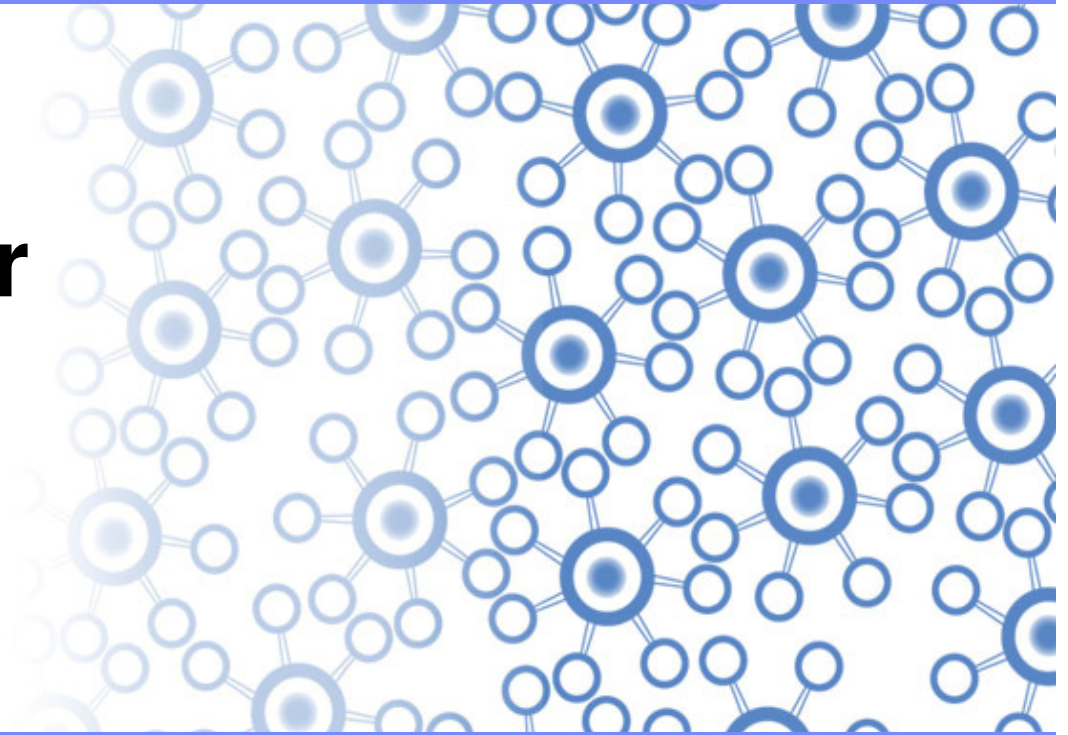




IBM Washington Systems Center

WP101532 on [ibm.com/support/techdocs](http://ibm.com/support/techdocs)

# Why WebSphere Application Server for z/OS



Version Date: 6/14/2011

© 2009 IBM Corporation

## Document Change History

July 30, 2009	Original Edition
July 31, 2009	WP101532 document number applied and republished
August 3, 2009	Various updates based on review by Dave Follis
June 14, 2011	Various updates based on WAS z/OS V8

The WebSphere Application Server for z/OS support team at the Washington Systems Center consists of: **John Hutchinson, Mike Kearney, Louis Wilen, Lee-Win Tai, Steve Matulevich, Mike Loos and Don Bagwell.**

**Mike Cox**, Distinguished Engineer, serves as technical consultant and advisor for all of our activities.

Particular thanks to **Dave Follis** of the WAS z/OS development staff for his review and guidance on this presentation.

For questions or comments regarding this document, send e-mail to Don Bagwell at [dbagwell@us.ibm.com](mailto:dbagwell@us.ibm.com)

## Introduction

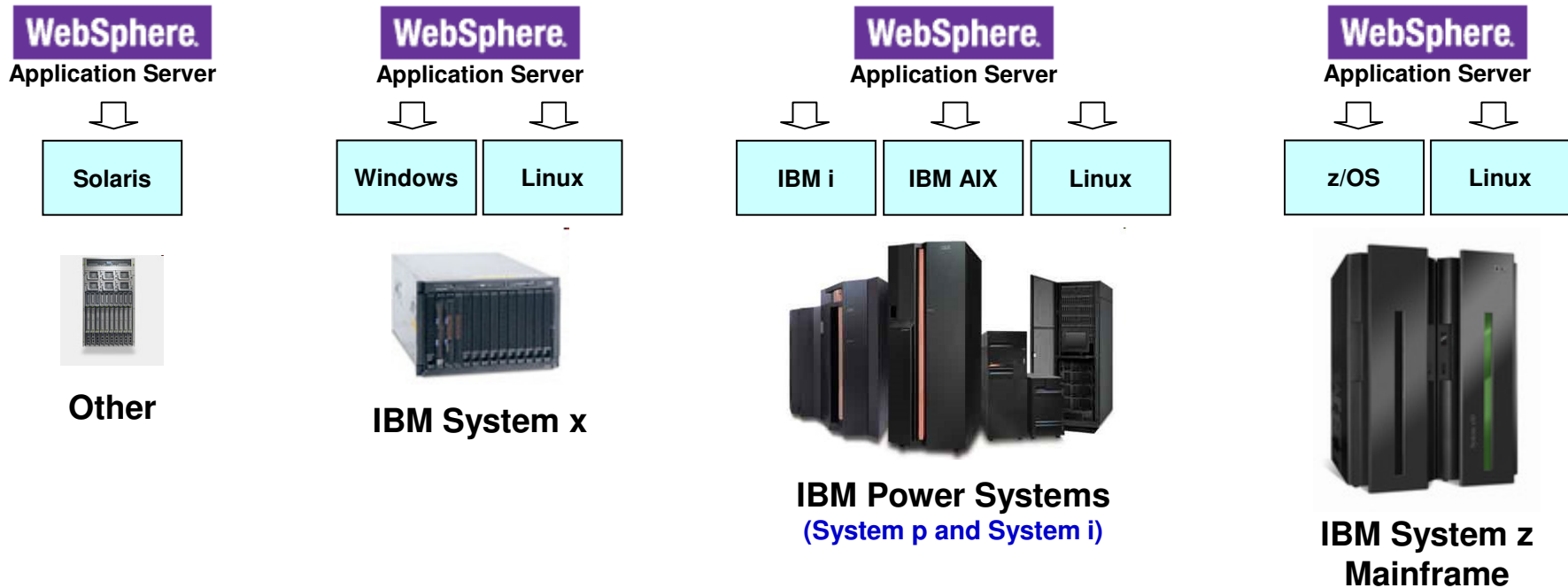
**The objective of this presentation is to illustrate the key technical differentiators of WebSphere Application Server running on System z and z/OS platform.**

**We will also show how those technical attributes contribute to business value.**

## WAS and the Platform Decision

Assume the decision for WebSphere Application Server has been made.

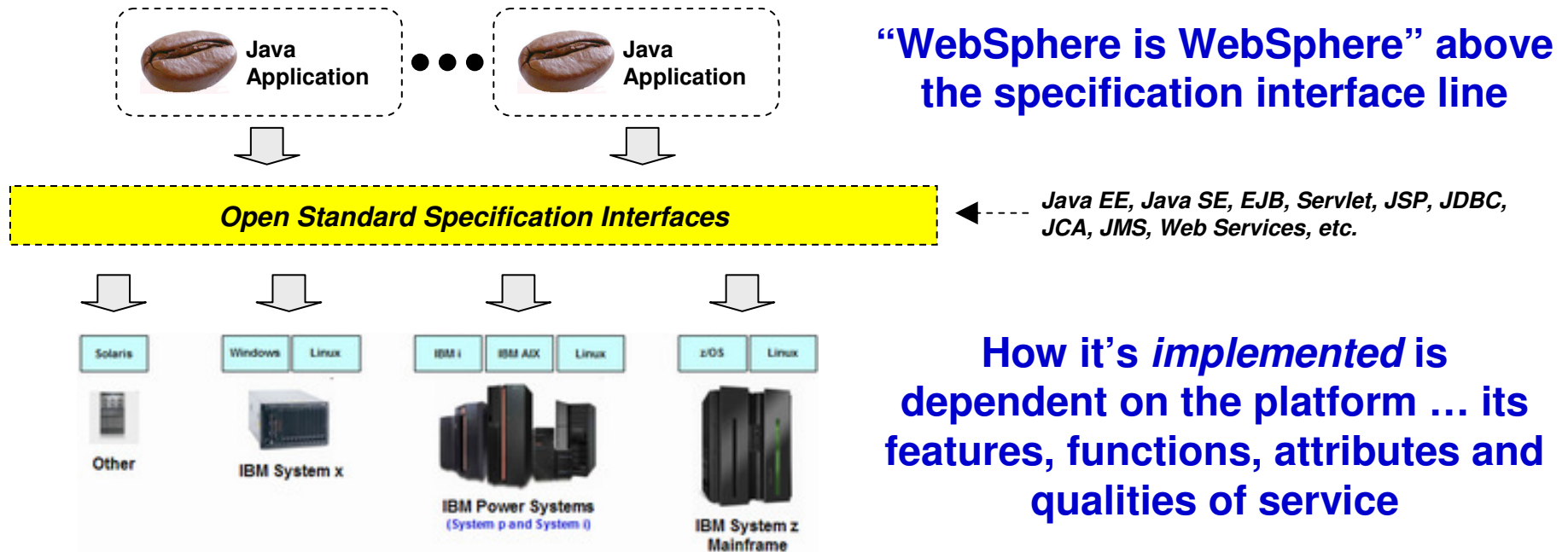
*WAS is a cross-platform product offering*



