP and Batch infrastructure
  - Batch as a service
  - Replacing existing java batch frameworks
  - Sharing business logic across OLTP and Batch

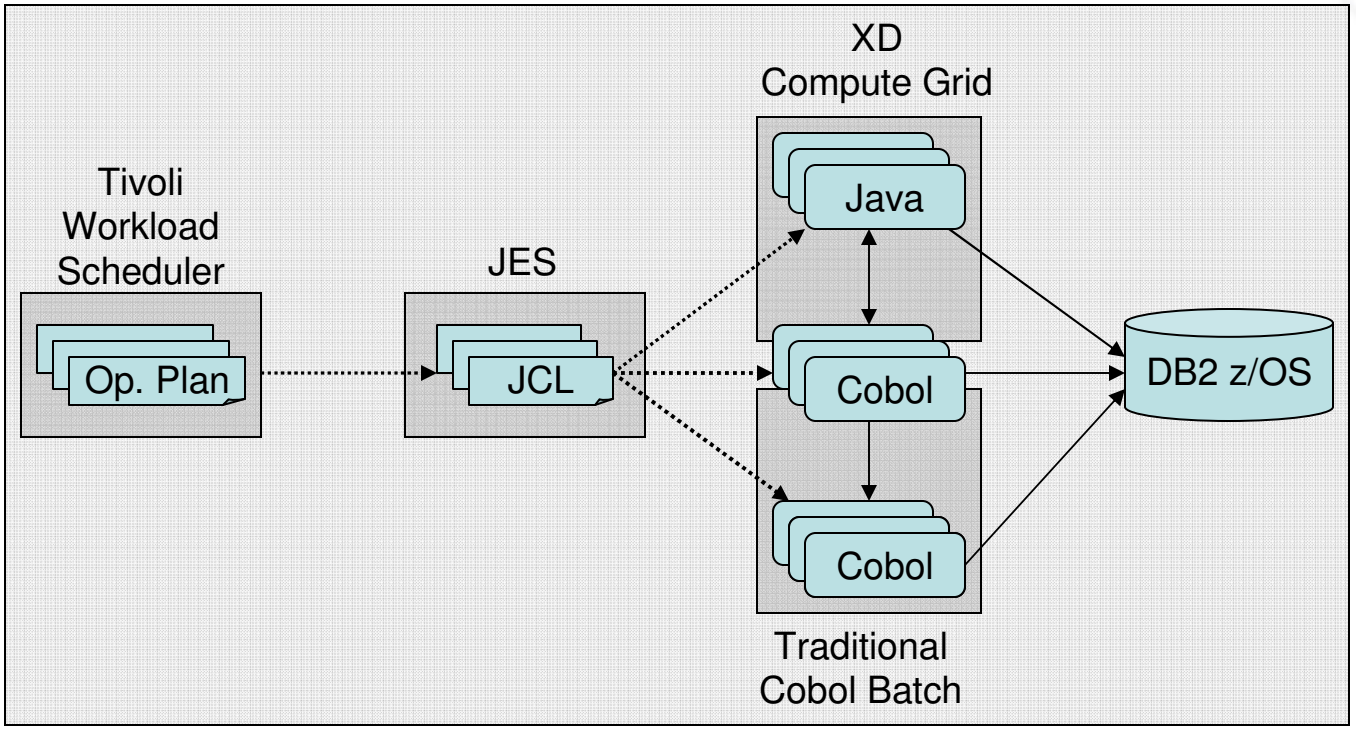
Batch Modernization Use-case

- **Motivations for modernization**
  - IT departments are challenged to absorb tremendous growth rates while executing with a constant IT budget

- **Primary strategic goals for batch modernization**
  1. No loss of performance.
     Can be achieved with: JIT compilers in Java, parallelization, caching, etc.
  2. No loss of qualities of service such as job restart, availability, security
  3. Reduced operations costs. Primarily delivered through zAAP processors

- **Secondary strategic goals**
  1. A more agile runtime infrastructure that can better tolerate future changes
  2. Common development, testing, deployment, security, and production management processes and tooling across OLTP and Batch

Batch Modernization with XD Compute Grid



Today: Executing tradition batch with Cobol

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

Phase 2: Share existing Cobol modules across both batch domains

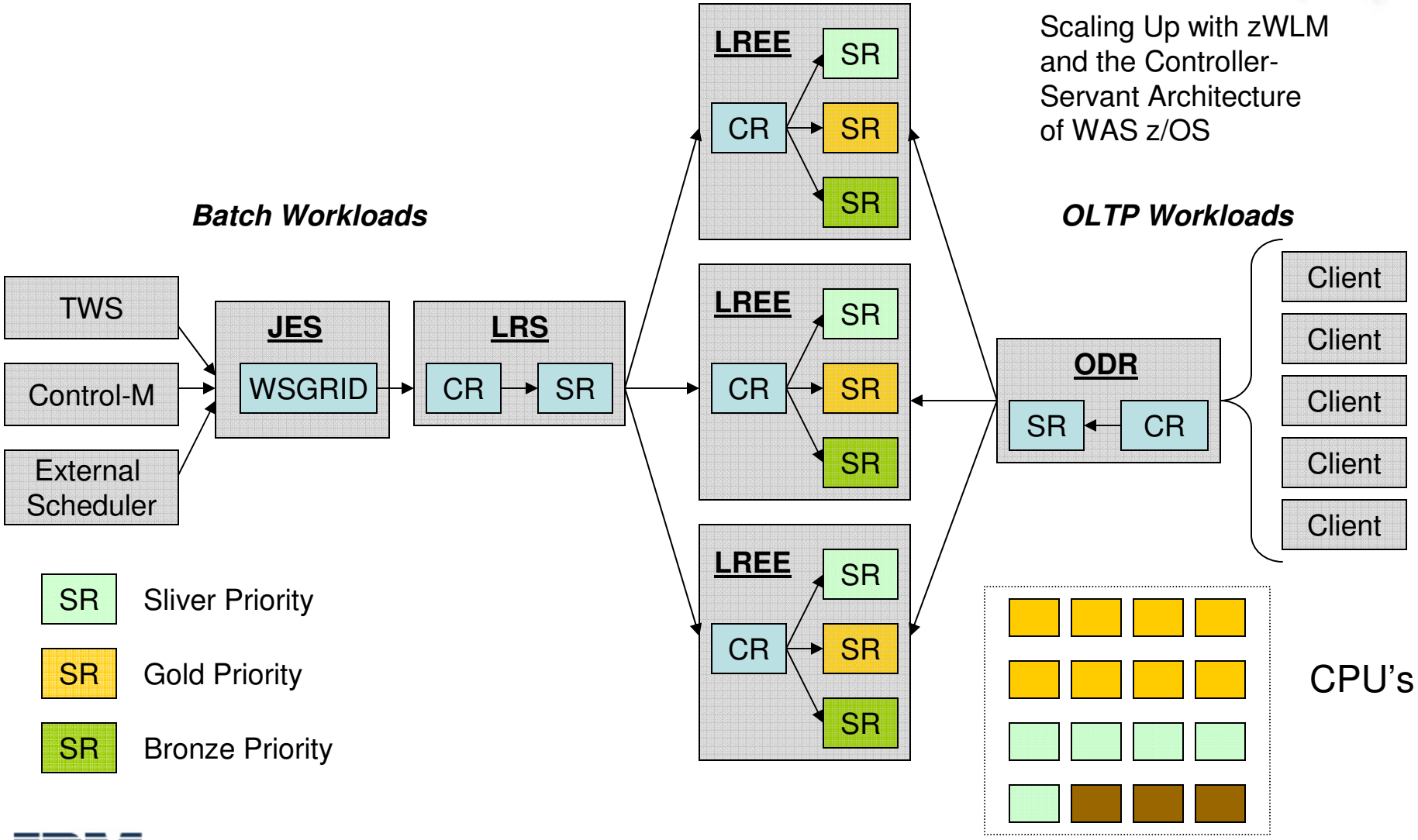
Phase 3: Incrementally migrate remaining Cobol Modules to Java with XD Compute Grid

