Why WAS z/OS

A systematic review of the bigger picture of
WebSphere Application Server z/OS
and the platform on which it operates

Agenda

- **"WAS is WAS"**
  A general review of WebSphere Application Server for all platforms, then a reminder of how WAS z/OS has portion of code designed to take advantage of z/OS platform
- **"Taking Advantage of the Platform"**
  A review of how WAS z/OS takes specific advantage of key elements of the platform
- **"Disaster Recovery and High Availability"**
  A review of DR and HA principles to reinforce how WAS z/OS participates in both
- **"The Cornerstone"**
  A review of System z, z/OS and Parallel Sysplex as the foundation for WAS z/OS
- **"Conclusion"**
  A brief wrap-up of the entire presentation

Preview

The "Why WAS z/OS?" question is ultimately answered with "Because of z/OS"

This section provides a review of key attributes of System z, z/OS and Parallel Sysplex

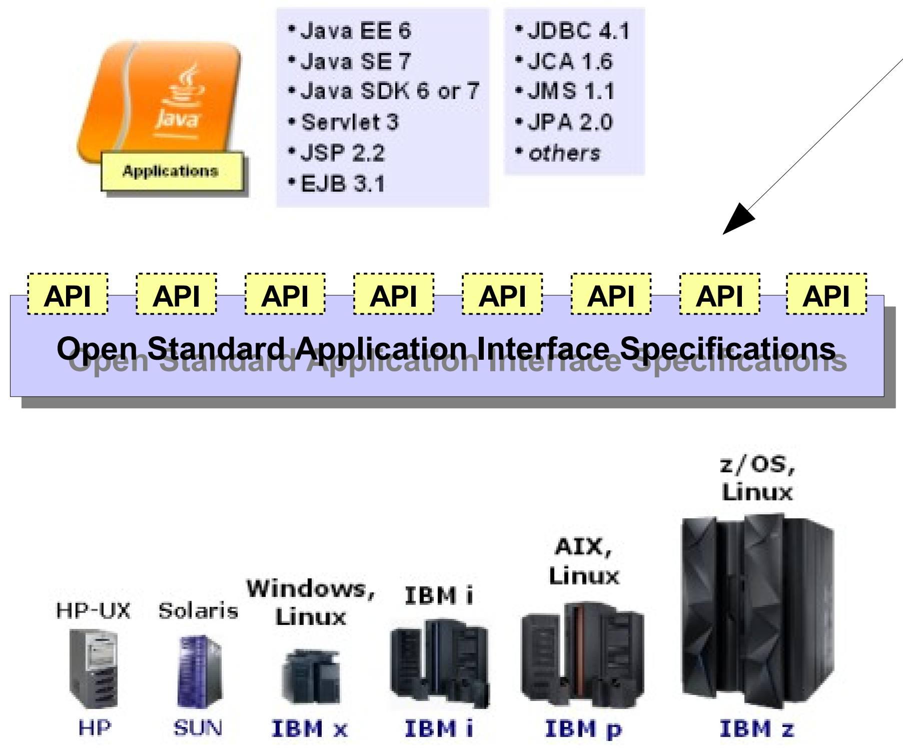
With that foundation established, we will then move into a discussion of WAS z/OS and illustrate how it takes advantage of the platform

Initial Overview

"WAS is WAS"

"WAS is WAS" Above the Standard Spec Line

A subtle but important point to make is that the standards supported by WAS are *common and consistent* across all the operating systems and hardware platforms:



•"WAS is WAS" at this level

That's of value because ...

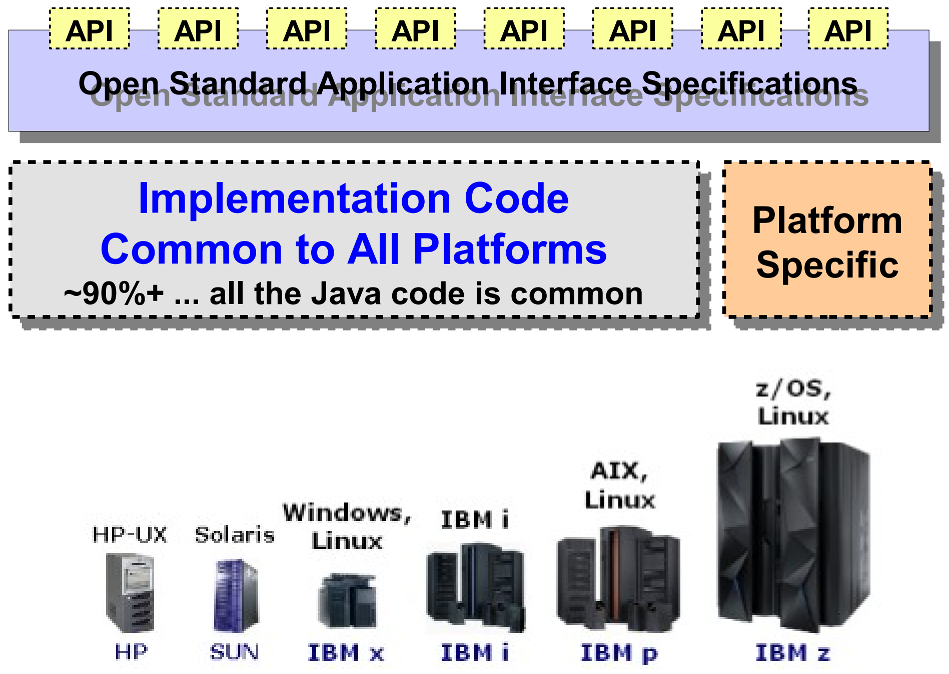
- ***Applications are platform neutral***
  Which means application design and coding does *not* require platform specific considerations
- **Allows you to consolidate on a common set of tooling**
  Which provides savings in licensing as well as skill development and skill utilization
- **Applications are portable across platforms**
  Which provides flexibility for code promotion from test and quality assurance on one platform up to production on another

It has been this way since Version 6.0

In addition to common spec support, new version and release schedules are aligned as are fixpack release schedules.

Implementation is *not* 100% Common

The code under the open specification APIs is not 100% common across all platforms. It is *mostly* common, but each platform has a *portion* of code specific to that OS:



- **Every platform OS has at least *some* code unique to that platform**
- **This becomes the focus when we get to section on WAS z/OS**
- **The z/OS operating system a long list of features and functions to take advantage of**
- **WAS z/OS is designed to know about and take advantage of many of those functions**

That's the core of the story -- the value of the System z and z/OS platform and how WAS z/OS leverages those features and functions to the benefit of the applications

WebSphere Application Server z/OS

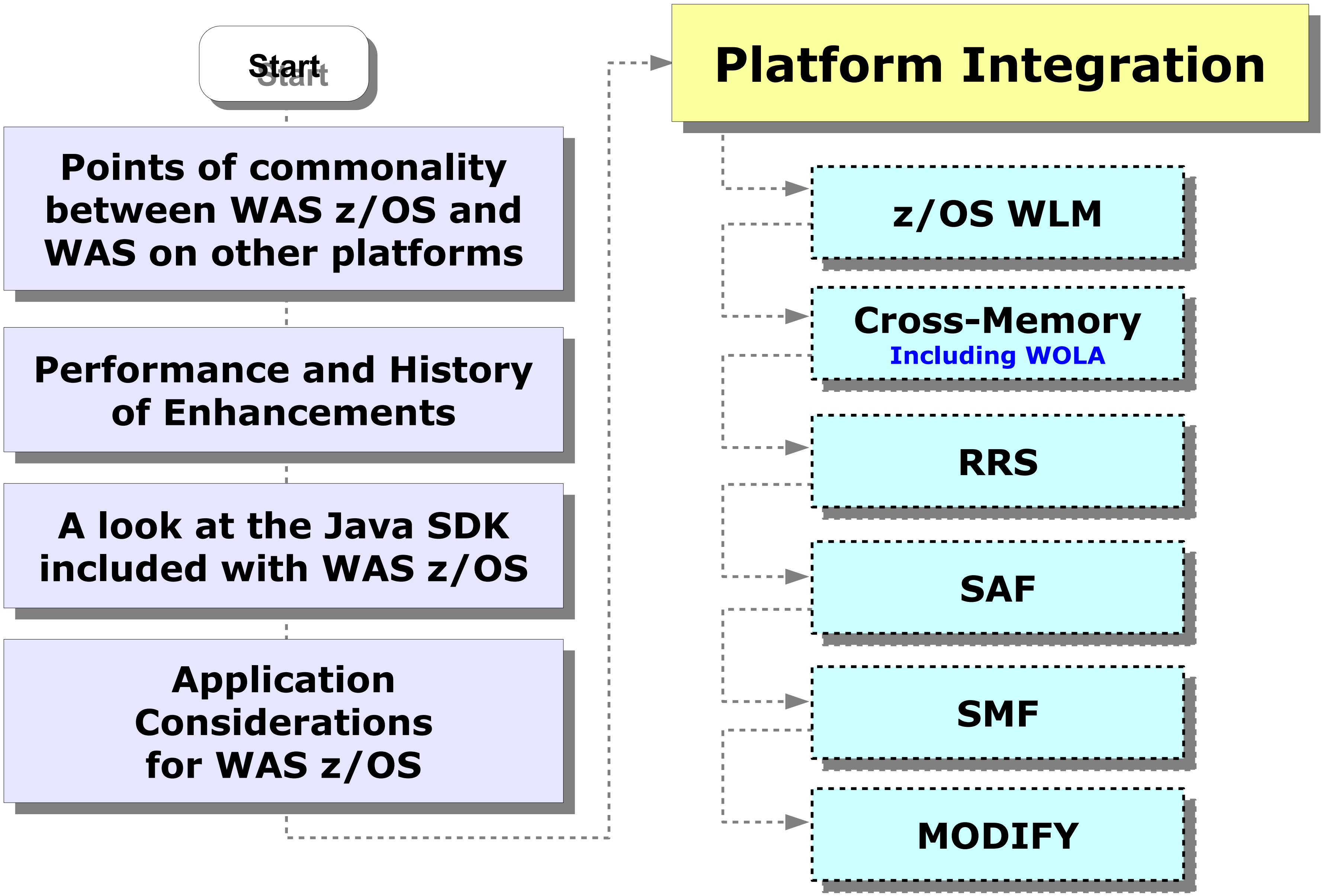
Taking Advantage of the Platform

Preview

The next step is to review how WAS z/OS takes advantage of these platform functions with an eye on the technical and business value that accrues

Outline of This Section

To give you a sense for what we'll discuss in this section:



Start
Points of commonality between WAS z/OS and WAS on other platforms
Performance and History of Enhancements
A look at the Java SDK included with WAS z/OS
Application Considerations for WAS z/OS
Platform Integration
z/OS WLM
Cross-Memory
Including WOLA
RRS
SAF
SMF
MODIFY

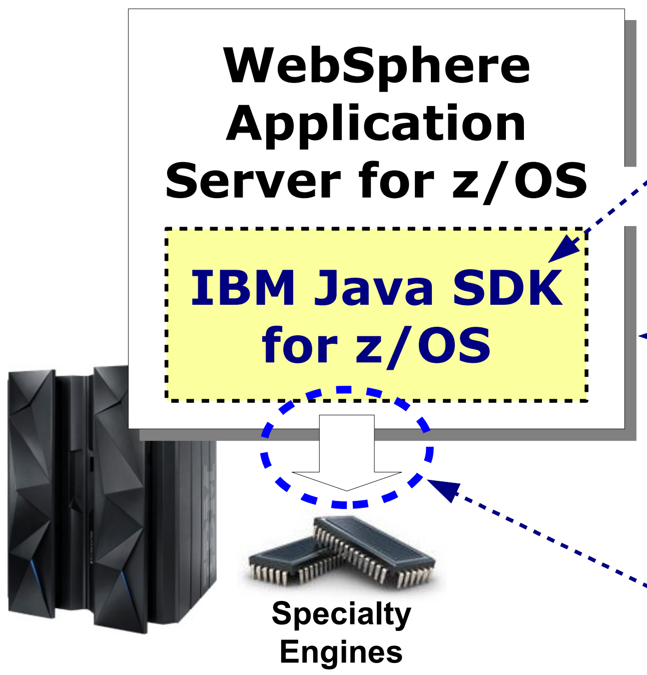
Points of Commonality / Points of Departure

WAS on z/OS has quite a few points of commonality with WAS on other platforms. The points of departure occur the closer you get to the platform:

Applications	Common and consistent open standard application interfaces across all platforms
Administration	
Admin Console	Administrative console nearly identical across platforms. Differences surface lower in areas related to platform specifics, such as START commands, z/OS security, z/OS WLM
WSADMIN	The WSADMIN command objects and methods are the same across all platforms. Differences surface in attributes related to platform specifics.
Application Deployment	Application deployment is the same across all platforms
Data Resources	Data resource definition *concepts* the same across platforms. Differences surface in areas related to cross-memory data sources and specification of native library locations
Security	Java EE security concepts the same across platforms. Differences surface in areas related to how definitions manifest in SAF
Logging / Tracing	Output on z/OS by default goes to JES spool, though starting with V8 the HPEL option makes output common across platforms
Operations	Starting and stopping servers from Admin Console the same across platforms. From OS command line z/OS specifics are used.
Construction	Construction of the runtime very unique to z/OS

IBM Java SDK for z/OS included with WAS for z/OS

WAS for z/OS is packaged to include the IBM Java SDK for z/OS. Install and configure WAS z/OS and the Java SDK is present:



- **The Java SDK binaries are packaged with the WAS z/OS binaries, and maintenance of Java is part of WAS z/OS maintenance**
  IBM coordinates Java SDK maintenance prerequisites for each fixpack of WAS z/OS
- **31-bit or 64-bit configurable by server**
- **With WAS V8.5 Java 6 or Java 7 is included and configurable by server**
  Default is Java 6 -- maintain same Java spec as previous versions of WAS if desired
  Java 7 is latest and includes specific exploitation of instructions in new IBM System z EC12 machine
- **Offload to specialty engines falls between 50% and 80% of WAS z/OS CPU**
  Offload *completely transparent* to the Java app in WAS
  The degree of offload is dependent on many factors. An application that performance extensive Java work (XML parsing, for example) will offload relatively more. Applications that make greater use of WAS z/OS native platform services relatively less.