*Making the hosting decision ...*

## Very Important Starting Concept

This point can't be stressed enough -- the differentiation is *not* in the open standard specification support offered. *That is common across platforms!*

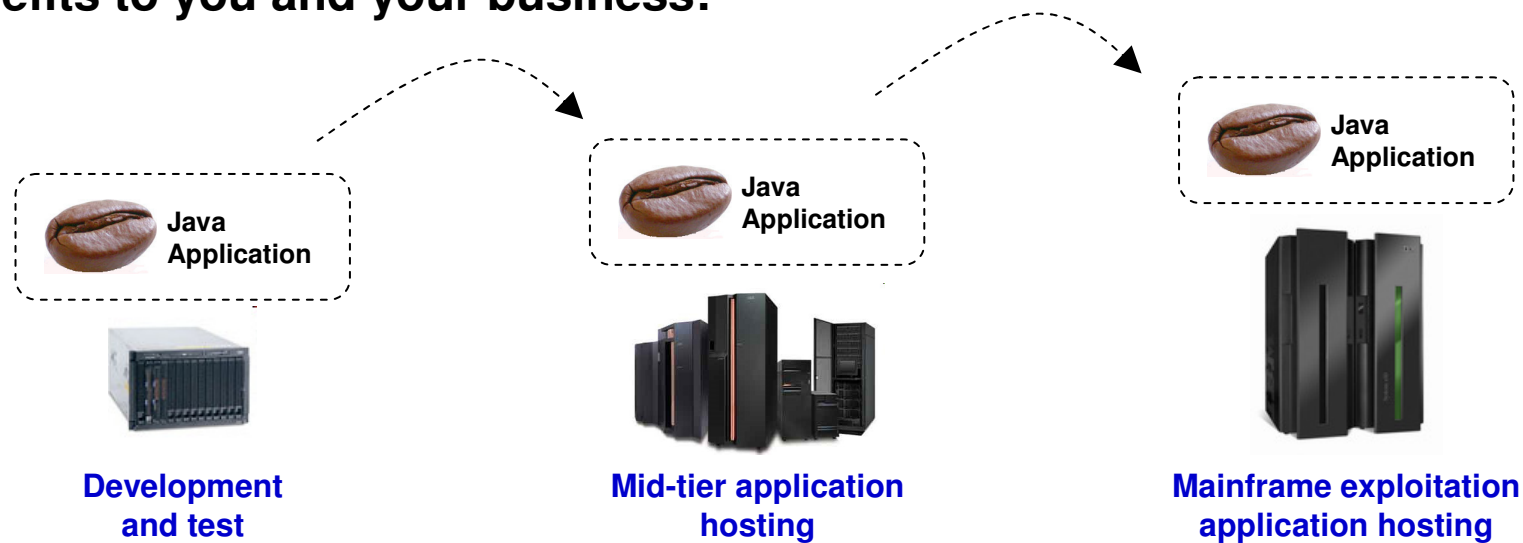


**Starting with V6.0 the code base merged into one with a single source**

Problems with code divergence solved. Code has ability to detect platform and invoke platform-specific exploitation as appropriate.

## Key Benefits of Alignment Across Platforms

The alignment of specifications across all the IBM platforms brings several key benefits to you and your business:



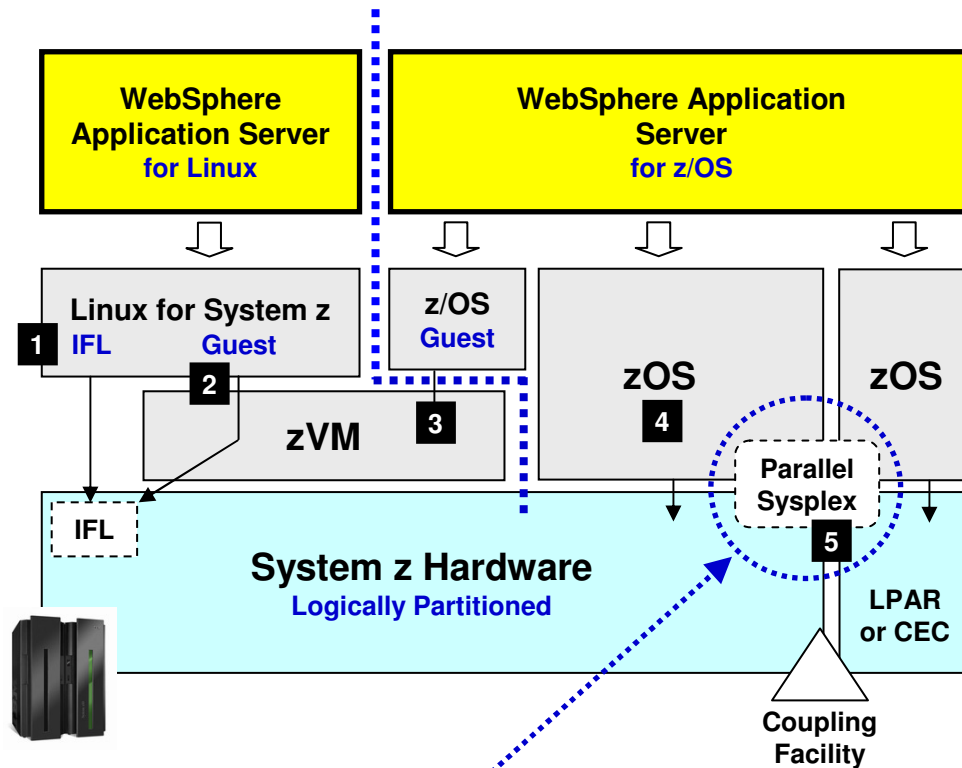
- Avoids costly rewrite of applications to change platforms; reduces the testing effort
- Ability to promote applications “up the ladder” without concern about loss of interface function
- Ability to architect application designs that span multiple platforms without having to make sacrifices based on the platform
- Ability to settle on a common set of application tooling across all platforms
- Ability to have an essentially common management interface across all the platforms

*Minor differences exist in areas related to platform specifics such as starting servers. More in a bit.*

**These benefits are intended ... this is why “WebSphere is WebSphere” across the platforms**

# System z, z/OS, Linux, and WebSphere Application Server

Here's a mapping of how the two flavors of WebSphere Application Server can be hosted on System z hardware:



WebSphere z/OS design and implementation capitalizes on the Sysplex environment

Much more to follow

1. **Linux for System z directly on IFL**  
Possible, but not very common. Solution where no zVM skills exist
2. **Linux for System z as guest on zVM**  
Very common. This provides excellent virtualization with zVM with Linux running as a guest. Runs on the IFL.
3. **z/OS as guest on zVM**  
Another example of zVM's virtualization capabilities. WAS z/OS as guest typically in a development or test environment.
4. **z/OS in a non-Sysplex environment**  
WAS runs directly on z/OS with no zVM virtualization. No Sysplex more common in test environments or small production.
5. **z/OS in a Parallel Sysplex environment**  
This is the flagship environment. This is where high availability, scalability and maximum platform exploitation takes place.