Completion: All Cobol batch modules are replaced with java and are running in XD Compute Grid

Overview

- XD Compute Grid Use-cases
  - Batch Modernization
  - Highly parallel batch jobs
  - **Dynamic OLTP and Batch infrastructure**
  - Batch as a service
  - Replacing existing java batch frameworks
  - Sharing business logic across OLTP and Batch

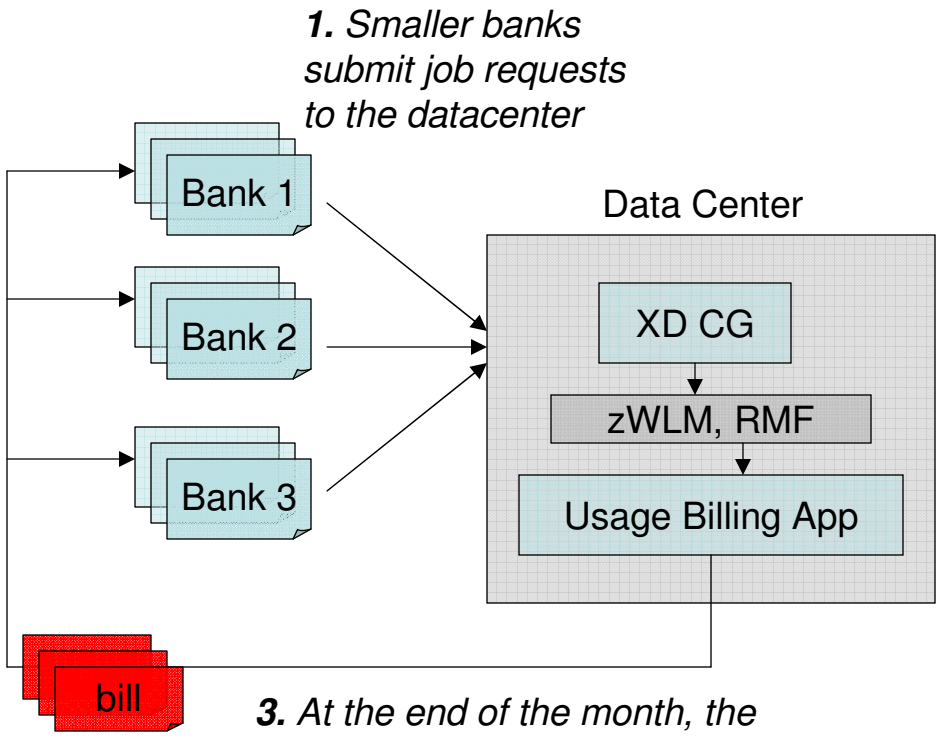
Dynamic OLTP and Enterprise Grid Runtime



Overview

- XD Compute Grid Use-cases
  - Batch Modernization
  - Highly parallel batch jobs
  - Dynamic OLTP and Batch infrastructure
  - **Batch as a service**
  - Replacing existing java batch frameworks
  - Sharing business logic across OLTP and Batch