WAS z/OS is a Java application runtime environment

WAS z/OS supplies the Java SDK

Application Considerations for WAS z/OS

"WAS is WAS" at the application specification level. The key consideration is in the area of performance budgeting and testing:

Three important points:

Applications for WAS z/OS are *no different* than applications for WAS in general

WAS applications are written based on open and accepted industry standards

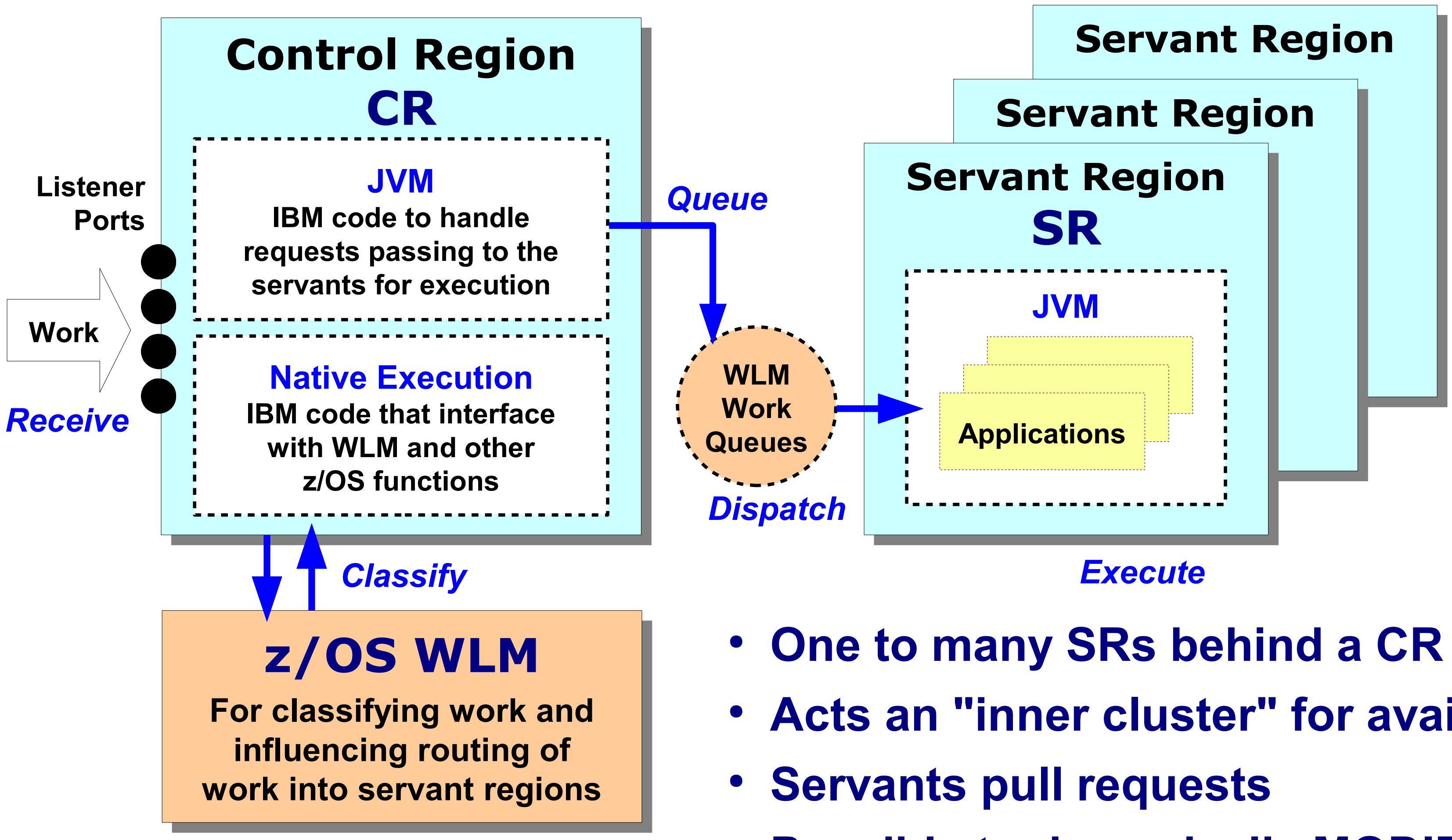
Any tooling capable of producing applications for Java SE or Java EE will work with WAS z/OS

That said, there are things to consider:

- Always follow good coding practices ... avoid hard-coded references to platform specific information
- Avoid server-specific singletons, which may create issues in clustered servers, including multiple WAS z/OS servants
- Be aware WAS z/OS is typically a very high volume environment ... inefficiencies in the code are amplified when exposed to very high levels of request activity
- If writing an application intended for high volume activity, have a performance goal in mind. Write smart, efficient code for heavily used functions. Cache where possible. Profile the application to determine possible inefficiencies. Load test before going live in production.

WAS z/OS Multi-JVM Application Server Model

WAS z/OS has a unique application server model -- it separates request handling from application serving. The result is a "pull" model with strong integration with WLM:

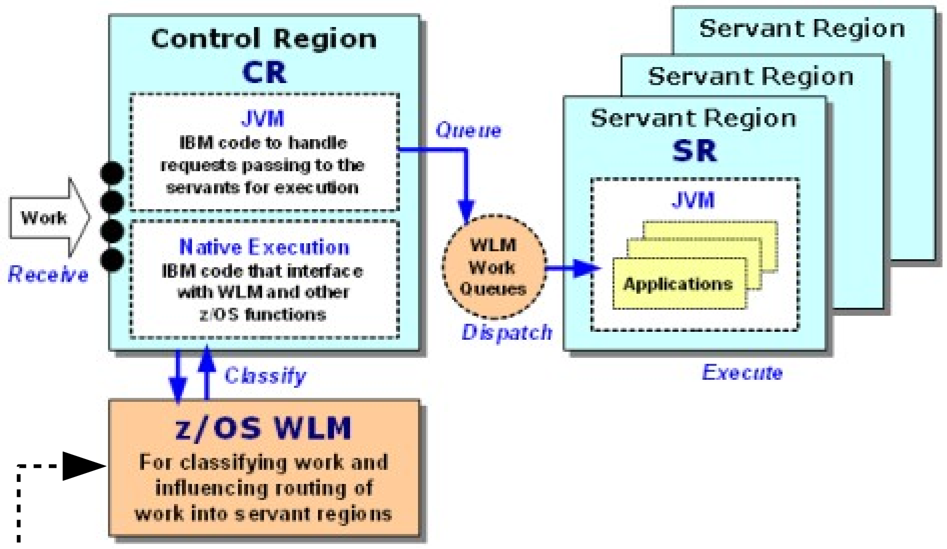

Listener Ports
Work
Receive
Control Region
CR
JVM
IBM code to handle requests passing to the servants for execution
Native Execution
IBM code that interface with WLM and other z/OS functions
Queue
WLM Work Queues
Dispatch
Servant Region
Servant Region
Servant Region
SR
JVM
Applications
Execute
Classify
z/OS WLM
For classifying work and influencing routing of work into servant regions

- One to many SRs behind a CR
- Acts an "inner cluster" for availability
- Servants pull requests
- Possible to dynamically MODIFY # of SRs
- WLM will automatically restart failed SRs

More detail: WP101740 at ibm.com/support/techdocs

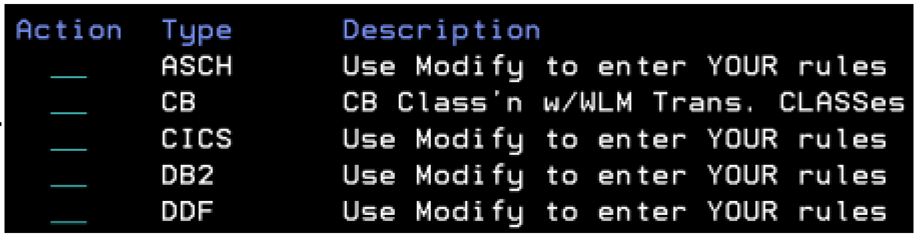
Platform Integration: WLM Classification

All work is classified ... with option to be more granular with your classification



Control Region
CR
JVM
IBM code to handle requests passing to the servants for execution
Native Execution
IBM code that interface with WLM and other z/OS functions
Work
Receive
Queue
WLM Work Queues
Dispatch
Servant Region
Servant Region
Servant Region
SR
JVM
Applications
Execute
Classify
z/OS WLM
For classifying work and influencing routing of work into servant regions

z/OS WLM Classification Rules for Subsystem Type = "CB"



```
Action   Type        Description
 __      ASCH        Use Modify to enter YOUR rules
 __      CB          CB Class'n w/WLM Trans. CLASSes
 __      CICS        Use Modify to enter YOUR rules
 __      DB2         Use Modify to enter YOUR rules
 __      DDF         Use Modify to enter YOUR rules
```

Optional XML file for classifying individual requests to WLM Transaction Classes

- **All work is classified ... in absence of any specific coding the CB rules defaults will apply**
- **Classifying by CN= qualifier gives you opportunity to have different classification *by application server***
  Different service class goals by appserver or different reporting classes for RMF reporting
- **Use of XML file allows you to assign WLM TC to *individual requests*, which provides:**
  - **Different WLM reporting classes for applications in same server**
  - **Different WLM service classes for applications, which then get placed to own servant region**
  - **Use of new "Granular RAS" function explained next**

Platform Integration: Granular RAS Function

New in V8 and exclusive to WAS z/OS, this allows you to identify individual requests and drive WAS behavior *down to the request level* ... not just appserver level:

```
<Classification schema_versio
   <InboundClassification typ
      default_transaction_cla
      <http_classification_info
            uri="/ApplicationABC/*" transaction_class="TRANCL"
            description="ABC" dispatch_timeout="300" />
      <http_classification_info
            uri="/XYZApplication/*" transaction_class="TRANCL"
            description="XYZ" dispatch_timeout="15" />
   </InboundClassification>
</Classification>
```

Two applications, *same server,* different timeout values associated with each based on URI

Functions:

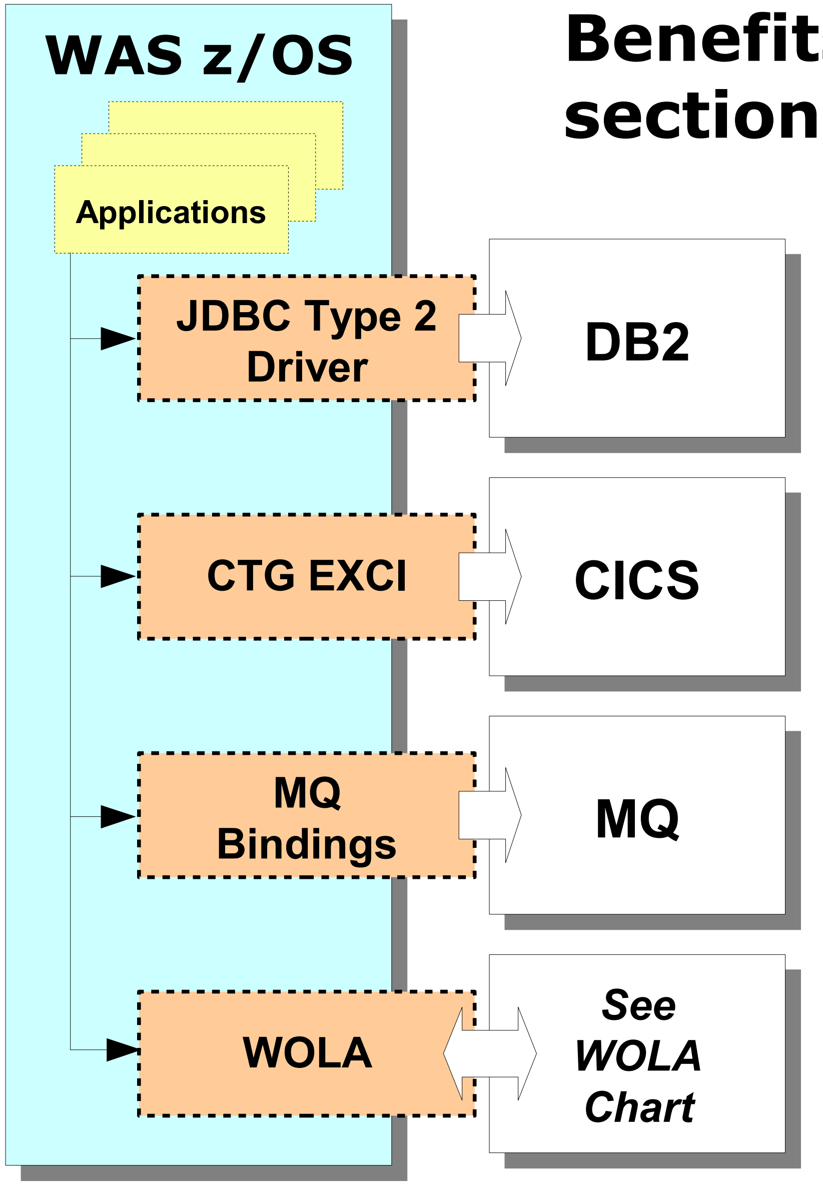
- Various Timeouts
- Stalled Thread Dump Actions
- CPU Time Used Limit
- Dispatch Progress Monitor (DPM) Interval and Dump Action
- SMF Recording
- Tracing
- Message Tagging
- Timeout Recovery Actions

Allows you to consolidation applications in a server and apply differential behavior to each

Allows higher degrees of utilization of existing appservers

Platform Integration: Cross-Memory

WAS z/OS is capable of taking advantage of native interaces to key data systems to exchange communications over a cross-memory boundary:

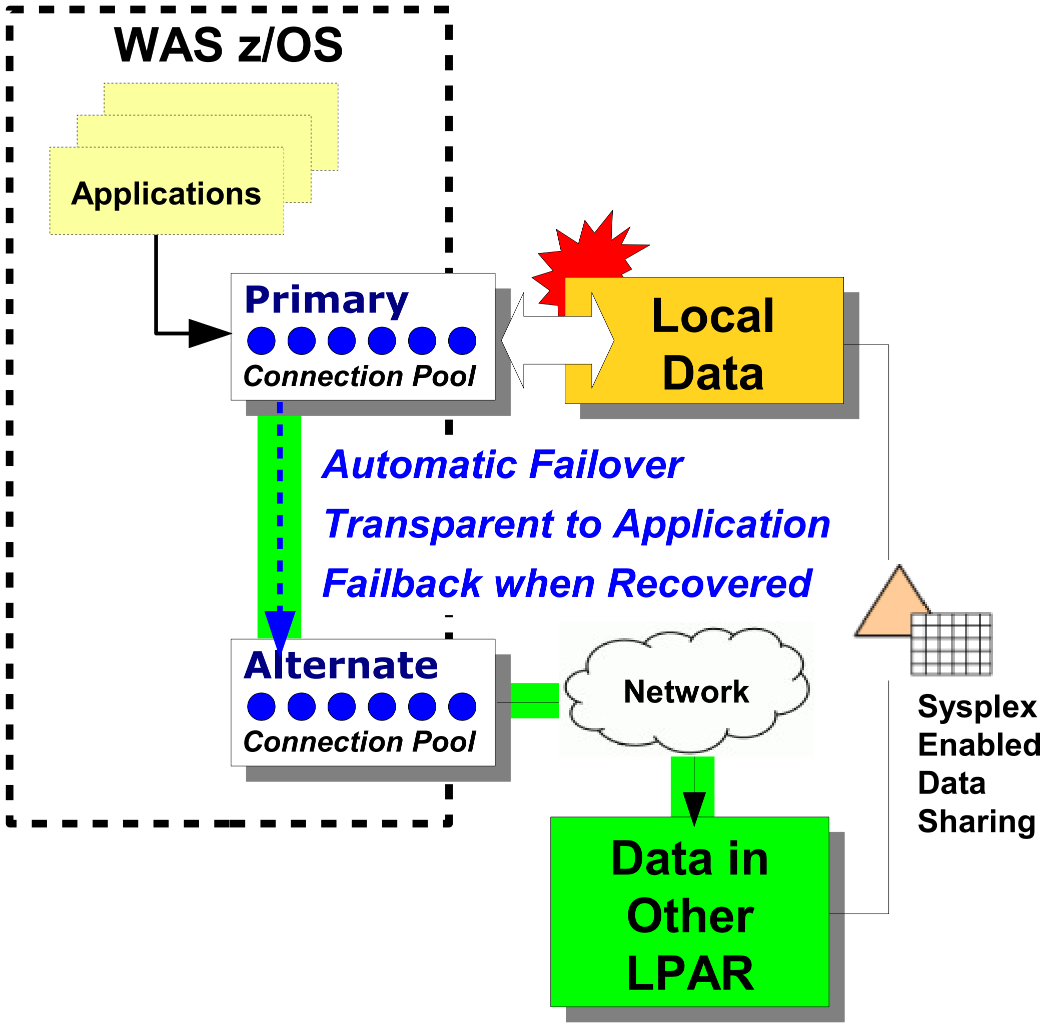


Benefits mentioned earlier under section on value of System z and z/OS:

- **Very low latency**
  The problem of latency tends to be additive in high volume, repetitive transactions
- **Very secure**
  Data can not be sniffed, intercepted or modified
- **Avoid encryption / decryption overhead**
  Since exchange is so secure, the need to encrypt may be eliminated
- **Security identity assertion across interface**
  Avoid coding identity "aliases" in different locations across enterprise
- **Avoid TCP/IP stack processing overhead**
  Reduces overall system CPU usage
- **Single thread of execution across interface**
  Avoid task switching overhead
- **Reduced complexity for debug and troubleshooting**
  When sender and receiver are in same OS environment one set of tools may be used.

New in V8* - Alternate JNDI Failover / Failback

Provides a way to define a secondary data source in the event the local data source is lost. Trigger is `getConnection()` failures. Fails back when local returns:

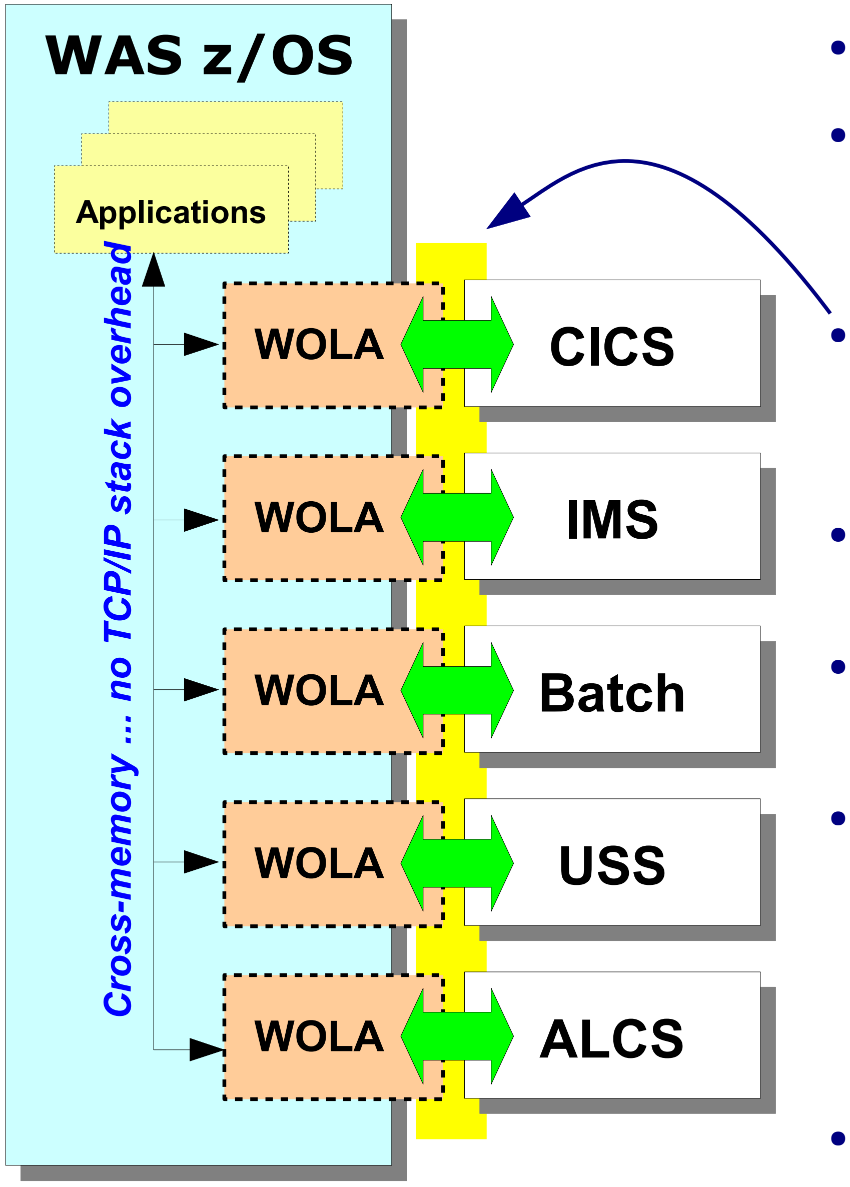


- **Applications bind to primary data source JNDI**
  They do not know about alternate JNDI definition -- transparent to them
- **If `getConnection()` failure on primary, then auto-failover to alternate data source**
  Number of failures before failover configurable
- **When primary data source comes back, then failback**
  Polling interval configurable
- **WAS z/OS extends with MODIFY to manually failover / failback**
  Important for planned outages of data systems
- **WAS z/OS and Parallel Sysplex with data sharing makes these a natural part of HA story**

* Basic function available in WAS on all platforms

Platform Integration: WOLA

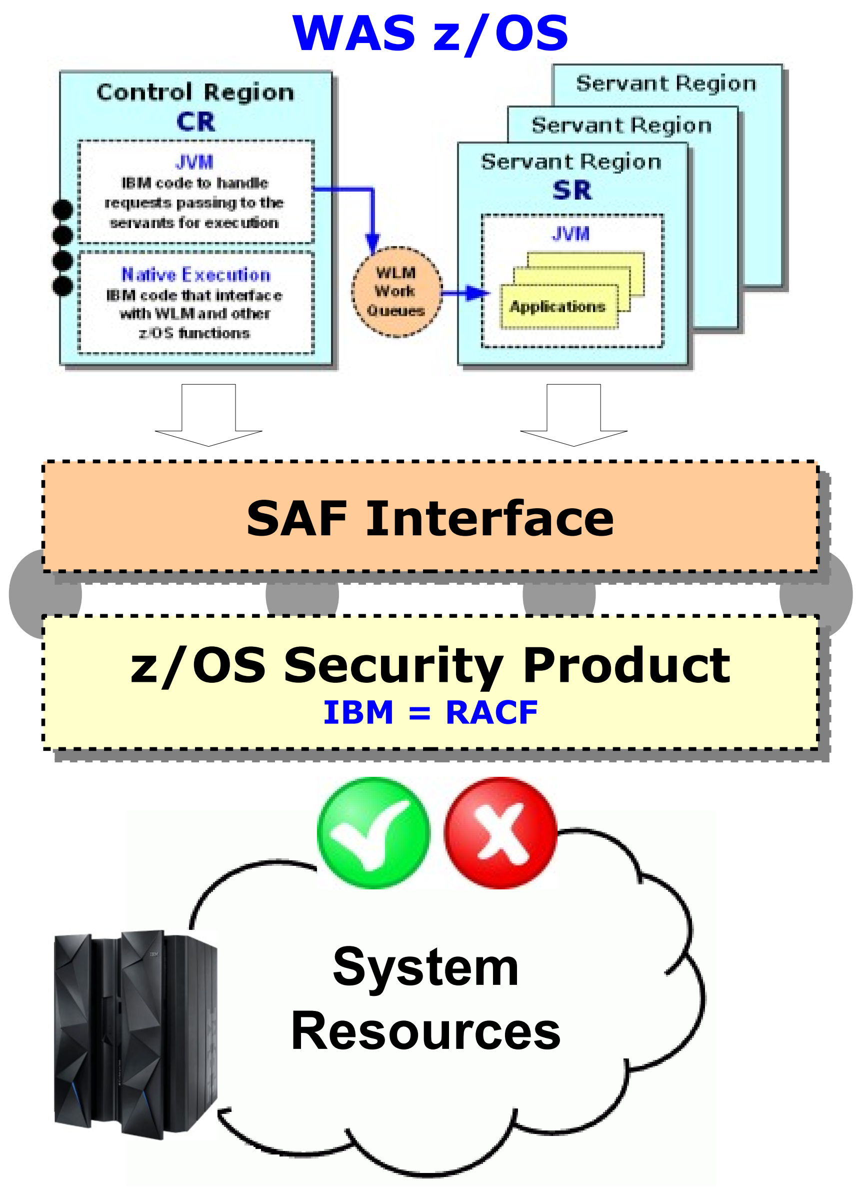
WebSphere Optimized Local Adapters provides an efficient low-latency mechanism to exchange data bi-directionally between WAS z/OS and other address spaces:



- **Very efficient byte-array transfer**
- **First available with WAS z/OS 7.0.0.4**
  - Enhanced several times since then
  - See WP101490 "History of Updates to WOLA" PDF
- **Bi-directional**
  - Outbound -- Java in WAS invokes program in external
  - Inbound -- Program in external invokes Java in WAS
- **Two phase commit, identity assertion**
  - In some cases; see InfoCenter or WP101490 Techdoc
- **Supplied JCA resource adapter for applications going outbound**
- **Supplied native APIs for cases where their usage is indicated**
  - COBOL, C/C++, PL/I, High Level Assembler
  - 31-bit and 64-bit modules
  - See WP101490 for "Primer" on API usage
- **WP101490 Techdoc for much more**

Platform Integration: SAF

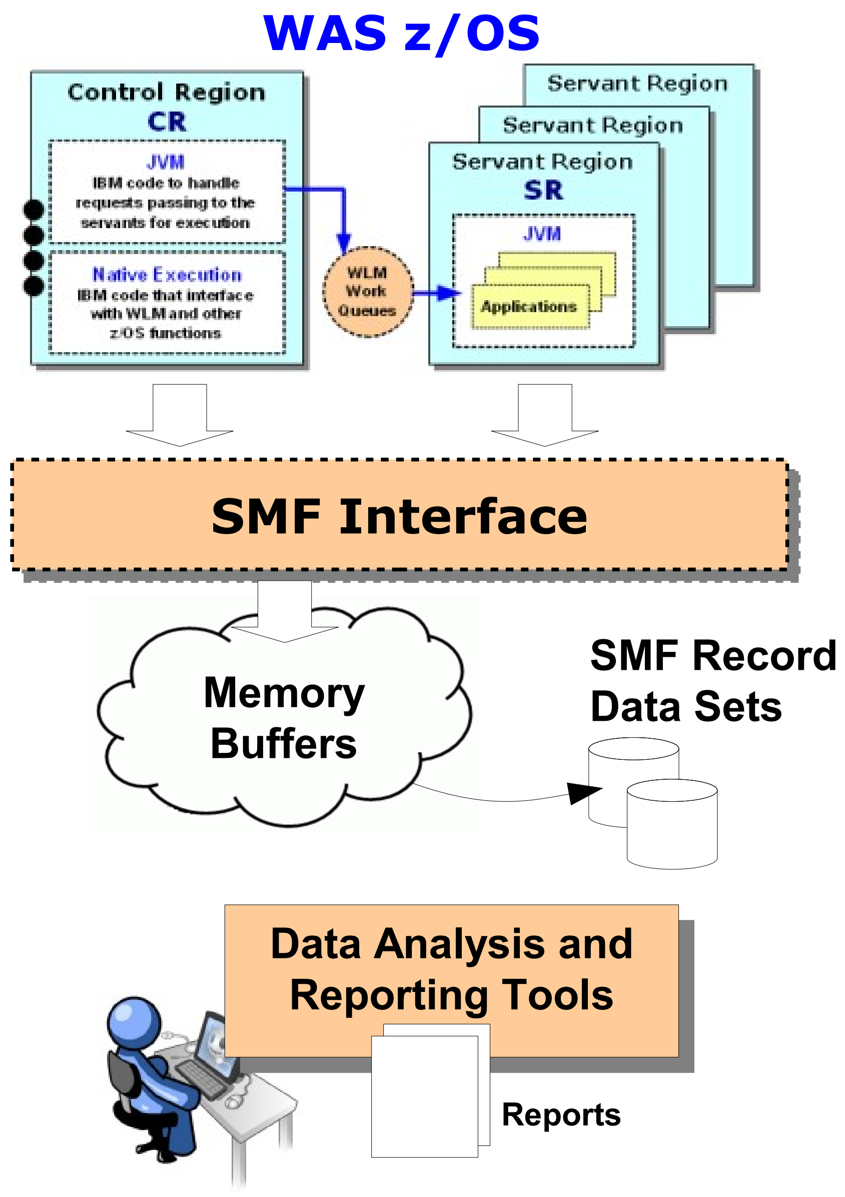
SAF -- and the security product behind it -- provides a security management tool for WAS z/OS ... infrastructure security, application security, user identity security:



- **SAF may be used to store user ID and password information for authentication**
  LDAP may also be used, and a WAS cell may have separate security domains such that SAF is used for part and LDAP for another
- **Digital certificates may be stored in SAF, eliminating separate keystore files**
- **Application EJB roles may be enforced by SAF**
- **SAF controls what address space ID is assigned to WAS z/OS started tasks**
- **SAF may be used to reserve TCP ports for WAS by WAS server name**
- **SAF provides a single point of security administration with a strong tradition of careful, controlled security management**

Platform Integration: SMF

SMF is a facility to capture information about z/OS and subsystem activity. WAS z/OS writes the SMF 120.9 and SMF 120.10 records with a rich set of request information:

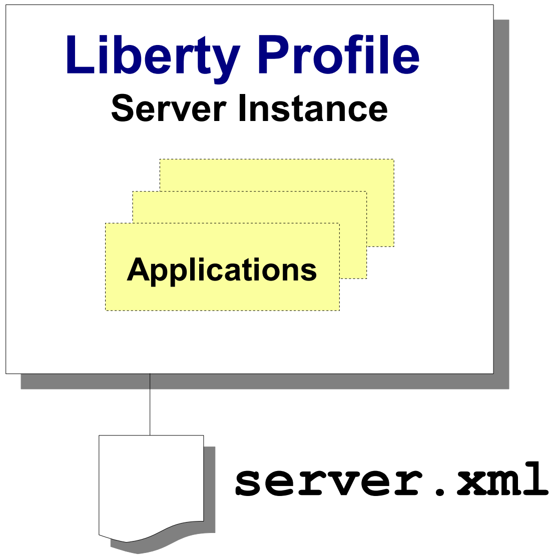


- **WAS z/OS: 120.9 and 120.10**
  - **120.9 -- detailed *inbound* request data**
  - **120.10 -- detailed *outbound* request over WOLA**
- **120.9 ... a record for each request:**
  - **System name, Sysplex Name, Jobname, ASID**
  - **WAS cell and server and servant information**
  - **Timestamps: arrive, queue, dispatch, complete**
  - **CPU time spent: overall and specialty engines**
  - **Network data: bytes sent / received, host and port**
  - **Request URI for HTTP**
  - **Security information**
  - **Much more -- see WP101342 Techdoc for more**
- **120.10 ... a record for each outbound:**
  - **Available 8.0.0.1 and later**
  - **Similar to SMF 120.9**
  - **Transaction ID**
  - **Security information**
  - **Outbound to CICS and outbound to IMS over OTMA info**

Excellent source of data for usage, trend and capacity analysis

Liberty Profile and z/OS

The Liberty Profile is a lightweight, dynamic, composable server model that became available with WAS V8.5. On z/OS it is extended to take advantage of the platform:



z/OS Extensions:

- SAF
- WLM
- RRS Transaction
- MODIFY

- **A single JVM server model:**
  - **Lightweight** - relatively small footprint
  - **Composable** - configure only function needed
  - **Dynamic** - configuration changes, application deployments
  - **Fast** - server starts in as little as a few seconds
- **Simple configuration and operations**
- **On z/OS start as UNIX process or started task**
- **z/OS Extensions:**
  - SAF - use SAF as repository for authentication data and SSL key and trust store
  - WLM - classify work into separate service or reporting classes
  - RRS TX - use JDBC Type 2 with RRS TX support
  - MODIFY - use z/OS MODIFY to process dump actions

A very good development and test complement to traditional WAS z/OS

WebSphere Application Server z/OS

Disaster Recovery and High Availability

Preview

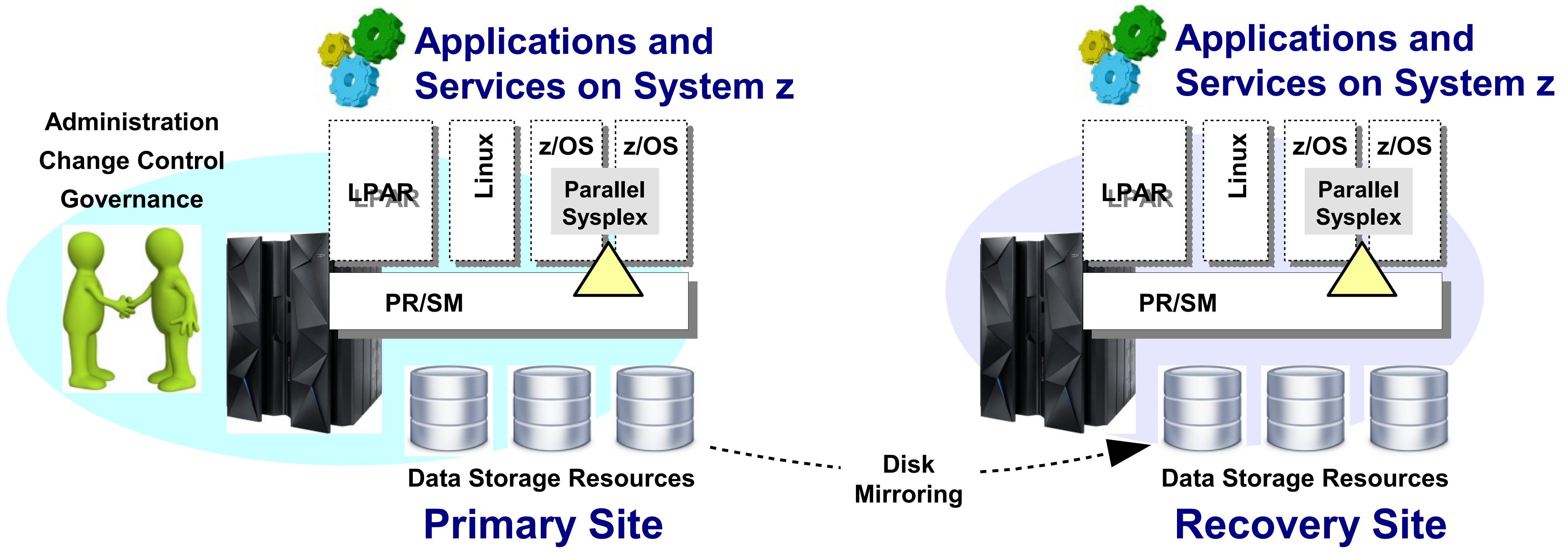
System z and z/OS are well known for being a highly available platform for business solutions

The ability to recovery from a site outage is also part of the story

In this section we will take a tour of both DR and HA and illustrate how WAS z/OS participates in both

Disaster Recovery

Because System z and z/OS are part of a consolidated, controlled server platform, disaster recovery planning and execution tends to be much more successful:



Disaster Recovery is more successful when all the components of a system are properly accounted for and implemented at the DR site

When all the components are part of a cohesive environment like System z, the probability of DR success is much higher

WAS z/OS and Disaster Recovery

All the components and artifacts of a WAS z/OS configuration are common elements to z/OS, and because of that are a natural fit within DR practices:

Configuration file systems

- Consists of a directory tree and XML files
- Contain cell, node and server information
- Contain all customization properties
- UNIX file system manifests as z/OS data set

Deployed application binaries

- Contained within configuration file systems

JCL start procedures

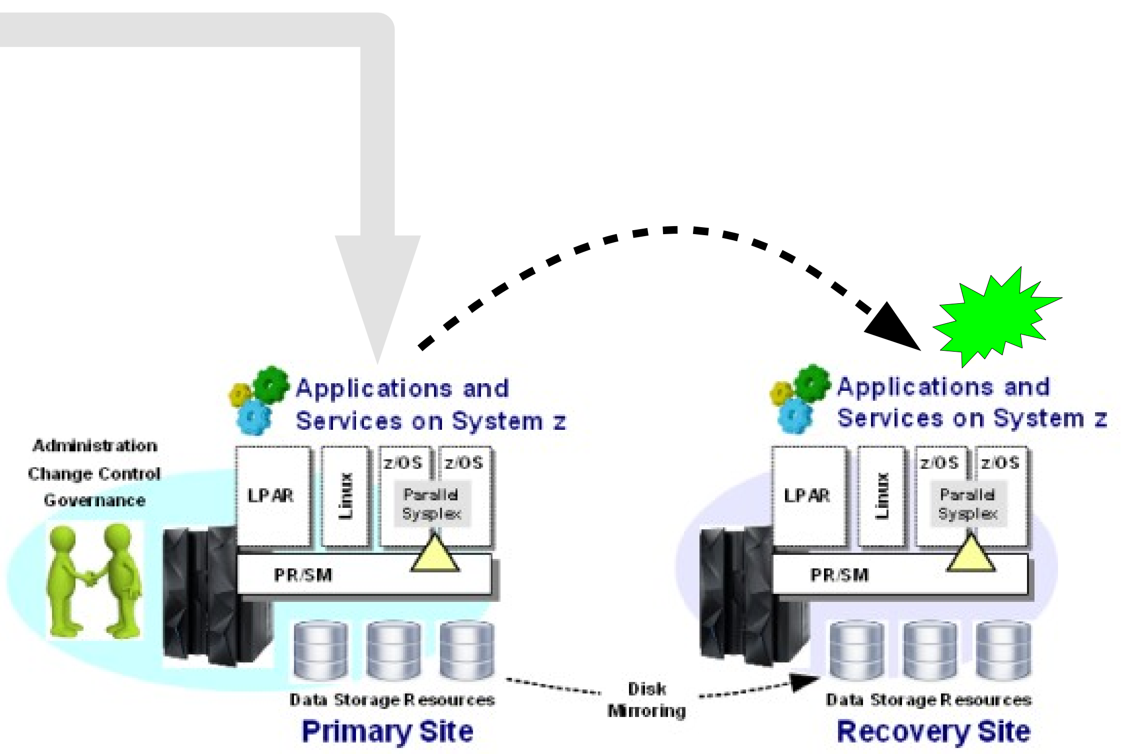
- Used to initialize the started tasks that are the servers

SAF security profiles

- Used to protect the z/OS runtime components
- May be used for user authentication (or use LDAP)

WLM classification rules

- Determines relative priorities of WAS z/OS tasks
- Optionally may be used to classify work requests

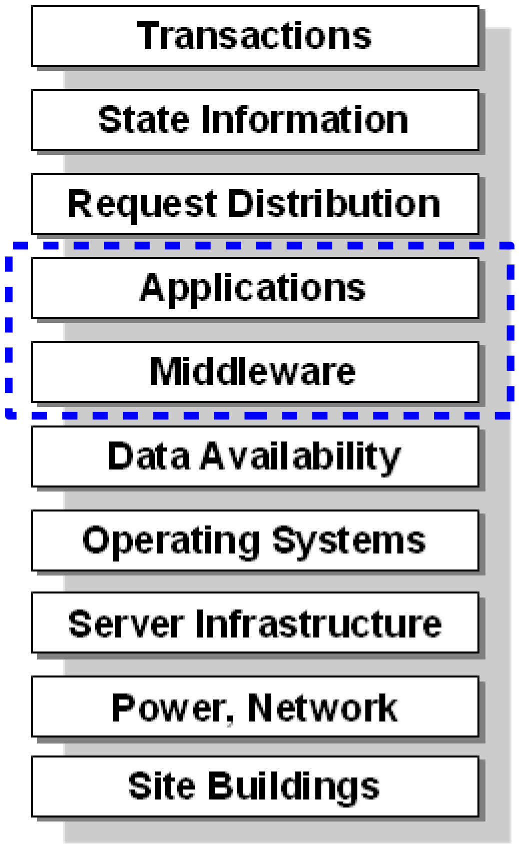


WAS z/OS manifests as standard z/OS artifacts ... as such they easily fit within any current DR practices that exist

If DR practices for z/VM well-formed, then WAS on Linux for System z would be similar

Duplicated Application Servers (WAS Clusters)

Two forms: duplicated CR/SR appservers as well as "inner clusters" of multiple SRs

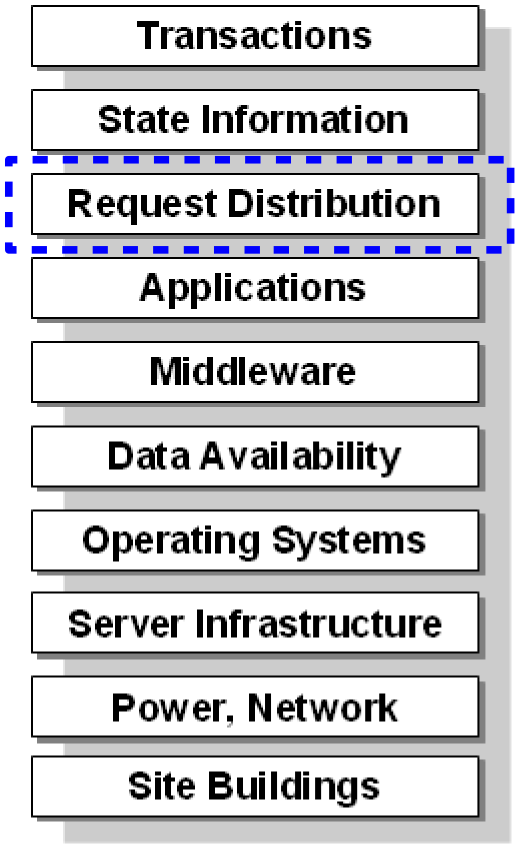


- WAS z/OS may be clustered across Parallel Sysplex members, with access to available shared data
- Clustered WAS z/OS application servers provide redundant controller regions (CR) with listener ports on separate OS instances on top of shared data
- Multiple servant regions (SR) in each appserver provide further redundancy within each OS
- WAS z/OS clusters provide internal replication services for state data and failover mechanisms for transaction recovery