## Preliminary Conclusion

**WebSphere is WebSphere at the specification line and above**

**Therefore, there is no platform differentiation *at that level***

**Differentiation occurs *below* the specification line**

**That's where the attributes of the platform are exploited by the implementation of WAS on that platform**

## Three Big Questions:

- 1. What are the qualities and attributes of the System z and z/OS platform?**
- 2. To what degree does WebSphere Application Server exploit those qualities and attributes?**
- 3. How do those qualities and attributes contribute to meeting your key business objectives?**

**Addressing those three questions is the objective of this presentation**



## Outline of Presentation

Going forward from here we'll touch on the following topics:

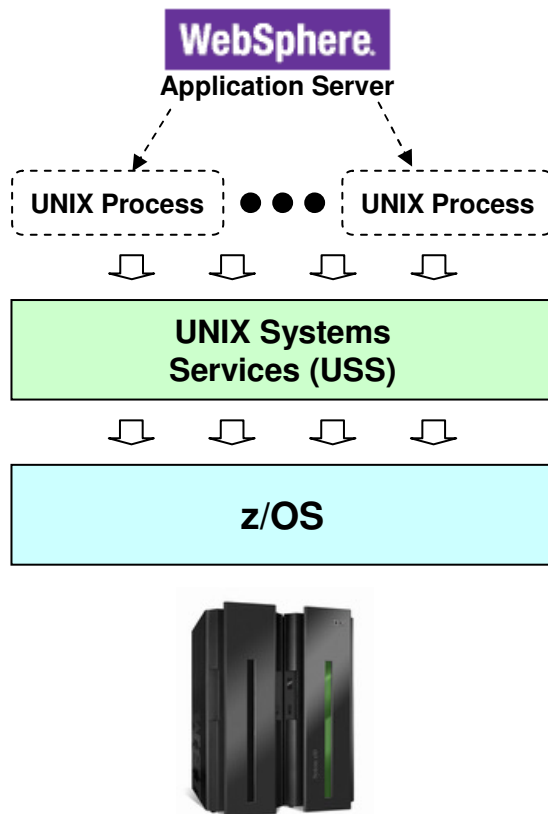
- **Platform exploitation and its two forms: passive and active**  
Both are important; active is where WAS for z/OS shines. We'll have a sub-focus in this section on the IBM SDK for z/OS and see how it too exploits the platform
- **Take a look at WAS on Linux for System z**  
To understand the key differences and position the two System z offerings
- **What makes Java code “z ready”**  
Hint: this is going to be a very short section!
- **Availability and scalability**  
We explore how WAS z/OS uniquely exploits platform attributes to provide both
- **Quick Look at Other WAS-Based Solutions**  
One chart to make the point that these solutions can ride on the benefits derived by WAS z/OS
- **Administration and skills**  
A common question is how current skills map to the WAS z/OS environment
- **The issue of cost**  
It's on everyone's mind ... so we try to tackle it head-on

# The Issue of Platform Exploitation

*Establishing key concepts related to platform exploitation*

## What *Could* Have Been ... But Thankfully Was Not

WebSphere Application Server “for z/OS” could have been implemented as a pure UNIX application, with no direct exploitation at all:



**Question: would there be any platform exploitation?**

**Answer: yes, but it would be very *passive*.**

Examples: redundant design of the hardware platform; efficient and scalable I/O subsystem; storage protection architecture; virtualization at the LPAR level; etc.

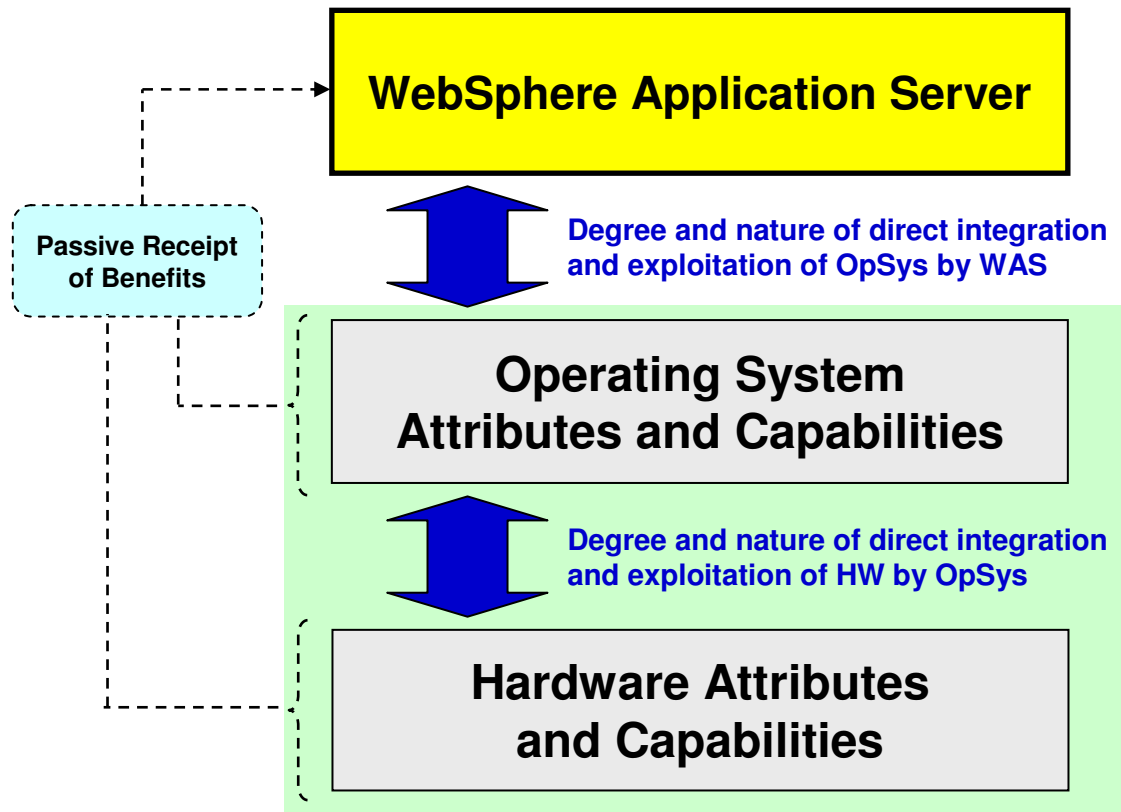
**This positions the concept of *active* and *passive* exploitation**

- Active**     direct exploitation of platform qualities and attributes by the code under the specification interfaces
- Passive**    benefits that derive simply by running on the platform ... or as some say, “just showing up”

*Let's expand on that a bit ...*

## Multiple Levels of Exploitation Taking Place

We need to understand that there are benefits from the hardware design, benefits from the operating system design, *and benefits from the integration between the two*



**Not all hardware designs are equal**

**Not all operating systems are equal**

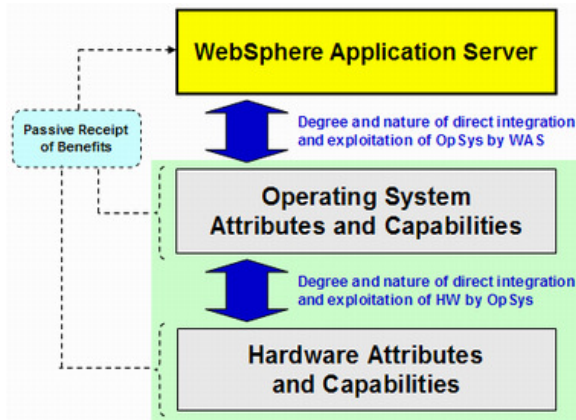
**Not all operating systems have the same degree of integration with the hardware**

Example: z/OS only runs on System z ... there are no tradeoffs to enable multi-platform flexibility. The OS is optimized for the hardware; the two are developed jointly

**That doesn't mean System z and z/OS are appropriate for all cases  
Nor does it mean other platforms and operating systems can do what System z and z/OS can do**

## Three Responses to “Value Statements”

When the features and functions of the System z and z/OS environment are discussed, we often see people respond in three ways:



### Understand and Agree

People with a background in System z. Focus then turns to the way WAS z/OS exploits those features.

### News to Me ... Want Better Understanding

Becomes a matter of going into greater detail on each of the functions. It can be a broad topic area and often requires time and a whiteboard.

We welcome these discussions. Please ask for more details if you're unfamiliar with the feature.

### Have Heard Before and Have Different Opinion

Addressing this involves understanding the nature of the differences between our view and yours.

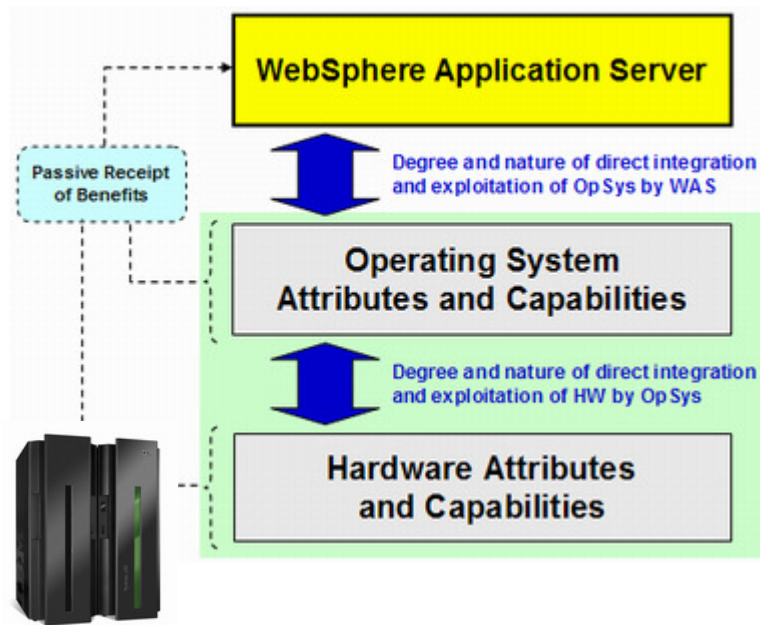
We welcome this discussion as well. Different perspectives are always useful. We'll take the discussion offline and see where we can agree and where differences in opinion still remain.