Batch as a service

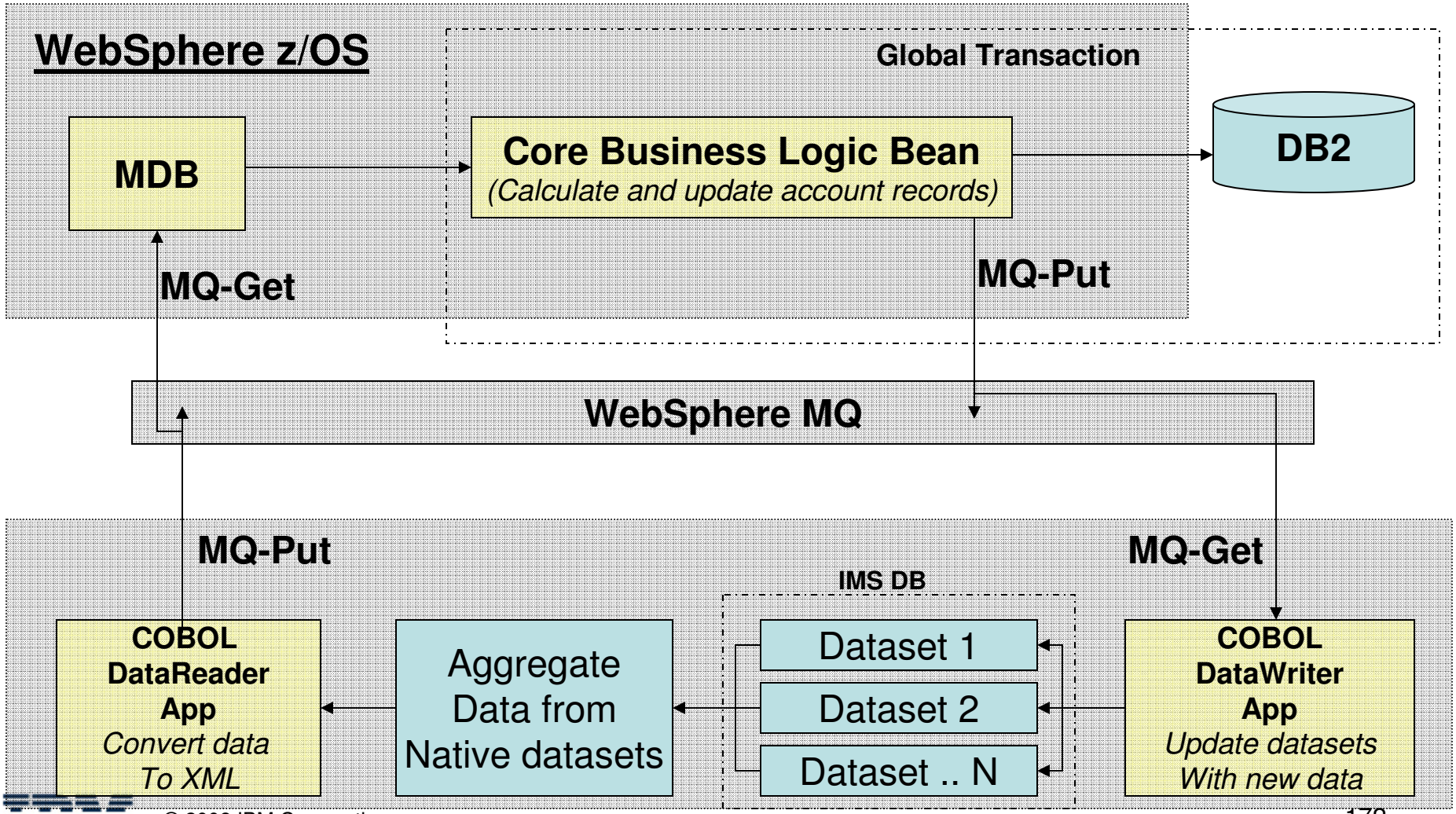


Overview

- XD Compute Grid Use-cases
  - Batch Modernization
  - Highly parallel batch jobs
  - Dynamic OLTP and Batch infrastructure
  - Batch as a service
  - **Replacing existing java batch frameworks**
  - Sharing business logic across OLTP and Batch

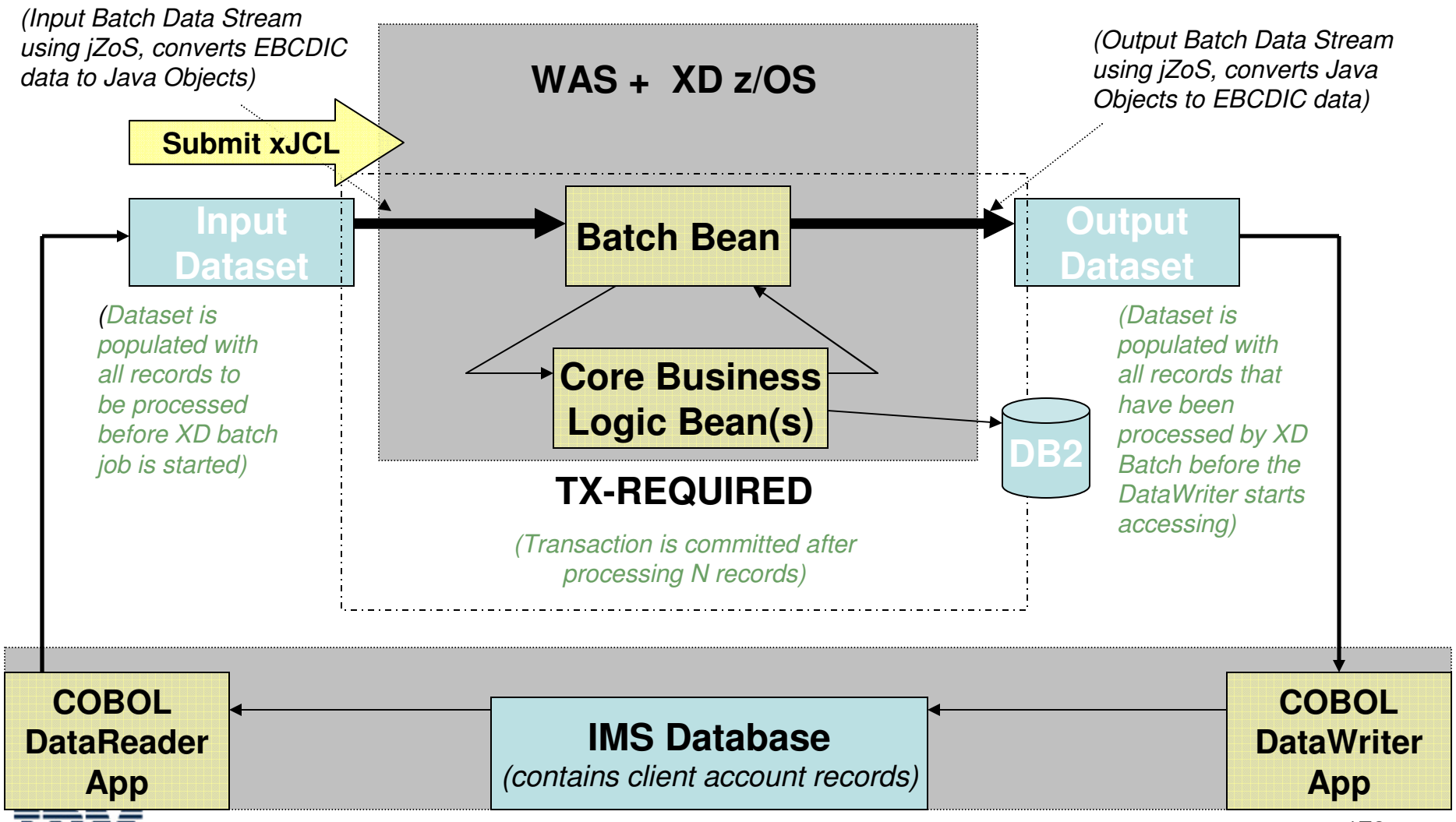
An Example: Java Batch Pattern (z/OS) - Good