Duplicated appserver and applications protects against outage of either of those elements in the stack

Client Request Distribution Across Cluster Members

Duplicated servers implies some device out front to distribute requests to cluster:

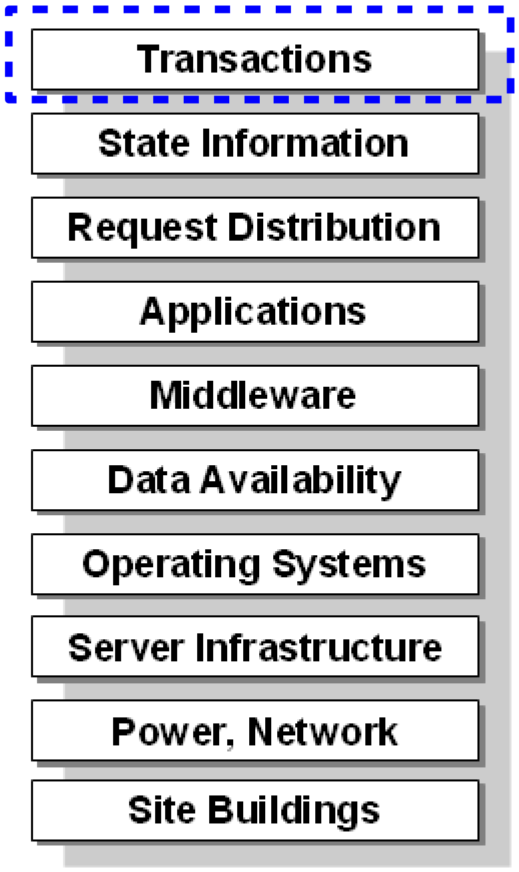


- Clustered WAS z/OS application servers present request listener ports on separate OS instances
- External request distribution mechanisms work well with WAS z/OS clustered servers
- WAS z/OS PAUSELISTENERS / RESUMELISTENERS provides dynamic means of turning off ports but leaving server up and working ... front-end distribution devices will not distribute to inactive TCP ports
- z/OS TCP provides Virtual IP address support and Sysplex Distributor for work distribution within Sysplex based on WLM routing advice

Key aspect of HA design is that client calls back if connection broken ... so having re-routing function to handle call-back is important

Transaction Recovery

Transactions left "in-doubt" may lock data, which reduces overall availability

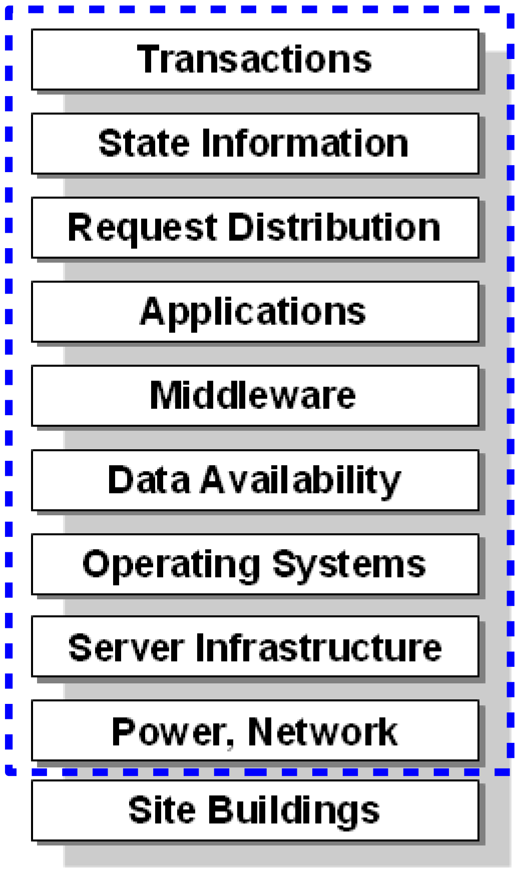


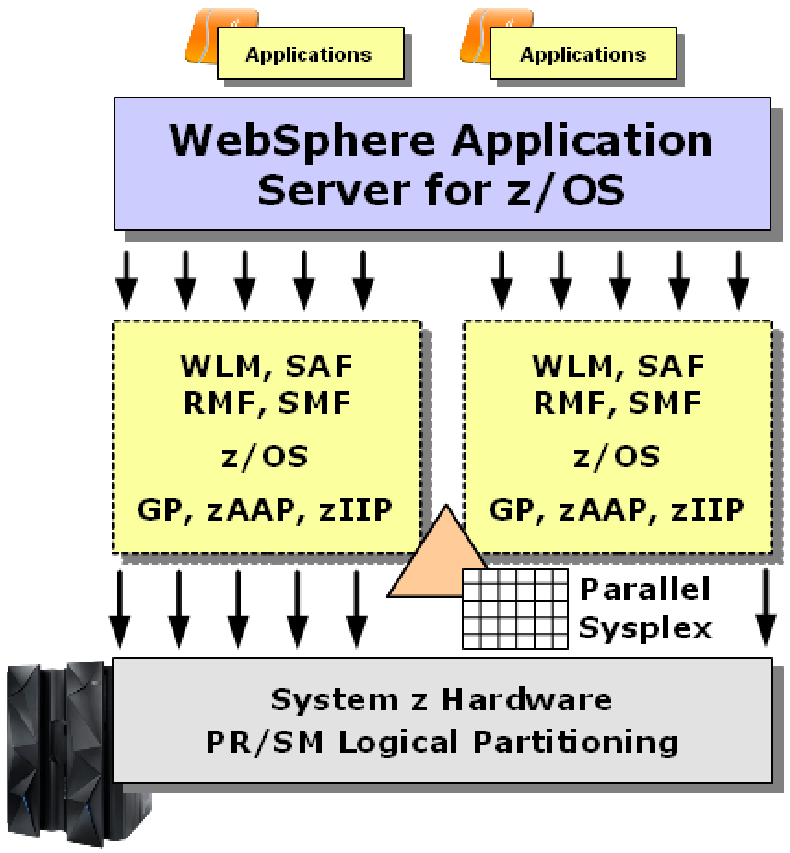
- WAS z/OS makes use of RRS, which is a transaction synchpoint coordinator that is Sysplex-enabled
- WAS z/OS as either TX manager or TX participant registers with RRS, as do key z/OS data systems such as DB2, CICS, IMS and MQ
- RRS maintains transaction logs in shared data structures within Parallel Sysplex Coupling Facility
- WAS z/OS clusters provide TX manager failover to surviving cluster member so in-doubt TX may be recovered. Since RRS is under cluster and is Sysplex aware, it has access to TX logs to perform recovery
- Data locks held by in-doubt TX are freed ... making data once again available

Transaction recovery is not exclusive to z/OS, but presence of RRS and Parallel Sysplex greatly enhances the TX recovery scenario picture

Overall HA Story with WAS z/OS and Parallel Sysplex

Over the last several charts we painted a picture that covered bottom to top:




Applications
Applications
WebSphere Application Server for z/OS
WLM, SAF RMF, SMF z/OS GP, zAAP, zIIP
WLM, SAF RMF, SMF z/OS GP, zAAP, zIIP
Parallel Sysplex
System z Hardware PR/SM Logical Partitioning

The graphic on the right is the picture we used to close the overview section on System z, z/OS and Parallel Sysplex

Essential message is one of a highly available overall system on which WAS z/OS operates

System z, z/OS and Parallel Sysplex

The Cornerstone

IBM System z Hardware

There's a long history of resilient design and very high data throughput:

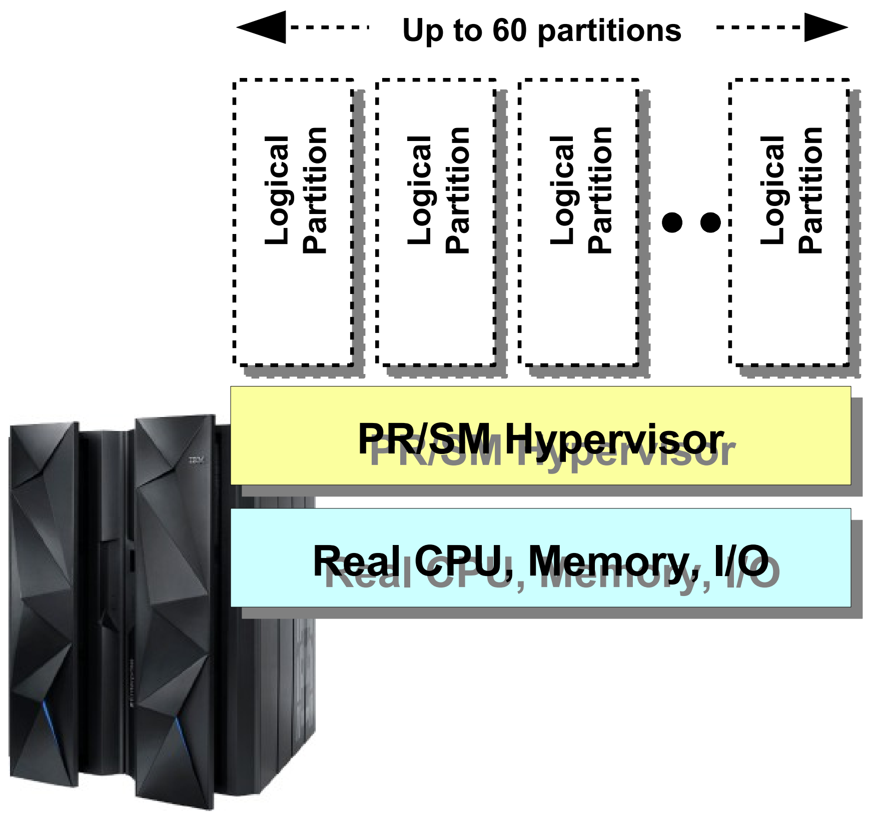
The IBM zEnterprise EC12

- **Many points of redundancy in design**
- **Engineered to automatically survive many component failures**
- **Support for non-disruptive configuration changes and dynamic replacement capabilities**
- **EC12 has 120 cores running at 5.5GHz**
- **Dynamic capacity on demand**
- **Integrated data compression and data encryption on the processor chip**
- **Continued refinements in execution processing, improved prefetch instructions, 2GB page sizes, Flash Express (internal solid state disk drive)**
- **Many other features ...**

This is a the starting point -- a server hardware base that is designed to be resilient, available, and perform at very high levels of throughput

Hardware Virtualization - PR/SM

The System z hardware allows for logical partitioning using PR/SM hypervisor:



- Allows you to partition physical hardware into up to 60 logical machines
- Each partition may be totally isolated from others (with EAL5+ rating) or communicate using network protocols
- Operating systems that may run in LPAR: z/OS, z/VM, Linux for System z, TPF, VSE
- LPARs may be dynamically added or deleted without impacting other LPARs
- Processor allocations between LPARs may be changed dynamically
- LPARs may be grouped into a Parallel Sysplex (more on this coming up)

PR/SM provides flexible partitioning of real server assets into a number of different OS environments if needed

System z "Specialty Engines"

System z processors that offload certain processing from the general processors. They have a lower acquisition cost and do not count towards processor-based licensing

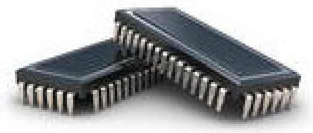
System z Application Assist Processor (zAAP)

For offload of Java work running on z/OS. Dispatching to the zAAP is transparent to the application.

The new EC12 machine is the last to have zAAP processors. After that, offload of Java work will be done using "zAAP on zIIP" (see below)

IBM System z Integrated Information Processor (zIIP)

For offload of certain work such as DB2 DDF processing, some DB2 utilities, and other select processing



"zAAP on zIIP"

A means of having one type of defined speciality engine (zIIP) and offloading both Java (zAAP) and zIIP-eligible work to pool of zIIPs

Integrated Facility for Linux (IFL)

A specialty engine for running Linux for System z or z/VM in an LPAR

IBM z/OS Operating System

A multi-user, multi-program OS with a long history of reliability and high-throughput operations.