# Passive Exploitation Benefits

*Benefits derived by WebSphere Application Server by virtue of running on the platform*

## Passive Benefits Fall Into Several Categories

Programs that run on System z and z/OS receive passive benefits in a couple of different areas:



### Hardware

- Inherent maturity and stability of design
- Redundancy and flexible updates
- Balanced design offers very high throughput
- Mature and proven virtualization through LPAR

### Operating System

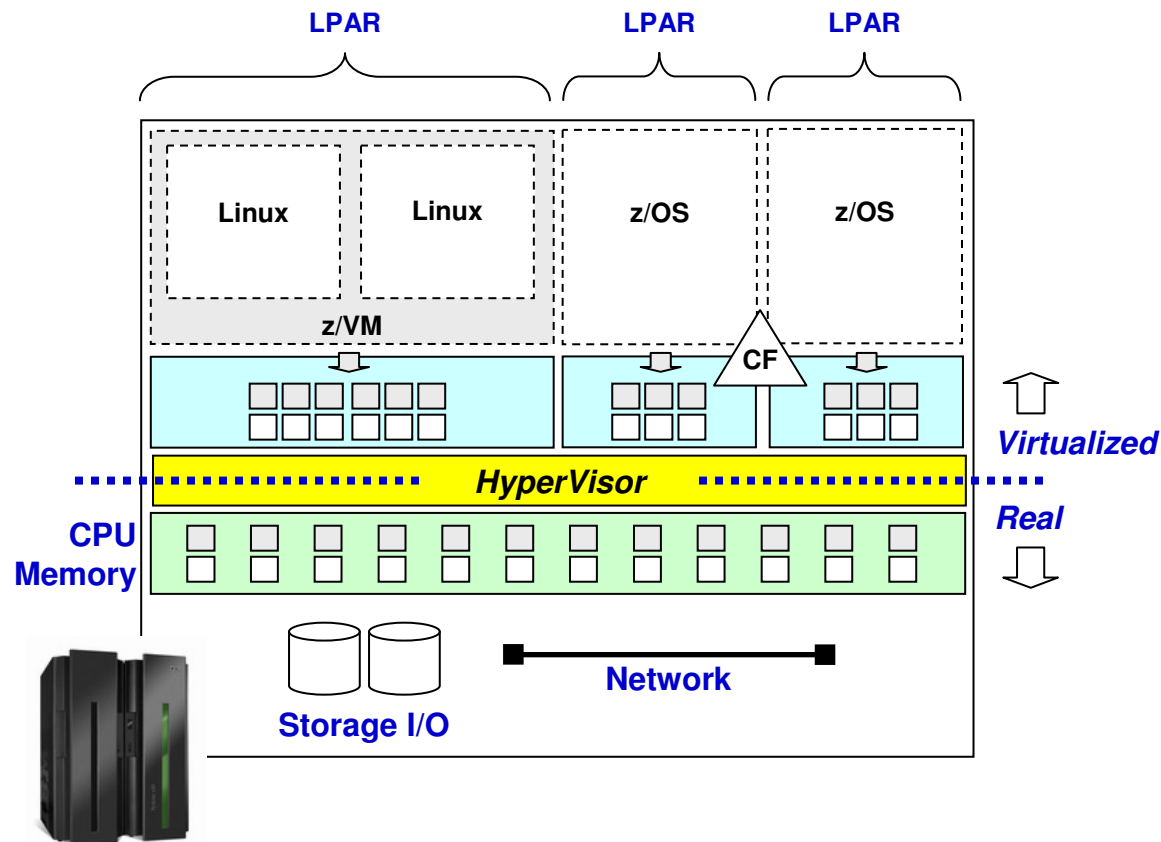
- Tight integration with server hardware design
- Extremely mature architecture
- Storage protection
- Workload Manager (WLM)
- Intelligent Resource Director (IRD)
- Local TCP optimization
- Mature systems management tools
- Proven disaster recovery capabilities

Let's explore some of these at a high level



## Hardware Virtualization -- Logical Partitions (LPAR)

An extremely mature virtualization technology that allows real resources to be shared across multiple logical partitions, each entirely separate



### Proven virtualization technology

- Years of proven reliability
- Partitioning of HW into logical partitions
- Further virtualization using z/VM with guest machines

### Each LPAR entirely separate from the other

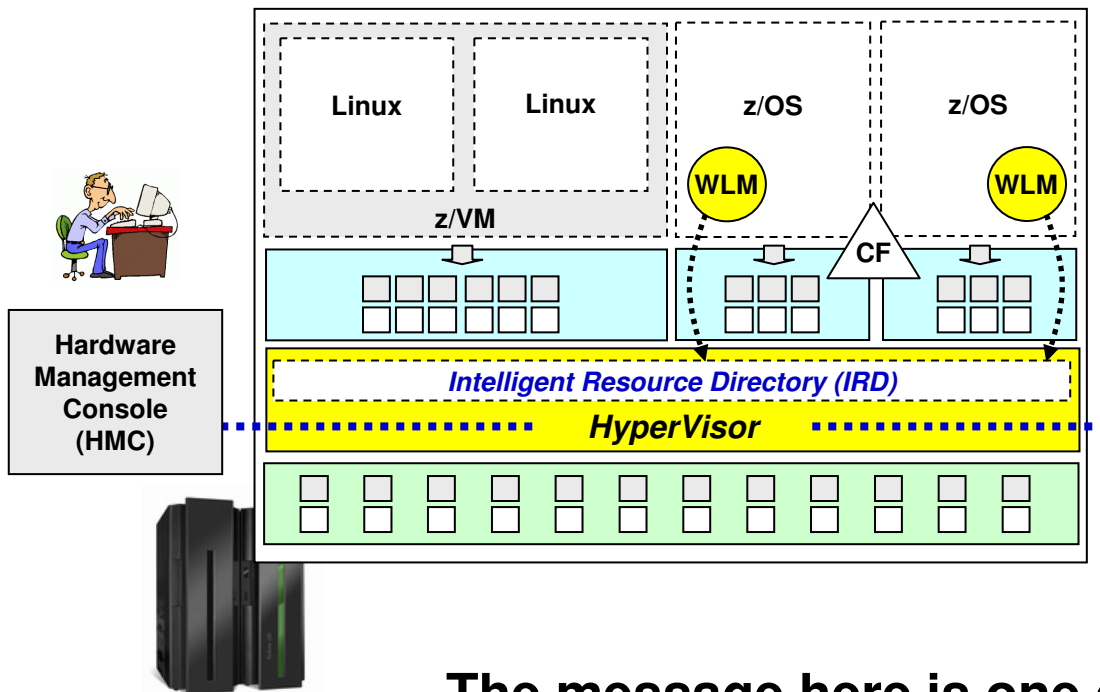
- Hypervisor protects one LPAR from monopolizing resources above what its allocated
- Complete memory isolation, so no overlay concerns
- Complete operating system isolation, so all elements of OS instances separated
- Complete network isolation, so no concerns about security breaches

## Benefits of consolidation with the advantages of isolation

That's what virtualization is all about. The difference is one of maturity and capability. The technical differences between virtualization approaches can become a complex topic quickly. Point here is that System z LPAR has a proven production track record

# Dynamic Modification of LPAR CPU Resource Allocations

Manual and automatic ...



Non-disruptively add CPU to the machine and assign to LPAR

Allow IRD to dynamically move CPU between LPARs

Dynamically vary I/O capacity across LPARs to solve bottlenecks

With z/OS you may have WLM advise IRD to reallocate CPU and I/O between LPARs in the Sysplex