The home-grown solution required many components to be integrated together…



An Example: Java Batch Pattern (z/OS) - Better

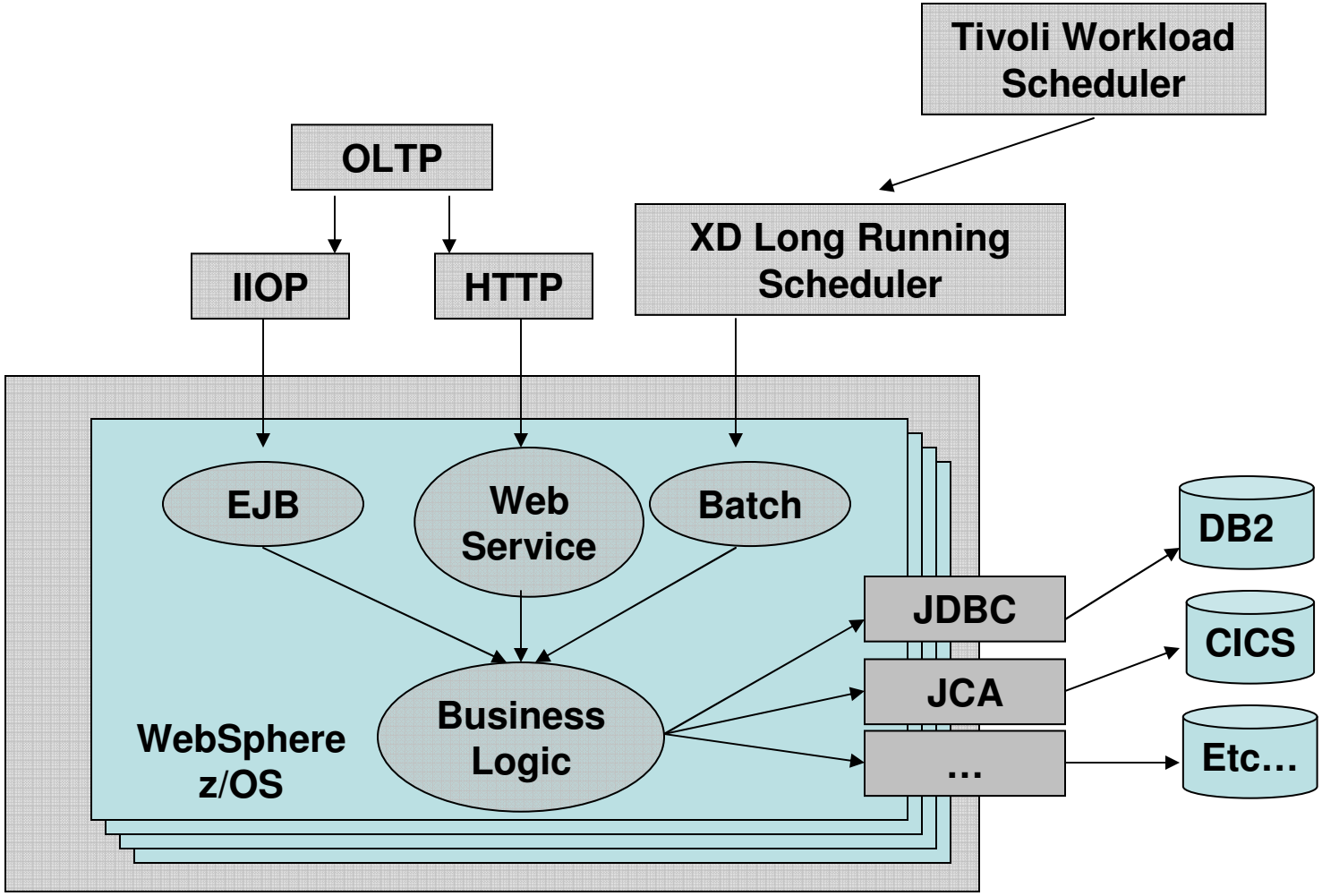
WebSphere XD streamlines the Java-centric batch processing on z/OS…



Overview

- XD Compute Grid Use-cases
  - Batch Modernization
  - Highly parallel batch jobs
  - Dynamic OLTP and Batch infrastructure
  - Batch as a service
  - Replacing existing java batch frameworks
  - **Sharing business logic across OLTP and Batch**

Running Mixed Workloads- OLTP and Batch



XD Compute Grid Value Proposition

- Delivers a zAAP-eligible enterprise java batch execution environment built on WebSphere for z/OS
- Enables the incremental migration of COBOL to Java thereby reducing the risks associated with a batch modernization project
- Integrates with existing enterprise batch schedulers such as TWS, CA7, Control-M, Zeke to help deliver a robust, cost-effective, WebSphere-based batch execution environment
- Enables new execution patterns including: Dynamic OLTP and Batch runtime environment built on WebSphere for z/OS; highly parallel batch jobs; and many others.
- Integrates with the overall SOA strategy of reuse by enabling one to share business logic across both the OLTP and Batch worlds
- Delivers high-performance batch processing by leveraging the System-z, z/OS, and WAS z/OS performance optimizations gained when executing within close proximity of the data.

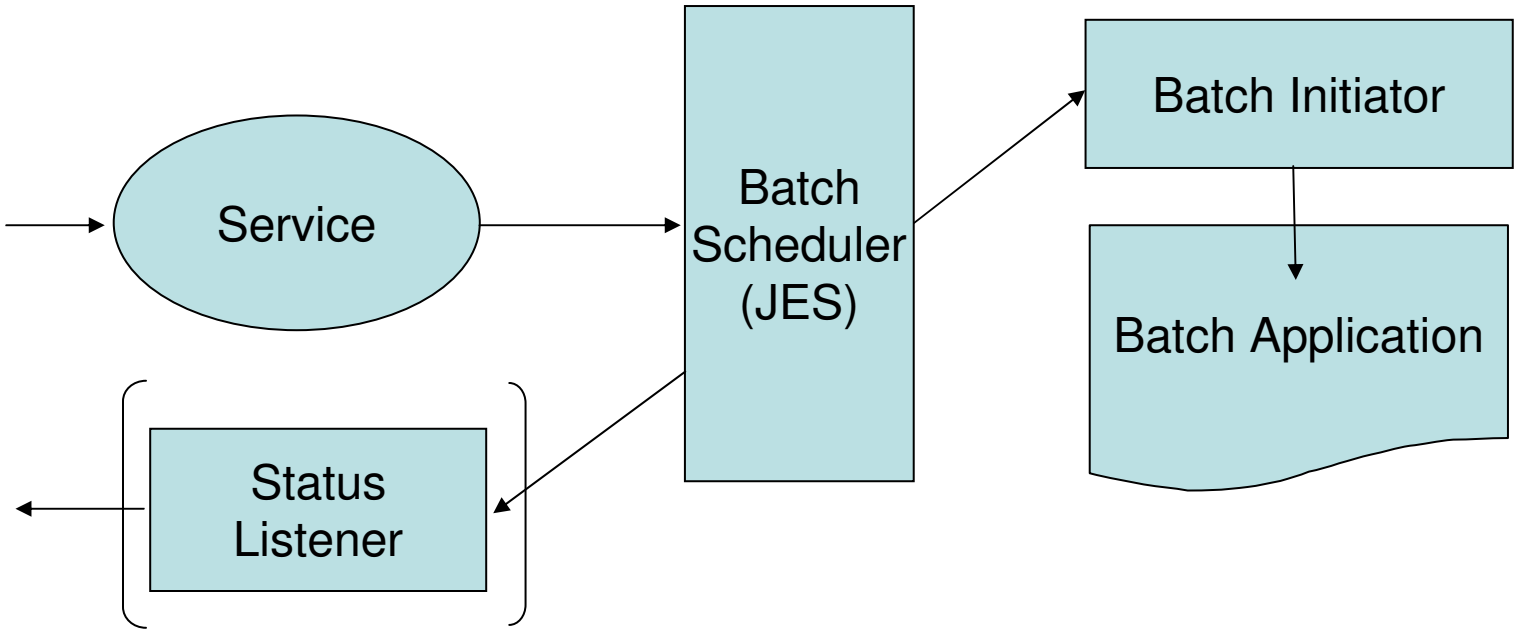
Appendix D-
Approaches for Batch Modernization