IBM z/OS
Version 1 Release 13
is the latest

- A very mature, stable and reliable OS
- A design philosophy and architecture* for RAS:

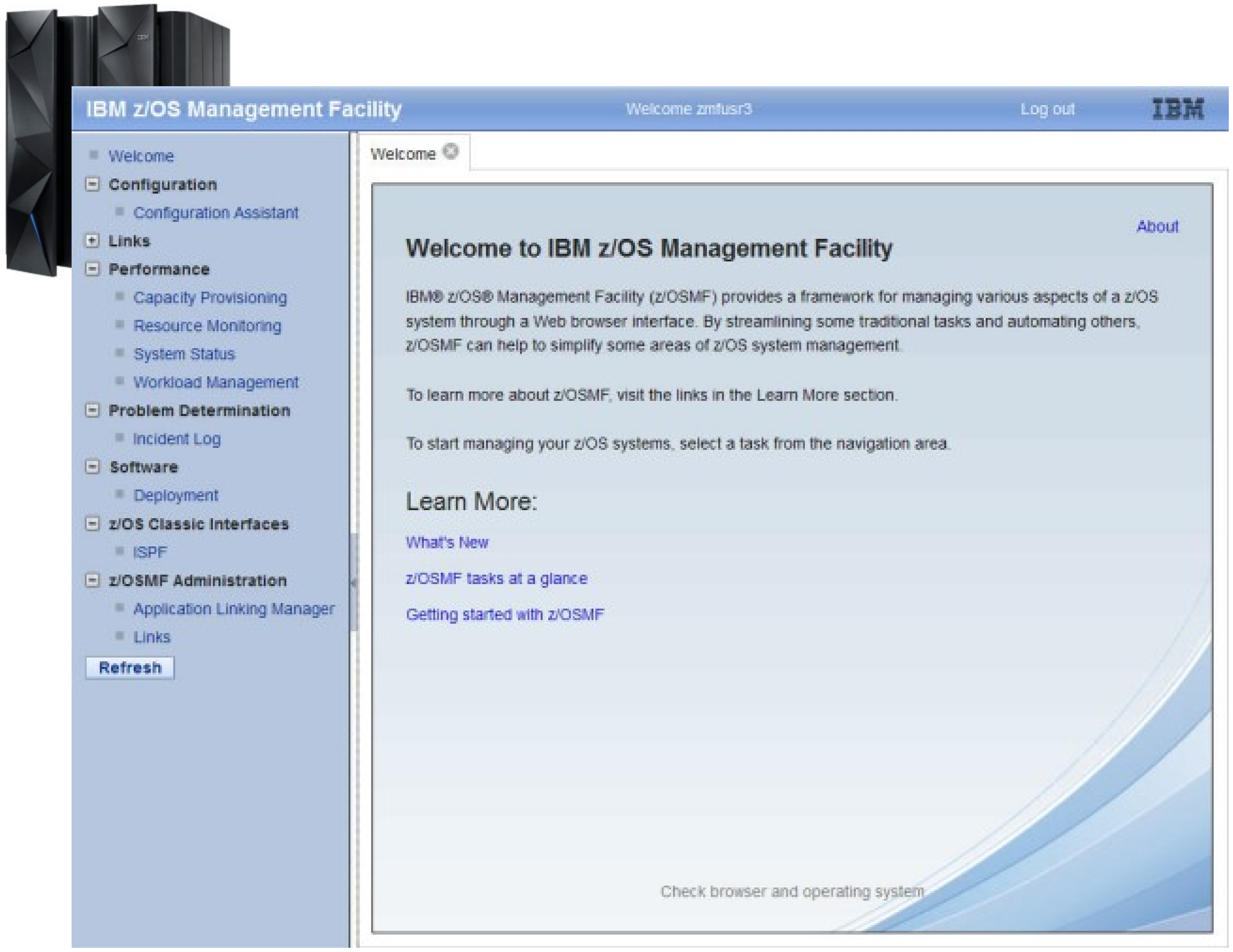
  Assume the work is mission critical
  Allow no undetected errors
  Isolate all failures to the smallest affected unit of work
  Provide diagnostics from the first failure enough to debug a problem
  Allow no program access to data it is not authorized to access
  All system (and subsystem) code is "covered" by a recovery routine.
  Critical code has "nested recovery" to cover the recovery routines.
  Diagnostic data specific to the error is gathered and reported.
  Retry is attempted whenever possible after repairing damage and isolating the failure.

- Strong integration with the underlying hardware
- EAL4+ security certification
- An rich set of subsystems such as WLM, RMF, SMF, SAF ... some of which we'll cover in more detail later
- An extensive catalog of mature systems management and monitoring tools

* From material authored by Bob Rogers, recently retired IBMer, who was very involved with the early design and evolution of IBM over the last 40+ years

z/OS Management Facility

z/OS MF provides a graphical interface to the traditional green-screen of z/OS:



Welcome to IBM z/OS Management Facility

IBM® z/OS® Management Facility (z/OSMF) provides a framework for managing various aspects of a z/OS system through a Web browser interface. By streamlining some traditional tasks and automating others, z/OSMF can help to simplify some areas of z/OS system management.

To learn more about z/OSMF, visit the links in the Learn More section.

To start managing your z/OS systems, select a task from the navigation area.

Learn More:

What's New

z/OSMF tasks at a glance

Getting started with z/OSMF

Check browser and operating system

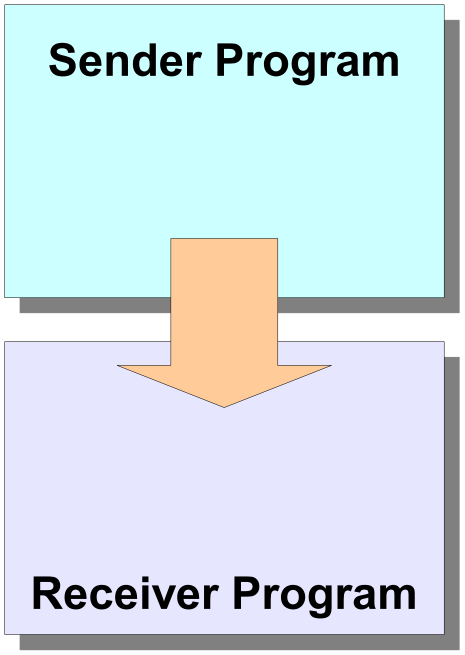
Enables easier acquisition of z/OS administrator skills

Staffing and administering System z and z/OS made simpler and easier with this facility

URL: `ibm.com/systems/z/os/zos/zosmf/`

Cross-Memory Communications

Program-to-program communications is possible between address spaces via a cross memory exchange:

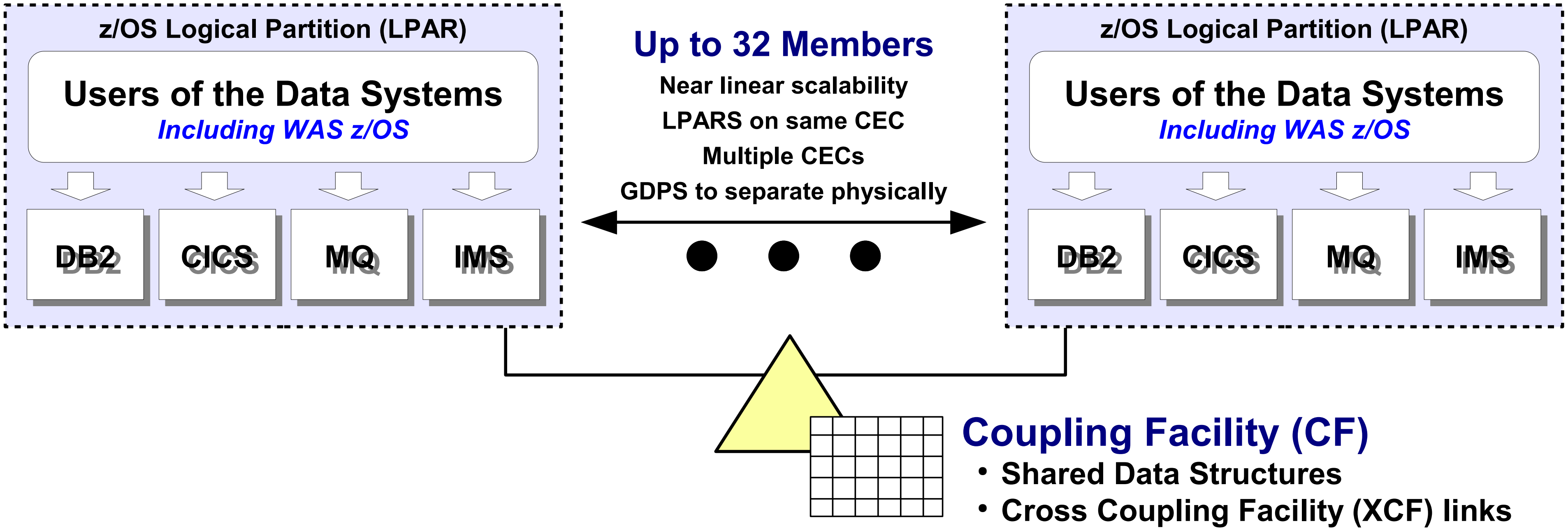


JDBC Type 2
CICS EXCI
MQ bindings mode
WOLA

- **Very low latency**
  The problem of latency tends to be additive in high volume, repetitive transactions
- **Very secure**
  Data can not be sniffed, intercepted or modified
- **Avoid encryption / decryption overhead**
  Since exchange is so secure, the need to encrypt may be eliminated
- **Security identity assertion across interface**
  Avoid coding identity "aliases" in different locations across enterprise
- **Avoid TCP/IP stack processing overhead**
  Reduces overall system CPU usage
- **Avoid serialization of parameters and result sets**
  Additional efficiency and reduced overhead
- **Single thread of execution across interface**
  Avoid task switching overhead
- **Propagate WLM enclave in most cases**
  Manage to a single WLM goal without having to manually coordinate classifications
- **Reduced complexity for debug / troubleshooting**
  Systems that span server and OS platforms tend to require separate data collection and analysis tools. When sender and receiver are in same OS environment one set of tools may be used.

Parallel Sysplex

This is a clustering technology with shared data model in the middle. It allows *multiple data system instances* with a *single logical shared copy of the data*:


z/OS Logical Partition (LPAR)
Users of the Data Systems
Including WAS z/OS
DB2
CICS
MQ
IMS
Up to 32 Members
Near linear scalability
LPARS on same CEC
Multiple CECs
GDPS to separate physically
z/OS Logical Partition (LPAR)
Users of the Data Systems
Including WAS z/OS
DB2
CICS
MQ
IMS

Coupling Facility (CF)
- Shared Data Structures
- Cross Coupling Facility (XCF) links to each participating LPAR

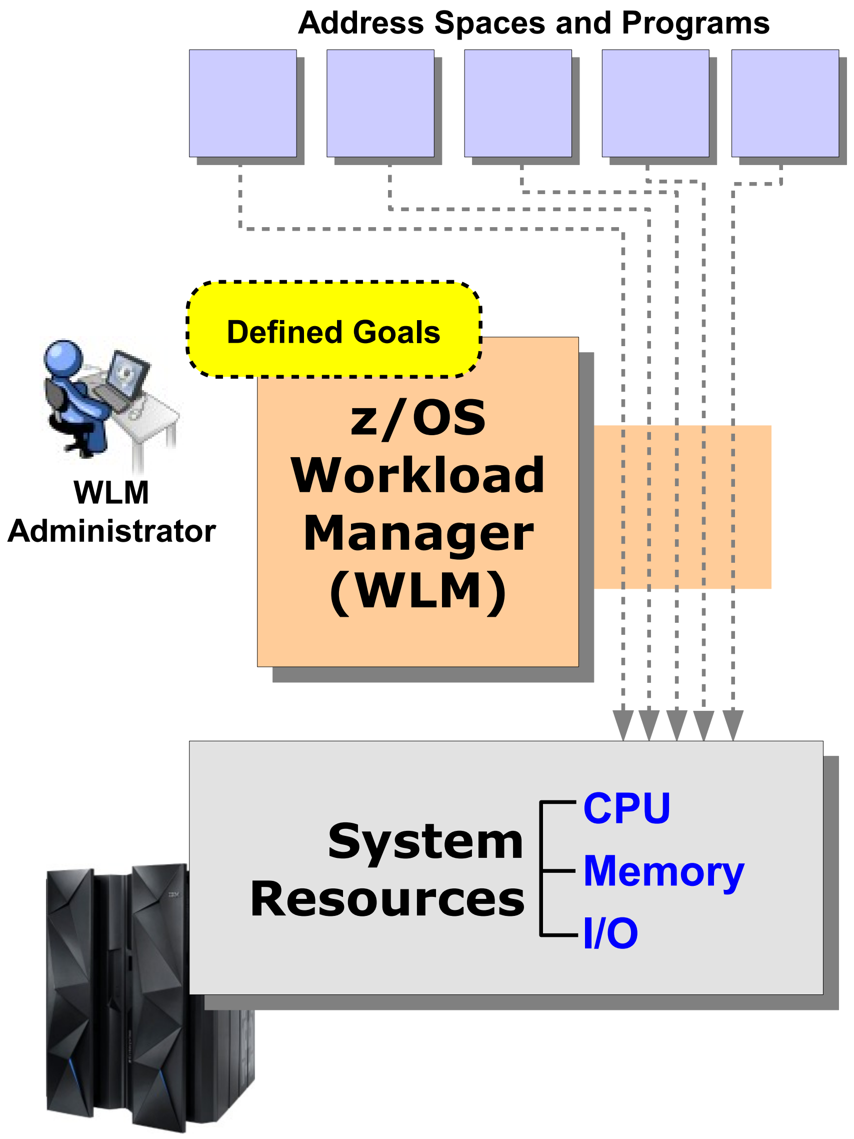
- **Each member of Sysplex has its own copies of data systems**
  Becomes a key part of the availability story ... more coming up
- **Data sharing and locking managed by Sysplex-aware data systems and CF**
- **Mature technology ... complexities and issues worked out so this is stable and scales very well up to 32 members**

Forms the lower-level cluster for WAS application server clustering above

A highly-available system involves integrated duplication and multiple levels of the architectural stack. More coming up later in chart deck.

z/OS Workload Manager

WLM is a key z/OS facility that monitors activity and adjusts system resource access (CPU, memory. I/O) according to the priority goals you define:



- **z/OS is a shared environment by its design. Many things going on at once.**
- **z/OS work tends to organize around several categories of relative priority**
- **WLM manages access to shared system resource according to goals you define that express relative priorities**
- **WLM is what allows z/OS systems to operate at very high levels of utilization**