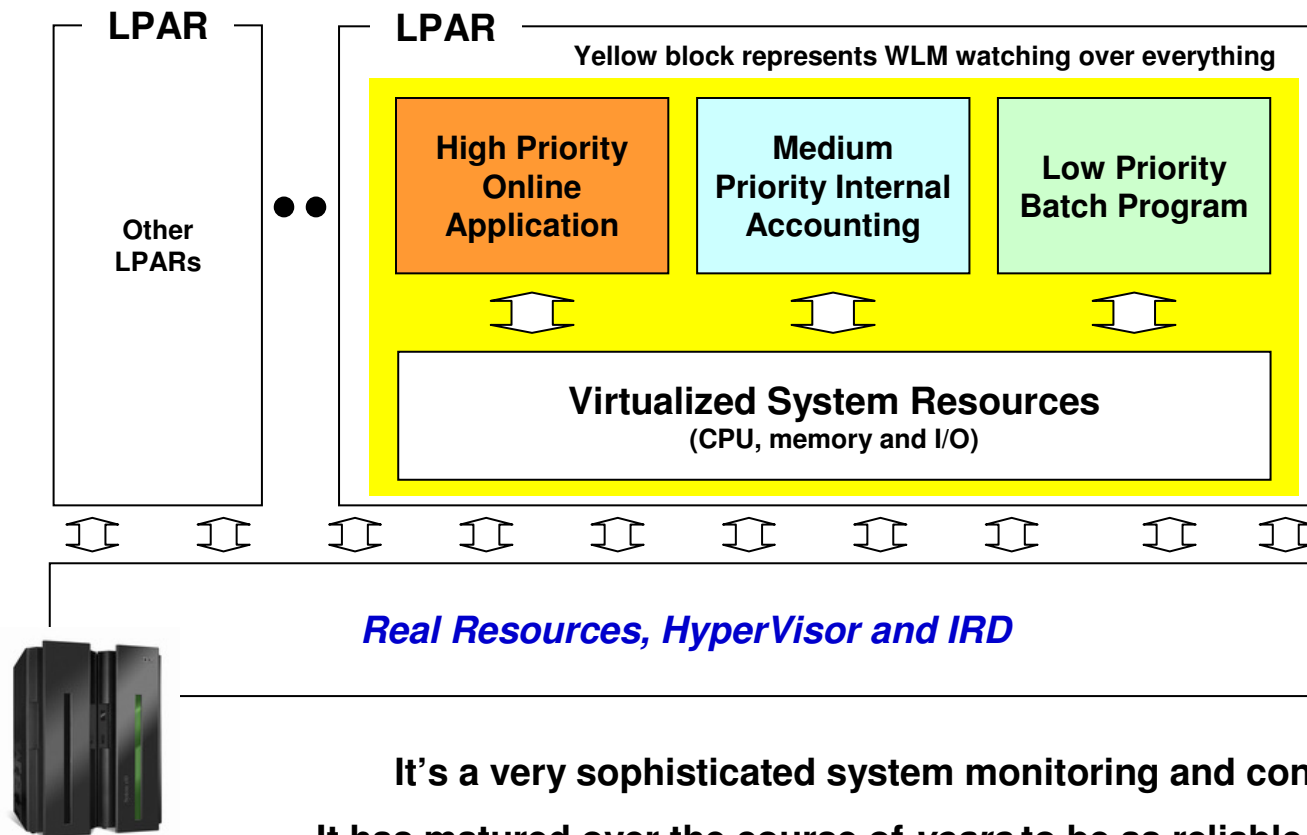
The message here is one of dynamic flexibility.

The pace of change is increasing ... rigid designs hinder rapid exploitation of opportunities

System z LPAR technology coupled with z/OS WLM provides a proven flexible and dynamic environment

## Workload Manager (WLM)

A component of the z/OS operating system, WLM keeps close watch on key system metrics and manages resources towards meeting your defined goals



### Five pieces to this:

1. WLM's real time monitoring of the overall system resource utilization
2. The WLM service level goals you've defined that determine how WLM will manage resources
3. WLM's comparing your service level goals against the actual system performance on a program by program basis
4. WLM's reallocating resources within the LPAR to make sure goals are met
5. WLM advising IRD if resource allocation across LPARs is needed

It's a very sophisticated system monitoring and control mechanism

It has matured over the course of *years* to be as reliable and effective as it is

Other solutions claim to offer "workload management" but are often rather weak in function compared to how z/OS WLM operates

## Virtual IP and Sysplex Distributor

Is a function of TCP on z/OS which allows you to “hide” duplicated resources behind a single IP address with WLM-assisted TCP connection placement

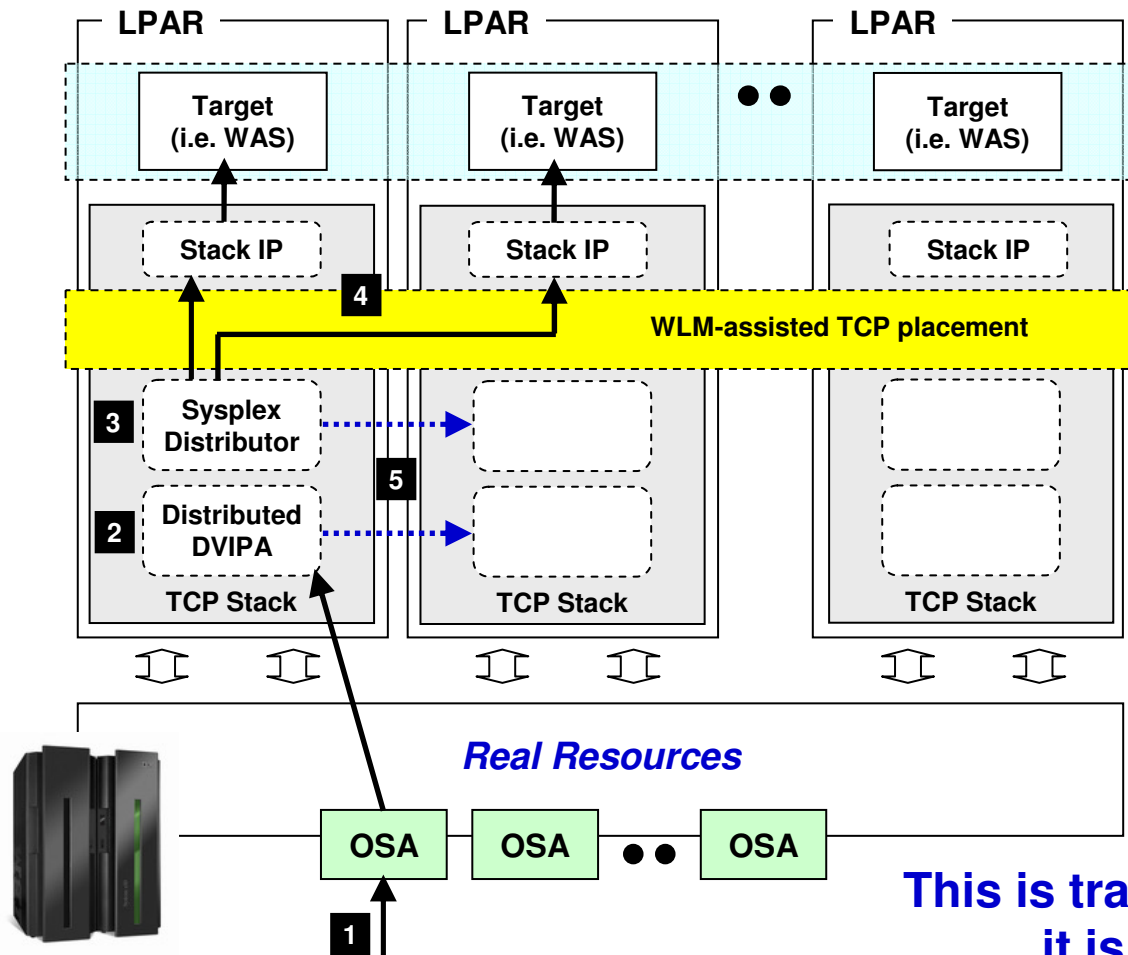
### What’s going on in this picture:

1. Clients in the world point themselves at a “generic” IP host name. Routers resolve that to one of the OSA adapters on the machine.

*Note: there are ways to have redundant OSA adapters for availability*

*Note: it’s not shown on this picture, but WLM can also advise some off-board Cisco routers.*

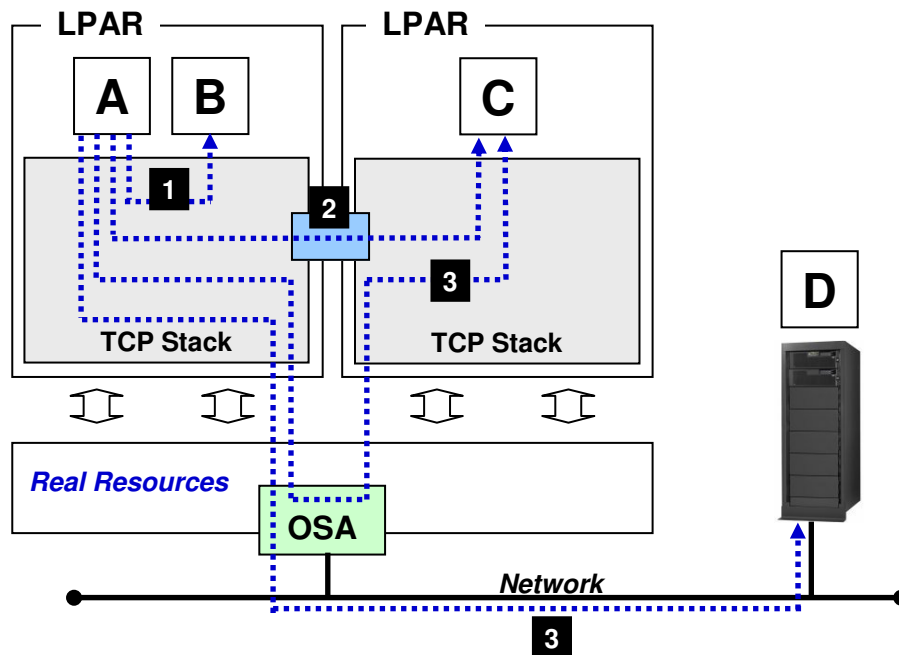
2. Request is mapped to the TCP stack in the Sysplex that’s hosting the Distributed Virtual IP (DVIPA) generic host.
3. Sysplex Distributor function determines which of the potential target LPARs is the best candidate to receive new work at that point in time.
4. TCP connection is made between client and the target
5. In the event of an outage of the hosting LPAR or TCP stack, the DVIPA and Sysplex Distributor functions automatically move to a defined “next in line” stack.