Approaches

- Enablement
  - SOA wrappers over existing jobs (submit, monitor, output)

SOA Batch Adapter (Enablement)

- Service Interface for Batch Submission, Monitoring, Output



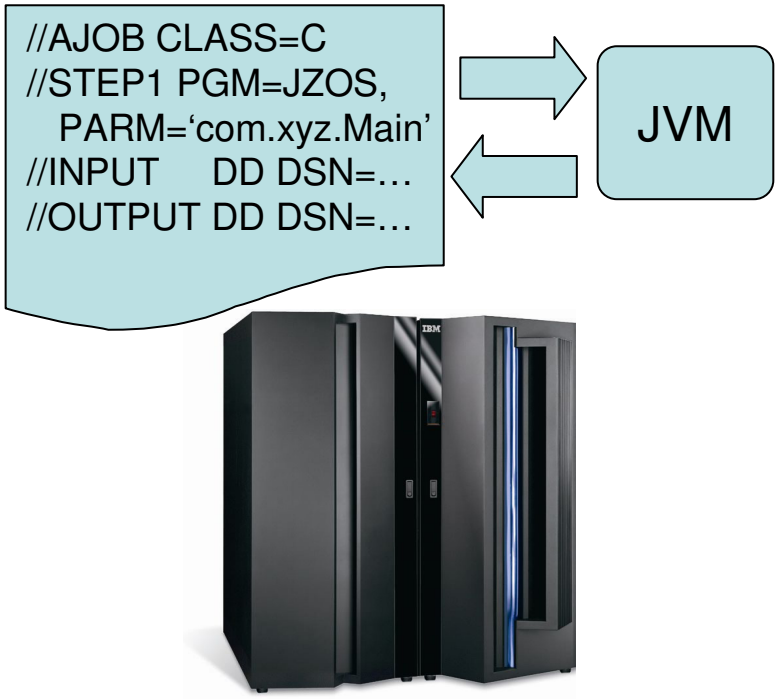
Approaches

- Enablement
  - SOA wrappers over existing jobs (submit, monitor, output)
- Enrichment
  - Infusion of new technology into existing jobs (e.g. JZOS, service composition)

JZOS – Java Batch for z/OS (Enrichment)

- Java as a batch language
- Launcher & toolkit
- z/OS only

- J2SE environment
- Free – include in z/OS JDK
- Costly execution



Approaches

- Enablement
  - SOA wrappers over existing jobs (submit, monitor, output)
- Enrichment
  - Infusion of new technology into existing jobs (e.g. JZOS, service composition)
- Evolution
  - Re-architecting/re-targeting traditional batch to execute at lower cost in a more agile environment

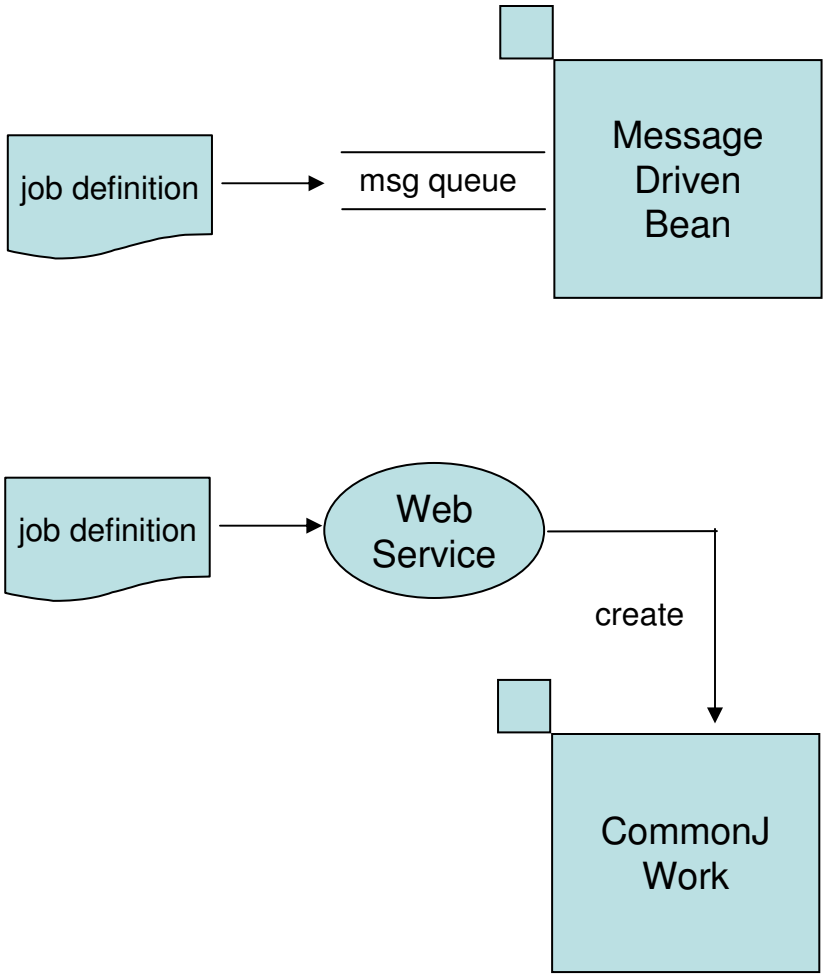
“Maverick” Batch Environment (Evolution)