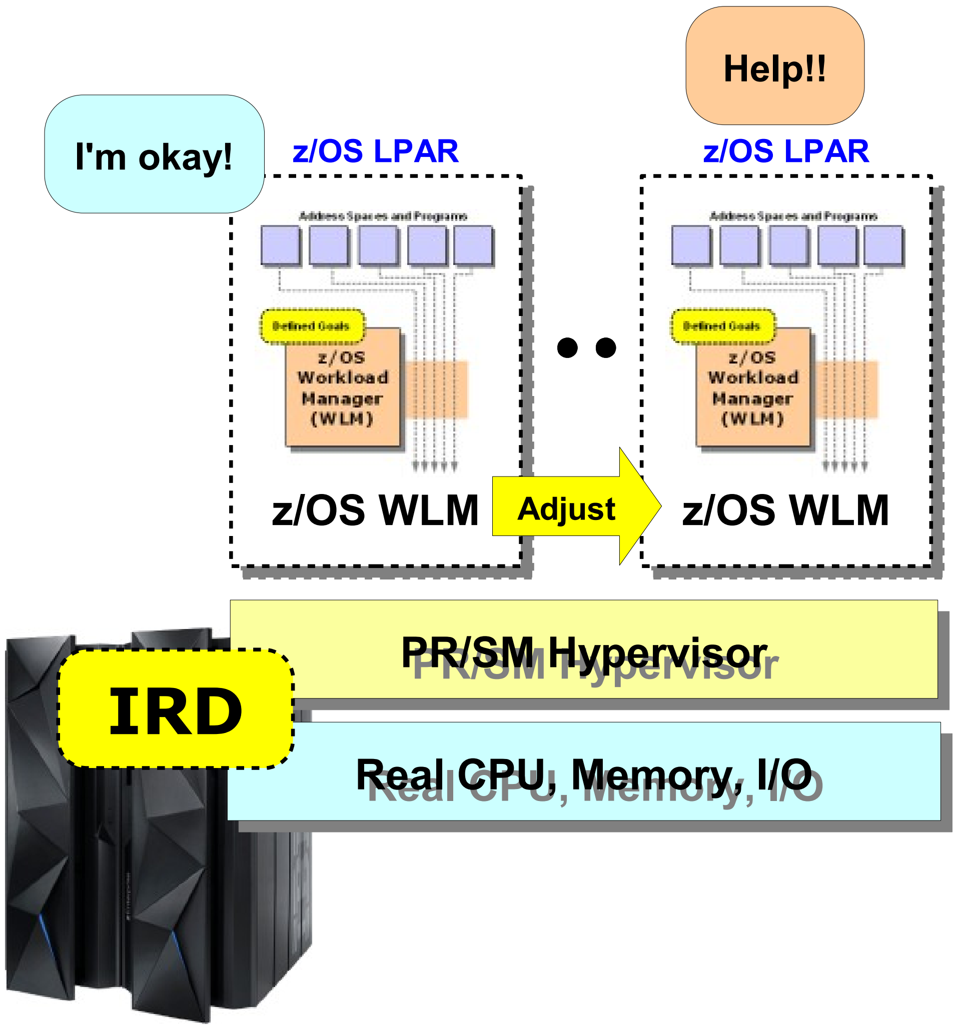
  Because WLM is intelligently managing access based on defined goals. When system utilization increases WLM helps reduce unproductive contention for heavily utilized resources.

WAS z/OS has extensive integration with WLM

We will explore that integration in the section titled "Taking Advantage of the Platform"

Intelligent Resource Director (IRD)

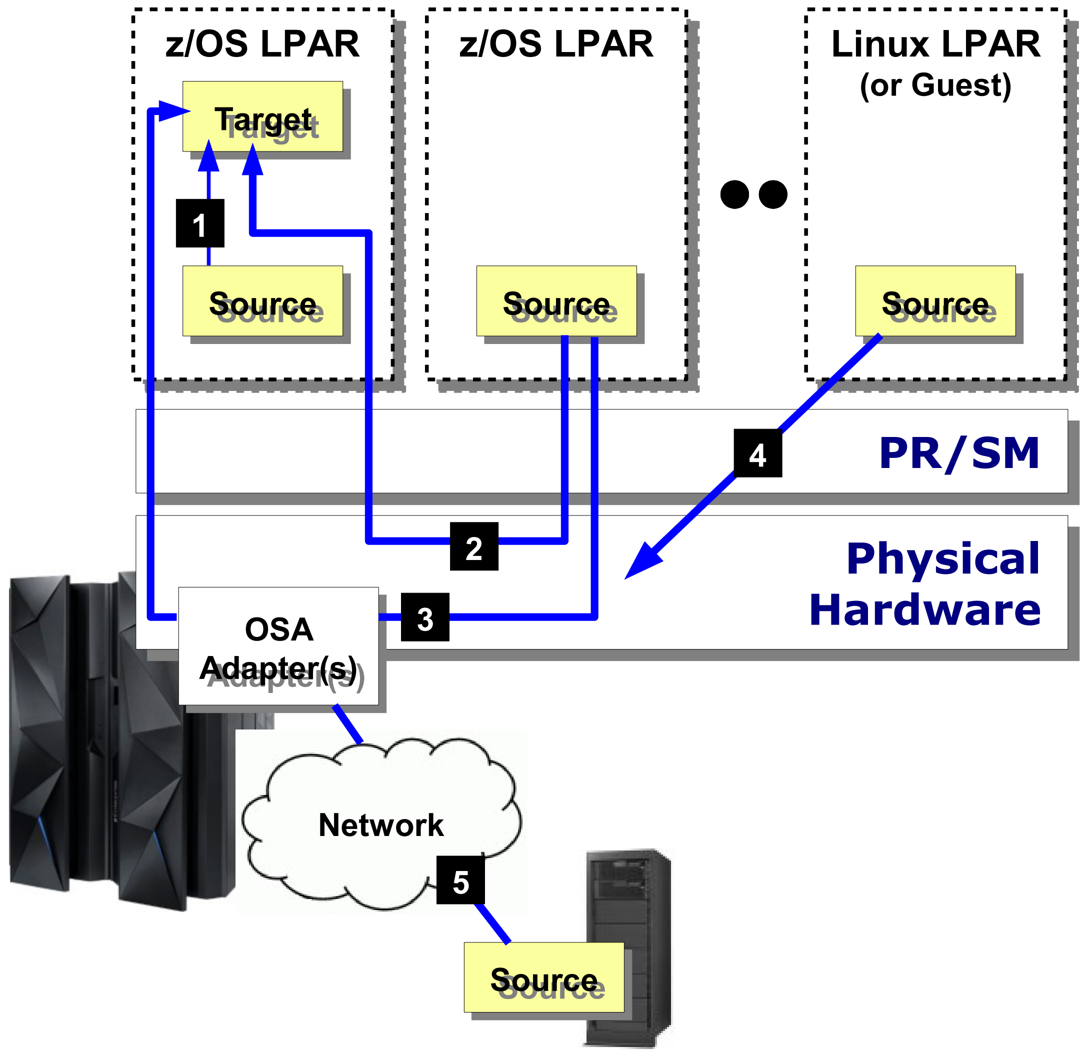
IRD provides a way for WLM in z/OS partitions to coordinate with PR/SM to make dynamic adjustments of resource between partitions ... based on WLM goals



- **WLM in each LPAR is managing work according to the service goals defined**
- **WLM service goals are known to each instance of WLM in a Parallel Sysplex**
- **If one LPAR is falling behind in its goal attainment, and WLM sees other LPARs may donate resources, then WLM uses interface to PR/SM to make adjustments**
- **Resources adjust back when WLM determines all goals can be met**

TCP/IP Networking and Network Optimization

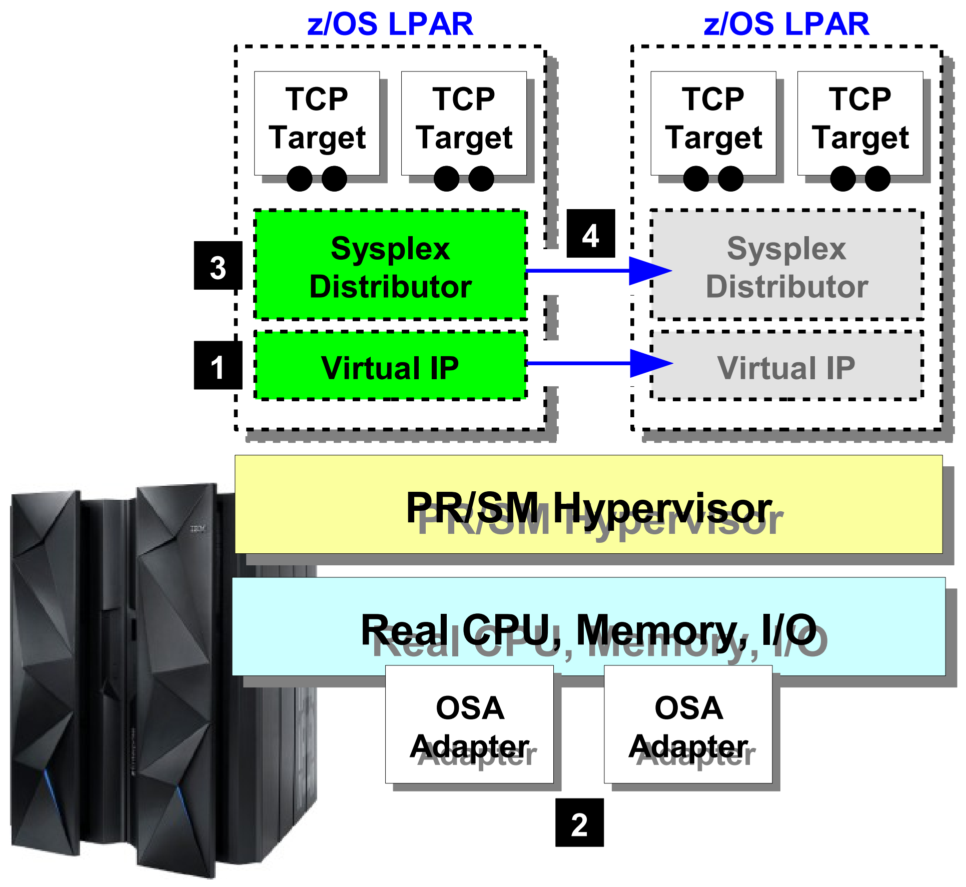
System z has a strong TCP/IP networking story, including network optimization based on proximity of source to target:



1. **Same z/OS LPAR**
   z/OS TCP knows when source and target on same LPAR; eliminates IP code-path and does cross-memory TCP only exchange
   For WAS z/OS, WOLA is a cross-memory mechanism that can eliminate TCP entirely. More on WOLA later in the presentation
2. **Cross-LPAR Hipersockets**
   Hipersockets is a TCP network mapped over memory. Very efficient and low-latency mechanism for cross-LPAR communications
3. **Cross-LPAR over OSA**
   If not Hipersockets then down to OSA adapter but not over any wires
4. **From Linux for System z**
   Linux for System z may participate in Hipersockets or TCP over OSA communications to other LPARs.
   If z/VM used for Linux virtualization, then z/VM provides virtual LAN between Linux guests
5. **From Off-Platform**
   Standard TCP/IP networking applies

Virtual IP and Sysplex Distributor

z/OS TCP/IP has the facility to host virtual IP addresses, to dynamically re-host them in the event of loss, and to distribute work based on WLM-advised routing metrics:



1. Virtual IP Address

Allows processes to listen on IP addresses separate from the IP of the physical adapter

Virtual IP can non-disruptively move to another LPAR, either through operator command or due to outage

2. Multiple Physical Adapters

System z permits the installation of multiple network adapters with Virtual IP availability across any

3. Sysplex Distributor

A WLM-iintelligent TCP connection placement mechanism.

Will place TCP connections to same LPAR or other LPARs in Sysplex from hosting stack

Sysplex Distributor function capable non-disruptively moving to another LPAR, either through operator command or due to outage

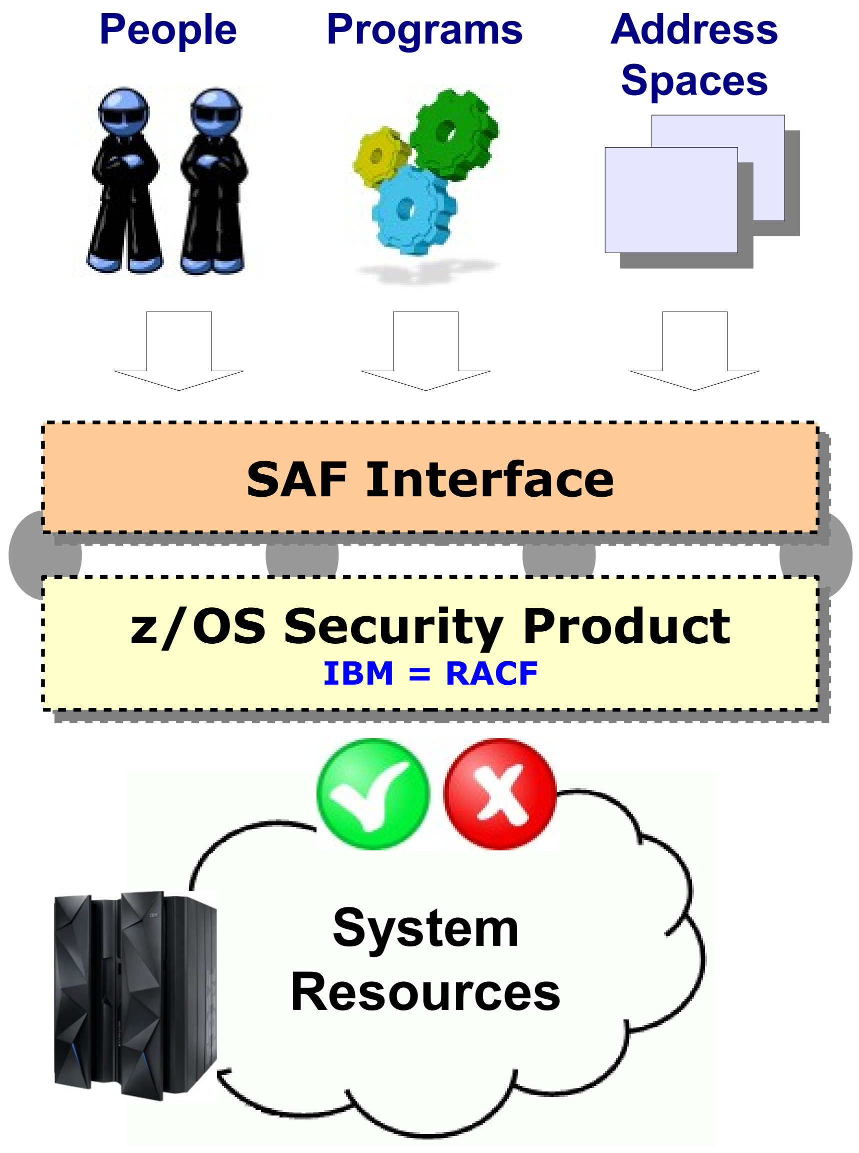
4. Non-Disruptive Move

Virtual IP and Sysplex Distributor may dynamically and non-disruptively move to another LPAR in the Sysplex.