**This is transparent to the application ...  
it is passive in this process**

## Local TCP Optimization

z/OS is smart ... it knows when client and target are on the same TCP and it optimizes the request with minimum code path employed



### 1. Same LPAR

Request resolved within the TCP stack. Never gets to wire. Doesn't even get to the OSA adapter. (Also known as "Fast Local Sockets")

### 2. Different LPAR, HiperSockets

Request flows memory-to-memory via HiperSocket network, which is a virtual network implemented by Hypervisor.

### 3. Different LPAR, *not* HiperSockets

Request flows to OSA, but does not touch the wire. No short loop cables. Request stays in OSA microcode and then up to other LPAR.

### 4. Off System z

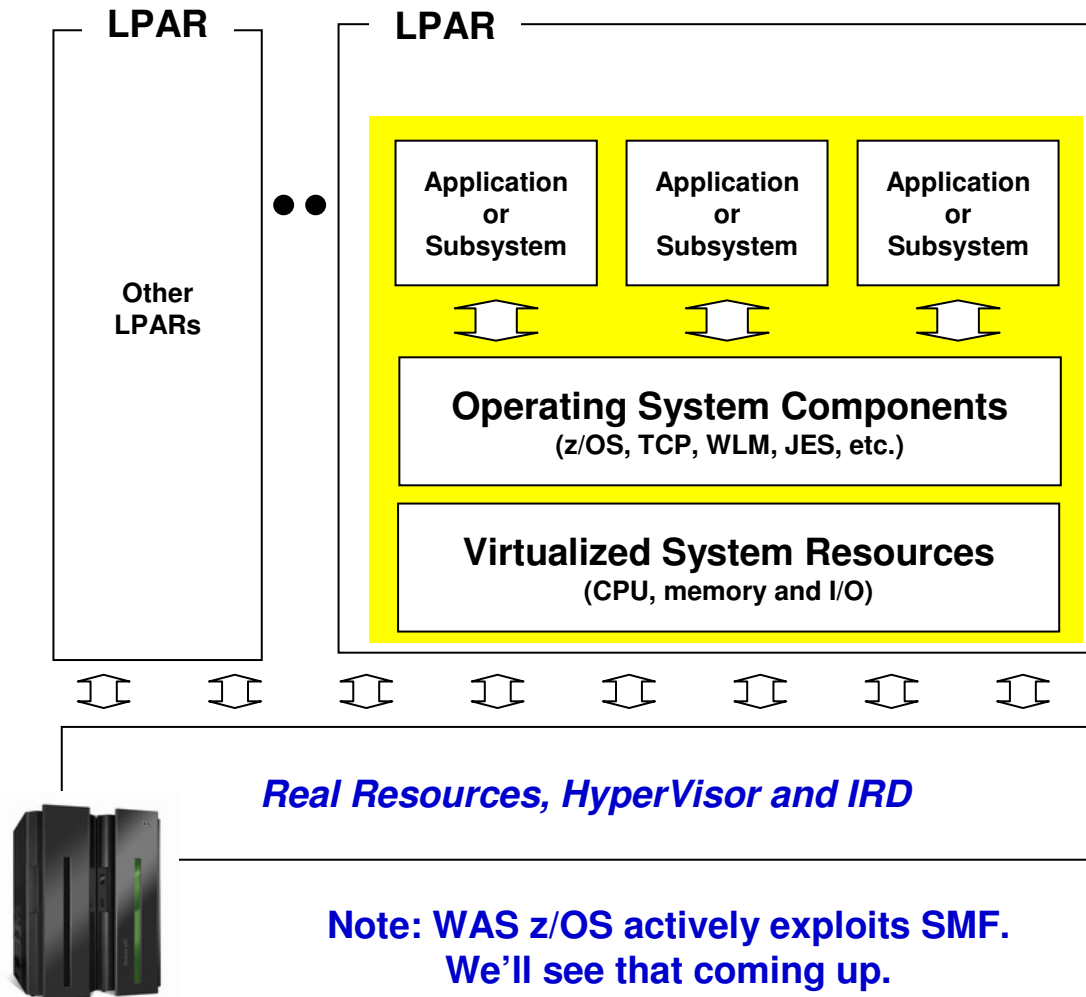
Here System z has no choice but to go to the wire.

This can make a measureable difference

Network latency adds up as workloads scale

## Resource and System Monitoring -- RMF and SMF

In order to manage your server environment effectively and efficiently, you'll need to understand who's using what and when



### SMF

A facility that components may use to write records to a system database.

Those records may then be used to analyze system usage for:

- Capacity planning
- Performance planning
- Accounting and chargeback

### RMF

Another facility that writes SMF to report on key system activities.

Invaluable for planning and investigation of issues

**If the platform is more manageable, then users of the platform derive indirect benefit from that**

## Summary of the Business Value Benefits of Passive Exploitation

### Broad Guiding Principles of Business Value:

- **Available and Reliable**

The System z hardware and the key operating systems (z/OS and z/VM with Linux) are mature and proven.

More reliability and availability is achieved when we get to active exploitation of the platform by WAS z/OS.

- **Manageable**

The platform has a rich set of systems management tools that help maintain the platform and keep it running.

- **Flexible**

For z/OS the shared-resource design is mature and allows for co-location with key isolation attributes. z/VM and Linux provides extensive virtualization capabilities.

- **Affordable**

The value proposition of System z and z/OS is centered around efficient sharing of resources. We cover the cost discussion at the end of the presentation.



# Active Exploitation Benefits

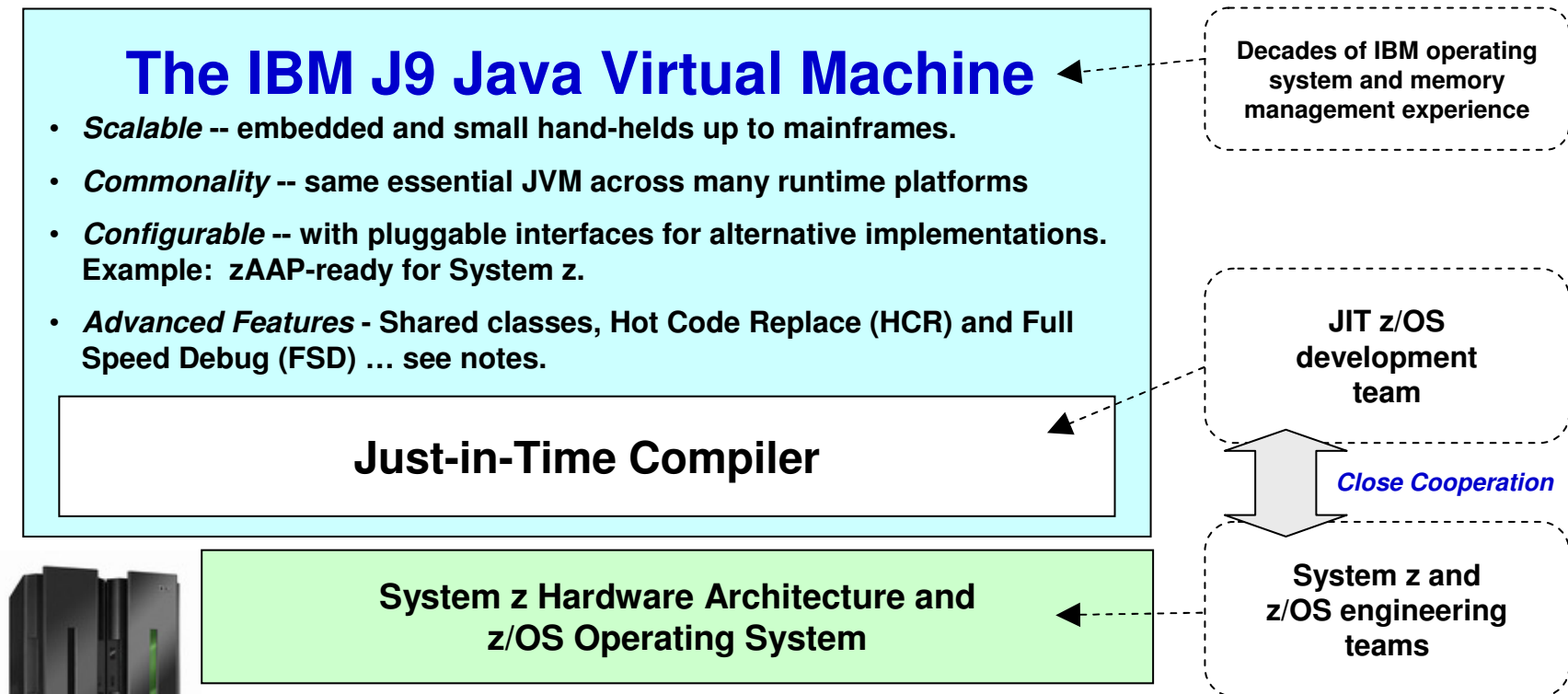
*In two parts: Java SDK for z/OS and WAS for z/OS*