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

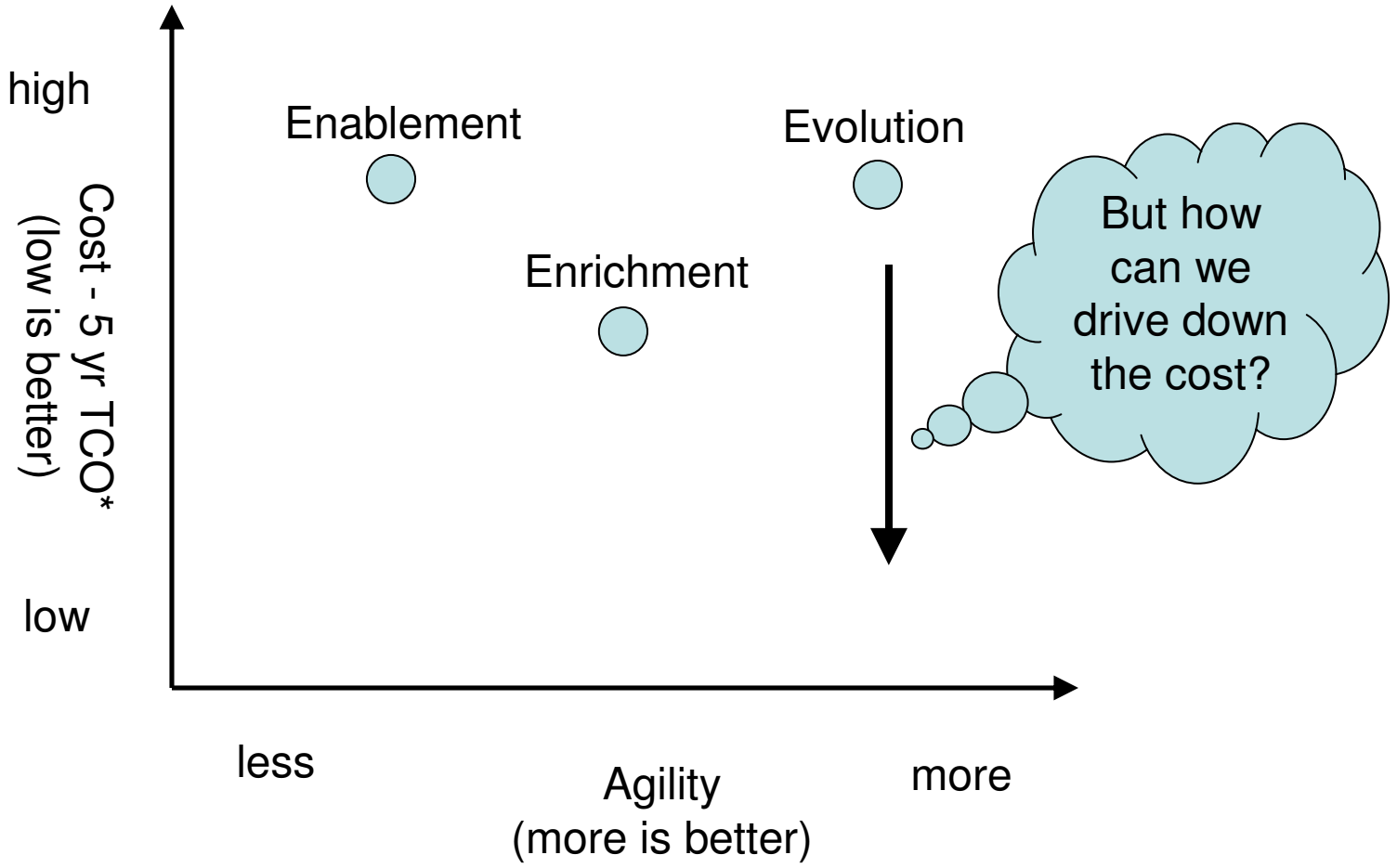
But …

- 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

…



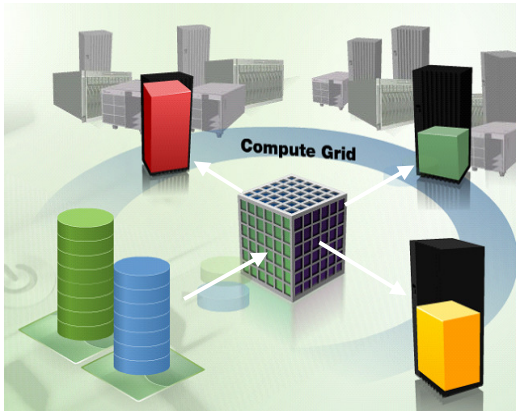
How do Approaches Compare?



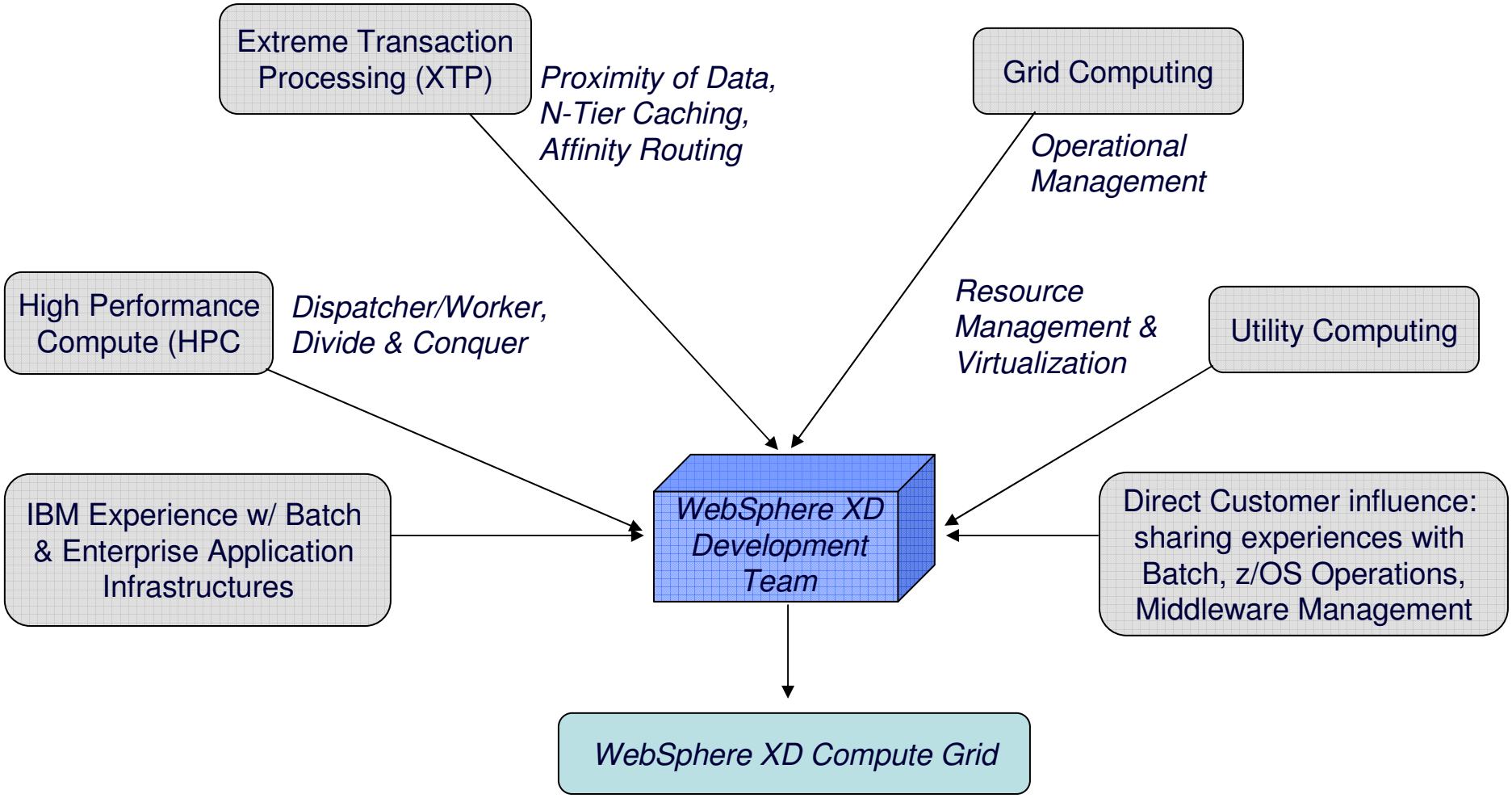
* hypothetical

Satisfying the Requirements: Why Compute Grid?