This capability is part of the broader High Availability story discussed later in the presentation

z/OS Security Access Facility (SAF)

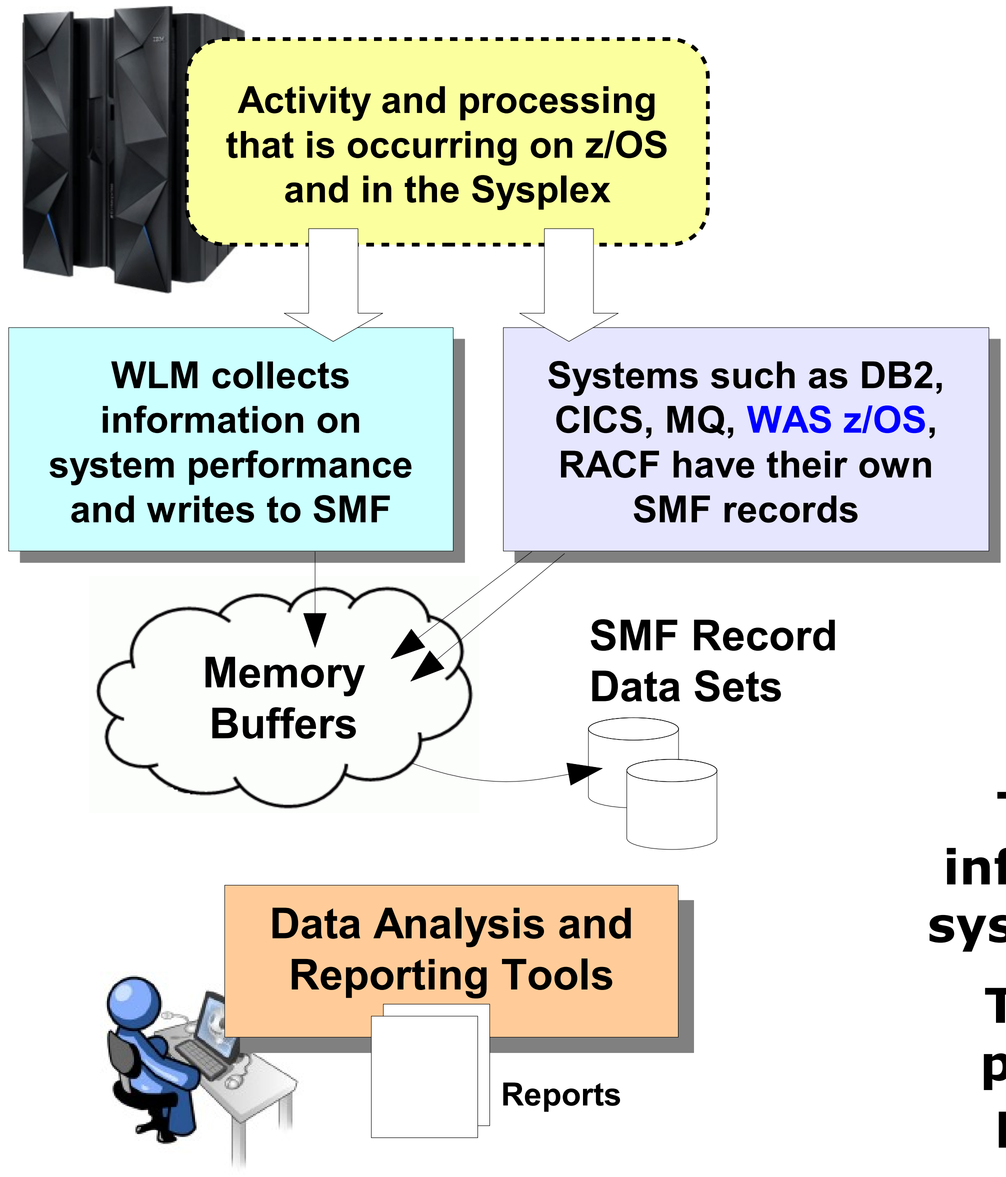
SAF is a system security interface, behind which you may run one of several security products. IBM's offering is RACF.



- **SAF provides the interface through which users gain access to z/OS resources**
- **The SAF interface is backed by a security product (IBM's is RACF)**
- **The security product authorizes or denies access to the resource based on defined profiles**
- **RACF does many things:**
  - **Authenticates users**
  - **Check authorization to access resource (data sets, issue commands, use TCP ports, application roles, much more)**
  - **Stores digital certificates**
  - **Logs access attempts and violations**
- **RACF has EAL5 security certification**
- **Provides a consolidated point for security administration, with a strong tradition of careful security policy management**

z/OS Monitoring and Reporting: RMF and SMF

RMF provides real-time and historical reporting of performance information for a system, SMF provides a record buffering and writing facility for capturing data:


Activity and processing that is occurring on z/OS and in the Sysplex
WLM collects information on system performance and writes to SMF
Systems such as DB2, CICS, MQ, WAS z/OS, RACF have their own SMF records
Memory Buffers
SMF Record Data Sets
Data Analysis and Reporting Tools
Reports

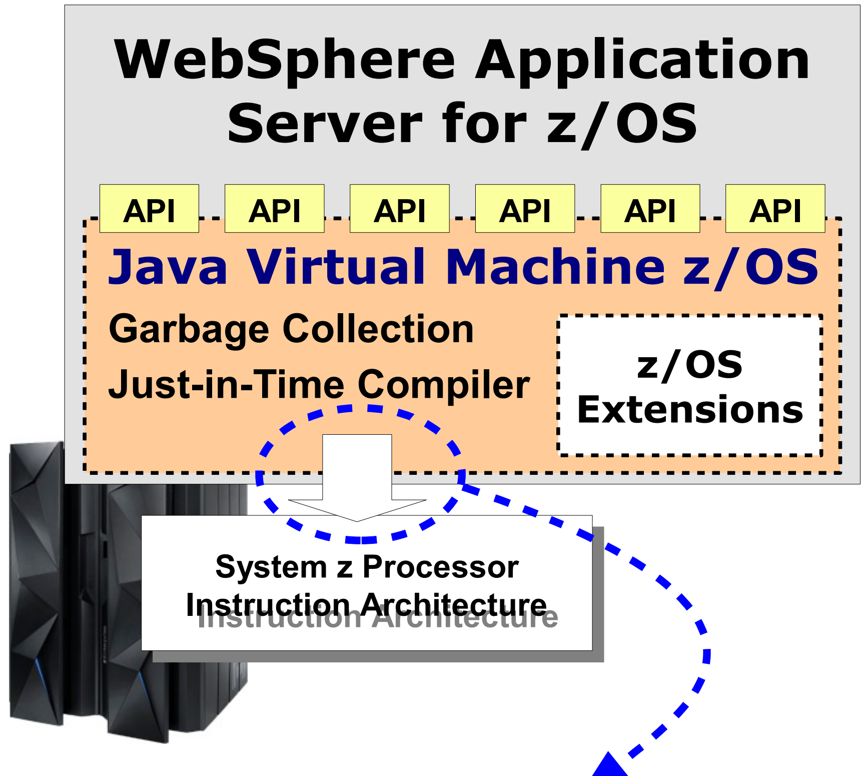
- **WLM collects and writes SMF records related to various points of system performance**
- **Other systems write their own SMF records for capturing more detailed activity records**
- **WAS z/OS writes its own SMF record with detailed information on request activity received**
- **RMF is a reporting tool that allows you to better understand the data held in the SMF records**

The objective is to provide the information needed to understand system utilization and performance

This is valuable for purposes of performance planning, capacity planning, and cost chargeback

IBM Java SDK for z/OS

IBM produces a Java SDK for System z and z/OS. Again, "Java is Java" at the API layer, but below that IBM has done considerable work to optimize:

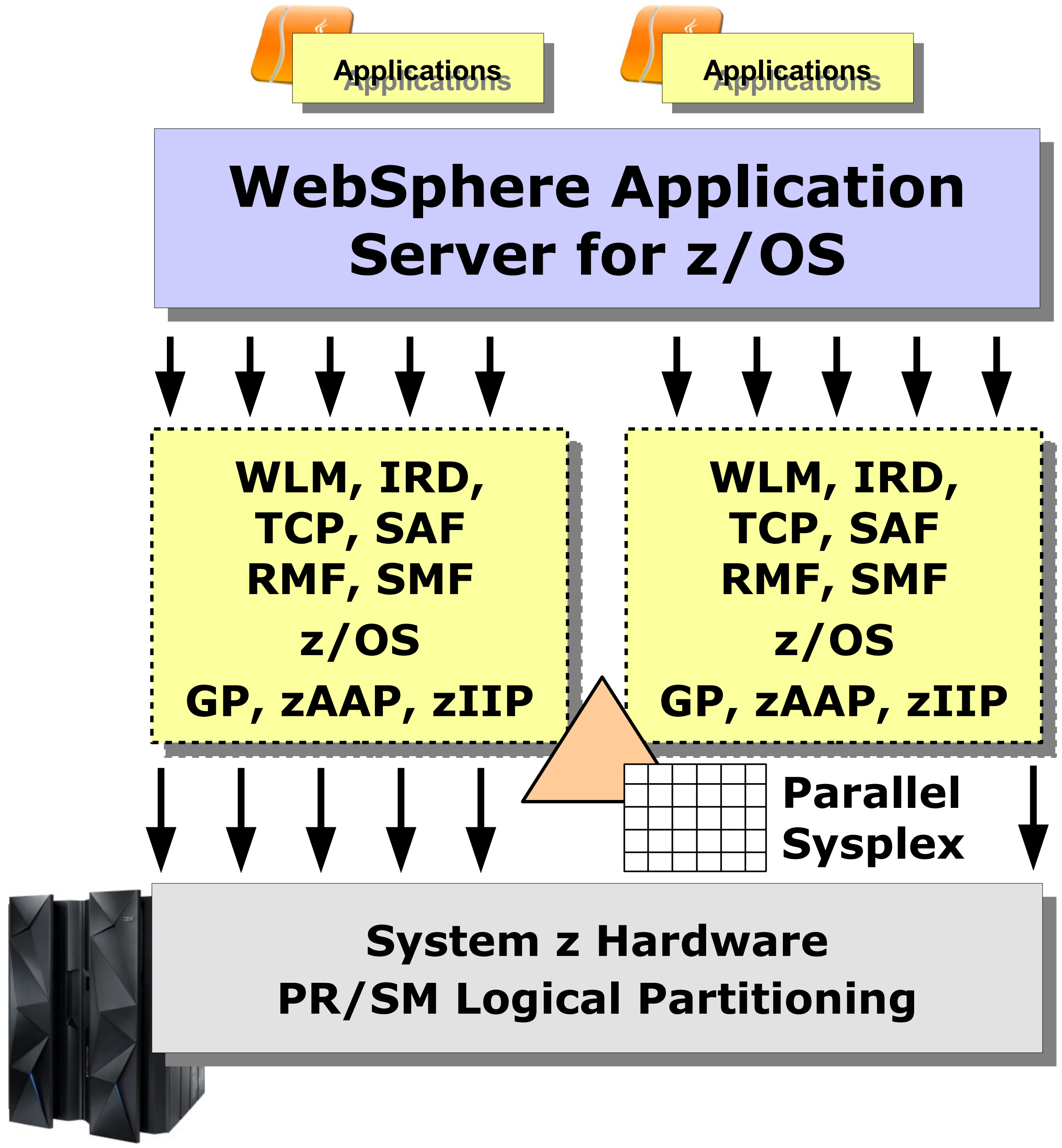

WebSphere Application Server for z/OS
API
API
API
API
API
API
Java Virtual Machine z/OS
Garbage Collection
Just-in-Time Compiler
z/OS Extensions
System z Processor Instruction Architecture

IBM z/OS SDK and z/OS dispatcher coordinate to offload Java work to specialty engines

- zAAP engines
- zIIP (via zAAP-on-zIIP)

- SDK 6 or SDK 7, 31-bit or 64-bit
- The application facing APIs are compliant with the open SDK specification (V6 or V7)
- The JVM is entirely IBM-designed and IBM-written code
  - Based on decades of IBM knowledge of OS design
- The JVM on z/OS is aware it's on z/OS and takes specific advantage of the platform:
  - JIT exploitation of HW instructions (z10, z196, EC12)
  - Customizable garbage collection policies
  - Shared classes; ahead-of-time (AOT) compilation
  - 64-bit compressed references and large page support
- Extensions for z/OS include:
  - Security extensions (see URL shown below)
  - z/OS Java Record I/O functions
  - JZOS Toolkit (for Java-to-z/OS function access)
- Java z/OS webite:
  - `ibm.com/systems/z/os/zos/tools/java/`

Section Wrap-Up


Applications
Applications
WebSphere Application Server for z/OS
WLM, IRD, TCP, SAF RMF, SMF
z/OS
GP, zAAP, zIIP
WLM, IRD, TCP, SAF RMF, SMF
z/OS
GP, zAAP, zIIP
Parallel Sysplex
System z Hardware
PR/SM Logical Partitioning

- **System z, z/OS and Parallel Sysplex provides the foundation for WAS z/OS and your applications**
- **Now we will explore how WAS z/OS takes advantage of this foundation**

WebSphere Application Server z/OS

Conclusion

Overall Presentation Conclusion

The value statement for WAS z/OS is *not* based on differences in the application API layer

"WAS is WAS" at that layer ... which is a very good thing from perspective of enterprise application design, tooling, test and development

The value of WAS z/OS derives from the value of z/OS ... and how System z, z/OS and Parallel Sysplex provide a rich platform for operations

The story we just finished is one of WAS z/OS taking advantage -- directly and indirectly -- of the System z platform ... to your benefit in terms of reliability, scalability, availability and servicability

Document Change History

January 3, 2013	Original document

End of Document