# Active Exploitation, Part 1

*IBM Java SDK for z/OS*

## Bit of History: The IBM J9 JVM

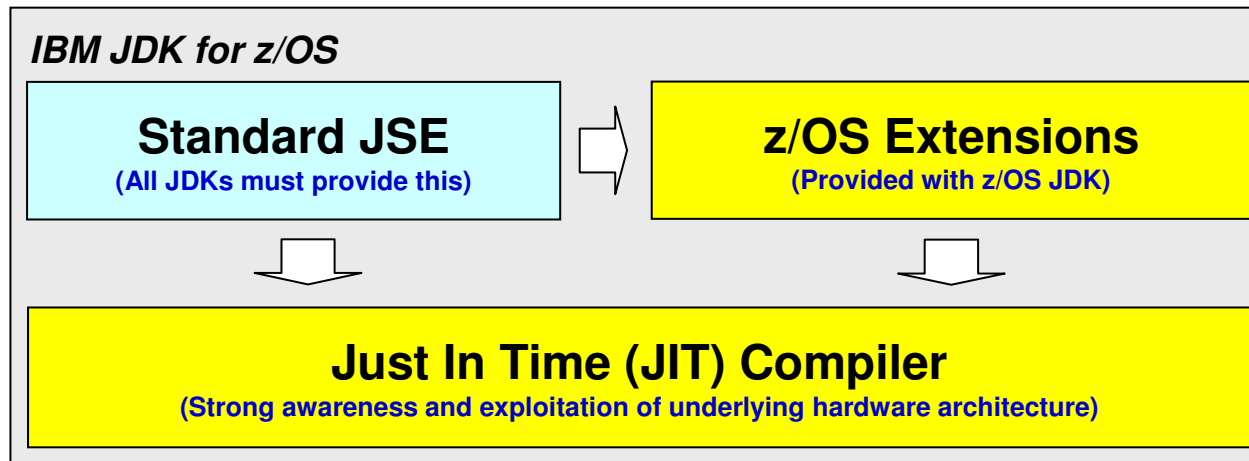
Starting with Java 5, IBM re-wrote the JVM from the ground up. Employed its extensive knowledge in operating system design ... platform-exploiting JITs



**Basic message is one of advanced JVM adhering to standards but exploiting platform specifics where possible**

## High-Level Overview of Things We'll Cover

“Java is Java” but not all JDKs are the same, and not all JDKs are fully aware of and exploit the underlying platform

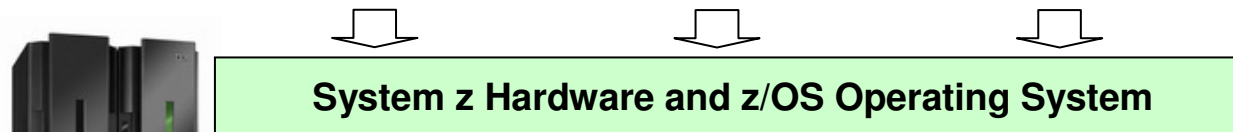


**Functionality *beyond* the standard JSE**

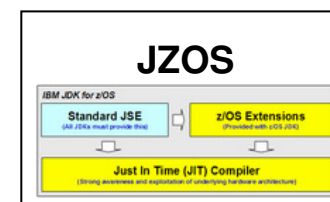
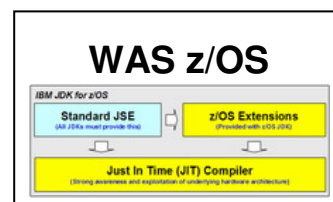
Particularly in the area of security, but a few other things as well as we'll see

**Taking advantage of System z hardware**

Which manifests itself in efficiency and performance



*This JDK used in IBM products on System z*



## Examples of System z Exploitation by JIT

System z hardware has evolved over the years to have some very sophisticated underlying features. JIT in JDK for z/OS is written to exploit them:



### Superscalar Dual-Pipeline

The z890, z990, z9 and z10 are superscalar dual pipeline designs. It permits the dispatching of three instructions per cycle. The JIT understand this and generates code that exploits this where possible.

### Register Allocation

13 of the 16 64-bit processor registers are available and the JIT makes extensive and efficient use of the registers to enhance throughput and performance.

### Compare-and-Swap

Compare-and-Swap is a feature that allows programs to use comparison of register values to provide a form of code access lock management. It's very efficient. The JIT uses this for Java synchronization.

### Private Linkage

Program linkage at the lowest possible level, allowing one method to call another with a minimum of overhead. Includes direct JNI dispatch from JIT.

### CISC Exploitation

System z hardware is a “Complex Instruction Set Computer” (CISC) design, with many very elaborate instruction features. The JIT knows about these and exploits them; for example, Java loop structures reduced to a very small set of CISC instructions.

### 64-Bit Instructions and 31-Bit JVMs

The System z hardware is a 64-bit design. If the JVM is running in 31-bit mode the JIT is capable of exploiting the 64-bit hardware for long arithmetic operations.

**The JIT is very much aware of the System z hardware features and exploits them directly for greater throughput and efficiency**

## z/OS-Specific Extensions to the JDK

In addition to the standard-compliant Java there are extensions to provide exploitations of System z and z/OS functions:

[ibm.com/servers/eserver/zseries/software/java/products/j6pcont64.html](http://ibm.com/servers/eserver/zseries/software/java/products/j6pcont64.html)

The IBM 64-bit SDK for z/OS, V6 provides a full function SDK compliant with the [SDK 6 APIs](#).

Documentation is available for content that is additional to the base.

- Security functions:
  - Java Cryptography Extension (IBMJCE)
  - Java Cryptography Extension in Java 2 Platform Standard Edition, Hardware Cryptography (IBMJCECCA)
  - Java Secure Sockets Extension (IBMJSSSE)
  - Java Certification Path (CertPath)
  - Java Authentication and Authorization Service (JAAS)
  - SAF interfaces
  - Java Generic Security Services (JGSS)
  - Java PKCS#11 Implementation Provider (IBMPKCS11Impl)
- RMI-IIOP
- Java Record I/O (JRIO)
- JZOS - Java Batch Launcher and Toolkit

Extensions to the JSE security standards to provide access to System z and z/OS facilities such as SAF and the Crypto Hardware

Access to VSAM files, sequential files, PDS directories and the system catalog

More on JZOS ...

All content above is shipped with the z/OS SDK product and is zAAP eligible.

**Everything the standard JSE calls for, plus additional function that exploits the platform for added benefit to you**

## JZOS and Even More Access to z/OS Functionality

**JZOS is acquired technology that allows Java to be run from batch or a started task with a more “z-Like” set of features and behavior**

```
//JZOSBAT JOB (999,XXX), 'JAVA JZOS', CLASS=A, MSGLEVEL=(1, 1)
// MSGCLASS=X, REGION=0M, NOTIFY=&SYSUID
//JAVAJVM EXEC PGM=JZOSVM14,
// PARM='HelloWorld'
//STEPLIB DD DSN=JZOS.LIBRARY, DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//STDOUT DD SYSOUT=*
//STDERR DD SYSOUT=*
//STDENV DD *
. /etc/profile
:
//
```

**Provides an environment where all sorts of System z and z/OS unique Java class libraries can be built and provided:**

Packages	
<a href="#">com.ibm.jzos</a>	Provides the primary classes for the JZOS toolkit.
<a href="#">com.ibm.jzos.fields</a>	Provides the field converter classes for mapping byte array fields into Java data types.
<a href="#">com.ibm.jzos.sample</a>	Provides sample classes for using the JZOS Toolkit API.
<a href="#">com.ibm.jzos.sample.dfsort</a>	Provides sample classes for using the JZOS toolkit DfSort class.
<a href="#">com.ibm.jzos.sample.fields</a>	Provides sample record mapping classes for using the JZOS field converter classes.

**Launch a batch job ... issue a WTO ... DfSort ... other stuff**

### Benefits over BPXBATCH

Flexible configuration of environment variables	Yes
Route output directory to SYSOUT datasets	Yes
Control output encoding separately from default JVM encoding	Yes
Condition-code passing between Java and non-Java steps	Yes
Use MVS datasets and DD statements	Yes
JVM runs in same address space	Yes
Communication with MVS console	Yes

**JVM launcher tailored to z/OS environment**



## 64 Bit Performance

64-bit JVMs provide relief from heap crowding, but initially came with a performance cost. Two new features match the 31-bit performance profile:


[ibm.com/partnerworld/wps/whitepaper/systemz/java\\_websphere/performance](http://ibm.com/partnerworld/wps/whitepaper/systemz/java_websphere/performance)

PartnerWorld > Products > Systems, servers, and storage > Technical >

### Match 31-bit WebSphere Application Server performance with new features in 64-bit Java on System z

by Kishor Patil, Marcel Mitran, Jim Cunningham

Last updated: 2009-05-20

 [Download](#) the white paper (285 KB)



### Compressed References

Function added to the z/OS Real Storage Manager (RSM) with APAR OA26294 provides a direct assembler interface which allow memory allocations in the 2GB ( $2^{31}$ ) to 32GB ( $2^{35}$ ) virtual address range. The JVM uses this API to allocate the heap in this virtual address range.