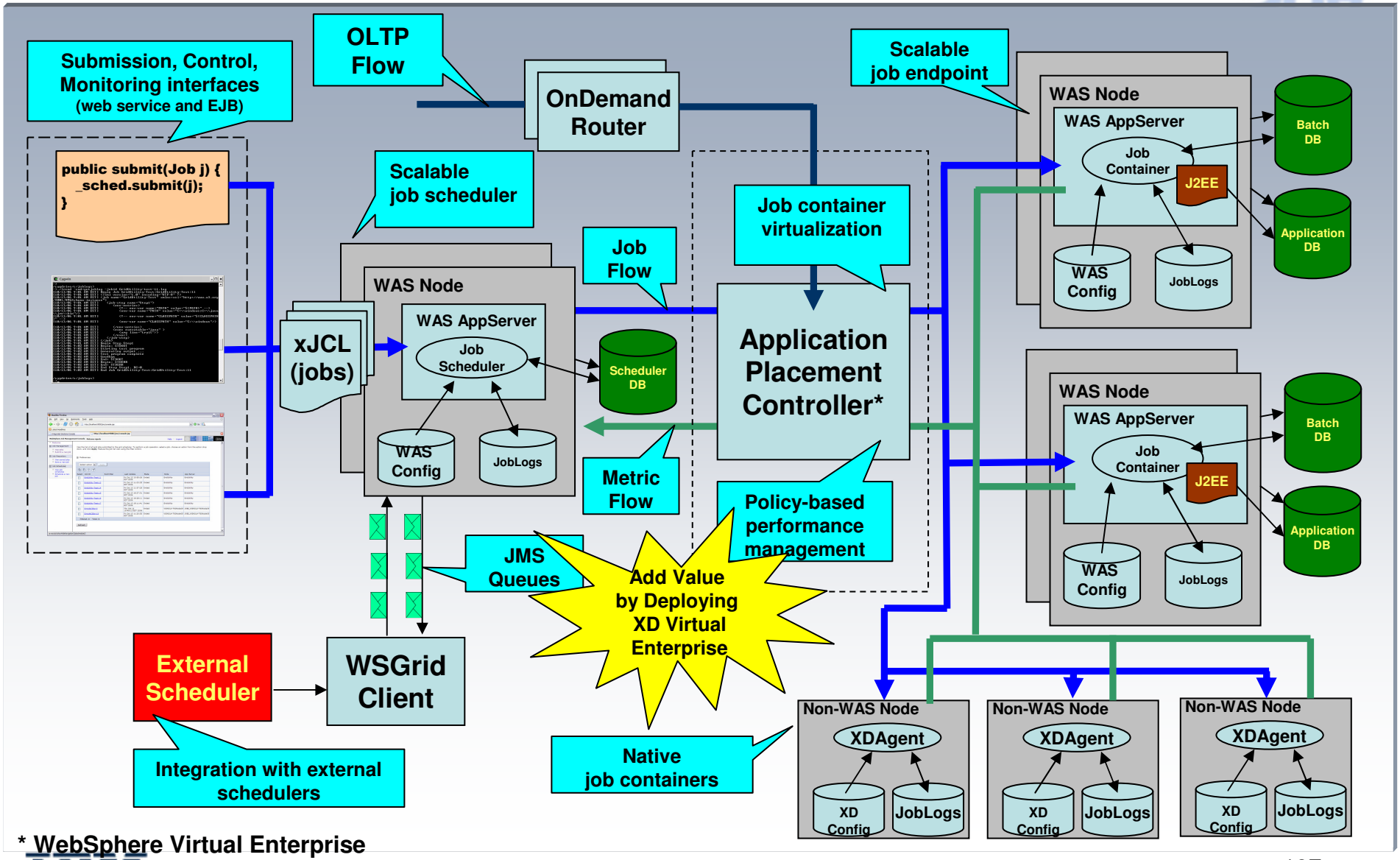
- Most complete Java batch solution available!
- Easy, modern batch programming model & tools
  - POJO programming, light-weight Eclipse-based tooling
- Modern batch processing system (infrastructure)
  - Job entry scheduler, operational controls, container managed checkpoint/restart, workload management, scalability/availability,
  - Job console, enterprise scheduler integration (e.g. TWS, Control-M, etc)
  - Highly Parallel and Extreme Batch support.
- Practices and Proof
  - IBM Services fully enabled to teach/assist
  - Proven track record with production deployments



Origins of WebSphere XD Compute Grid

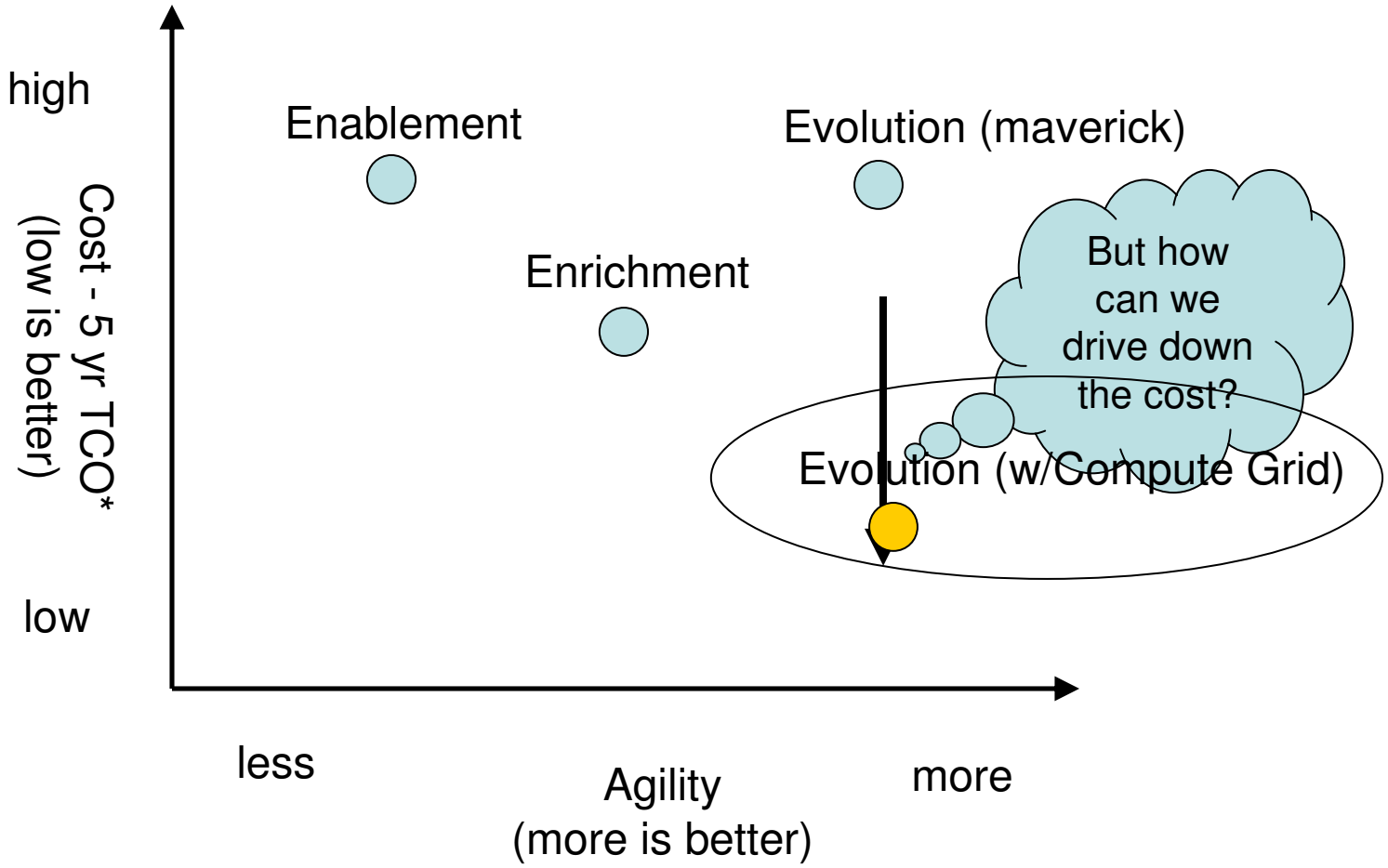


Anatomy of Compute Grid Job Processing Environment



* WebSphere Virtual Enterprise

How do Approaches Compare?



* hypothetical

Customer Experience's with Batch Modernization

Swiss Re

- The History
- The Business Case
- The Project Approach
- The Results
- Next Steps

IBM and Swiss Re: A Mainframe Success Story

Swiss Re

- One of the earliest buyers of the IBM 650 (1955)
- Adoption of DB2 when product was in first beta release (1986)
- One of the earliest System z10 adopters (2/2008)
- Early adopter of WebSphere XD for z/OS

Swiss Re

Swiss Re's Batch Environment

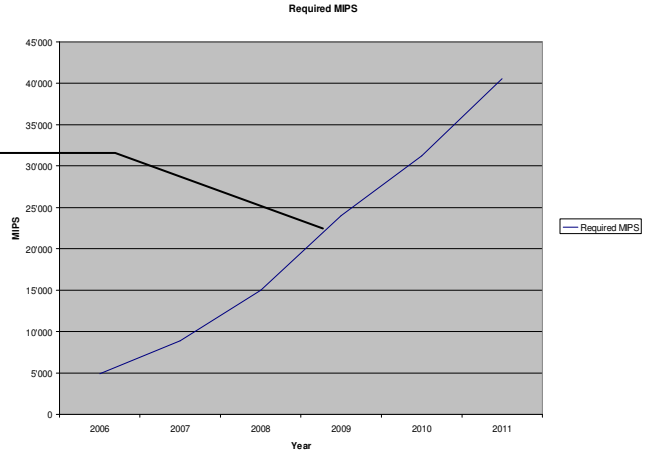
- Based on Cobol and DB2
- Managed by Tivoli Workload Scheduler (TWS)
- ~21'000 Cobol modules
- ~40'000 batch jobs a day
- Reuse of Cobol (batch) modules for online (CICS) processing, accessed by non-mainframe Websphere
- Minor workload has been moved to a home-grown "Java for Batch" environment

Swiss Re

Swiss Re's Business Motivation for Batch Modernization

- Due to several acquisitions, general growth in the reinsurance business and globalization of the application landscape, workload is expected to grow dramatically
- Budget has to remain ***flat***
- Over 80 000 MIPS expected in 2011

MIPS requirement per system



Swiss Re's IT Motivation/Requirements for Batch Modernization I

Swiss Re

- No big bang approach, co-existence of new solution with existing Cobol based solution
- Use of powerful, reliable, scalable z/OS environment/platform (scheduling, batch processing, DB2)
- Increase development speed and time to market
- Decrease of maintenance & development costs
- Manage risks & costs (shortage of COBOL skills)
- Modernize software development (including a "smooth migration path")
- Apply SOA principles to build an agile application infrastructure

Swiss Re's IT Motivation/Requirements for Batch Modernization II

Swiss Re

- Common business services should be shared across OLTP and batch applications
- Performance of compute grid better or equal to COBOL batch
- Same look and feel for operational staff

Swiss Re

Solution

- To keep the budget flat, we have to use zIIP and zAAP
- To use the new processor type efficiently, we have to change our application language from COBOL to JAVA
- Because most of the growing workload is batch, we also have to use the long running scheduler and execution environment from WAS/XD
- Smooth integration into z/OS infrastructure like TWS, Output etc…

Key Success Factor: Cooperation Model Between Swiss Re and IBM

Swiss Re

- True partnership between Swiss Re and IBM to implement next-generation batch processing runtime on z/OS
- Strong collaboration with the: WebSphere XD development team, IBM Software Services (ISSW), and Tivoli Workload Scheduler development team
- The close relationship – architects on-site, constant communication with the lab - was key to success as it allowed fast response times from both sides
- Important product designs were influenced by Swiss Re and subsequently delivered by IBM

Swiss Re

Next Steps I

- Leverage WebSphere XD Compute Grid's Parallel Job Manager for highly parallel batch jobs
- Performance features for pre-fetching data
- Code generation and tooling for application development
- Integrated batch monitoring infrastructure
- Begin incremental COBOL to Java migration

Swiss Re

Next Steps II

- Development tooling for COBOL – Java integration
- Technical features from IBM to facilitate COBOL – Java co-existence
  - Memory management model
  - Performance implications of cross-language calls
  - Transactional contexts and 2-phase-commit
- Tighter integration between Tivoli Workload Scheduler and WebSphere XD Compute Grid
- Selecting right database access technologies for different execution paradigms (OLTP, Batch):
  SQLJ, Hibernate, OpenJPA, Pure Query, etc
- Java and DB performance with EBCDIC, Unicode, ASCII
- Integrating transformation technologies