### Large Page Support

System z10 processors introduced support for 1MB pages (APAR OA20902 and OA25485 for z/OS 1.9). The 64-bit JVM can achieve performance gains by using large pages, which results in fewer Translation Look-aside Buffer (TLB) entries needed. Fewer TLBs needed for the data footprint of the JVM means more TLBs are available for the executable code. Fewer TLB misses in instruction fetches occurs, which enhances performance.

**System z/10 with these two features allows a 64-bit JVM to operate with a larger heap and match the performance seen with the smaller heap 31-bit JVMs.**

# Active Exploitation, Part 2

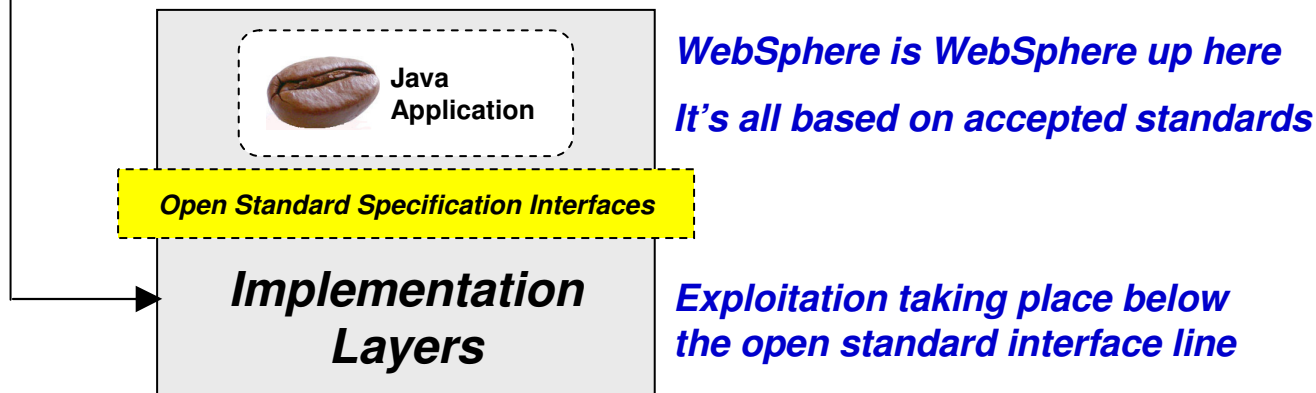
*WebSphere Application Server for z/OS*

## Eight Things We'll Explore in This Section

1. Exploitation of JES and common z/OS facilities
2. Exploitation of zAAP specialty engines
3. Exploitation of **WLM**
4. Exploitation of RRS
5. Exploitation of SAF and Crypto
6. Exploitation of SMF
7. Exploitation of z/OS exclusive Cross Memory Communications

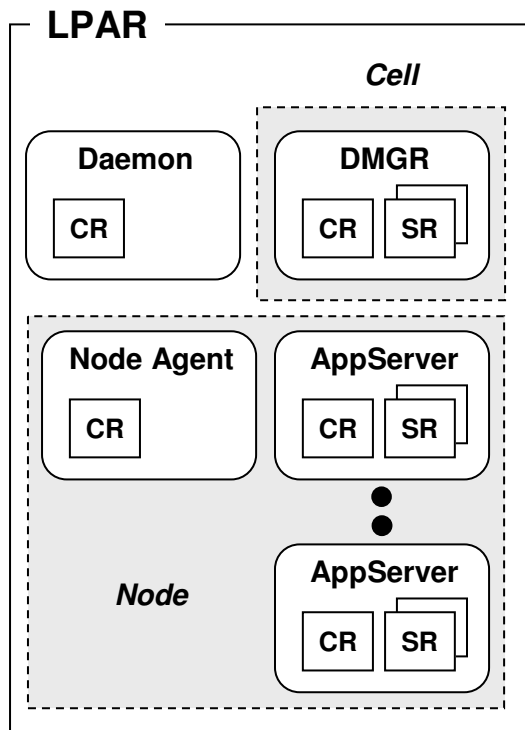
We'll focus heavily on WLM exploitation because that's at the heart of the "Why WAS z/OS" question

These are all z/OS value attributes



## Exploitation of JES and Common z/OS Facilities

And that means that existing z/OS system programmers will be comfortable with the essential operations of WAS z/OS ... it maps to their present skills



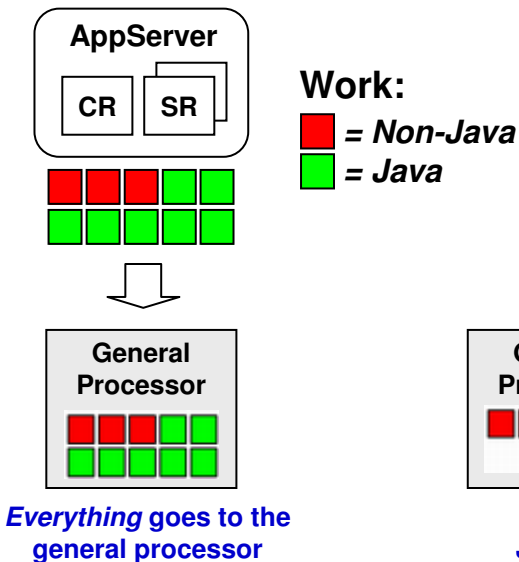
- WAS z/OS runtime implemented as a series of started tasks
- Standard JCL and START commands employed
- JCL START procedures maintained in PROCLIB
- Output written by default to JES
- JES manages output and storage
- Started tasks and address spaces displayable like any other
- Started and stopped like any other
- Able to use MODIFY commands for dynamic operations
- Configuration held in HFS or ZFS file systems
- **Allows system automation tasks to control operations**

**This is all standard stuff. The key is that WAS z/OS was implemented to be compatible with existing z/OS skills, and to take advantage of existing z/OS facilities. WAS z/OS is *not* merely UNIX processes running in USS.**

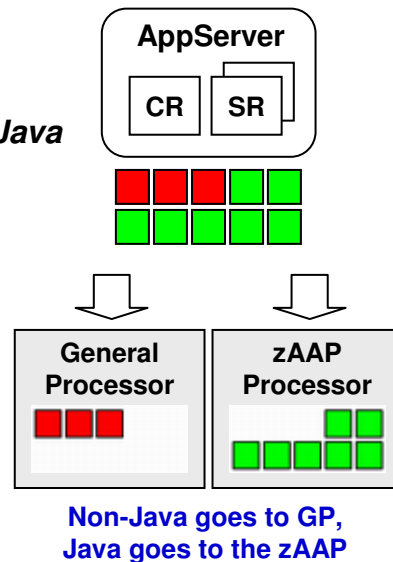
## Exploitation of zAAP Specialty Engines

zAAP engines are Java offload engines. They enhance the financial picture of the z/OS platform, and they free up GP for other key subsystem processing

### Before zAAPs



### With zAAPs



### Keys to understanding value of zAAPs:

- zAAP processors have a considerably lower acquisition cost compared to GPs
- Offloading Java to zAAP frequently allows growing non-Java work to live within existing GPs, thus avoiding capital acquisition
- Monthly license charges based on capacity of the system can be influenced by the presence of zAAPs, which do not count towards charges

There are many technical details left unsaid here with respect to how they're configured, the rules for dispatching, when Java might go to GP, etc. Objective here was key points, not details.

This is really a function of the Java SDK and the dispatcher of z/OS.

The zAAP-enabled Java SDK is packaged with WAS z/OS, so WAS automatically takes advantage of zAAPs if they're present and configured

## Exploitation of WLM

Many view WLM exploitation as the heart of the platform exploitation model for WAS z/OS. There are four main elements of this exploitation ...

### Intelligent Dynamic Capacity Expansion

The ability to increase the number of JVM instances based on WLM goals and configuration settings.

This is the “Controller / Servant” structure you may have heard about

### Intelligent Workload Flow Control

An element of the Controller/Servant structure. Inbound work is queued and held, waiting for a thread to select it, based on importance and arrival. It’s a pull model rather than a push. Applications in JVMs take only what they can handle.

### Intelligent Management of Mixed Work in Server

Multiple servants allows differently classified work to be placed in different servant regions. This allows WAS/WLM to understand what kind of work is in each and to manage system resources accordingly.

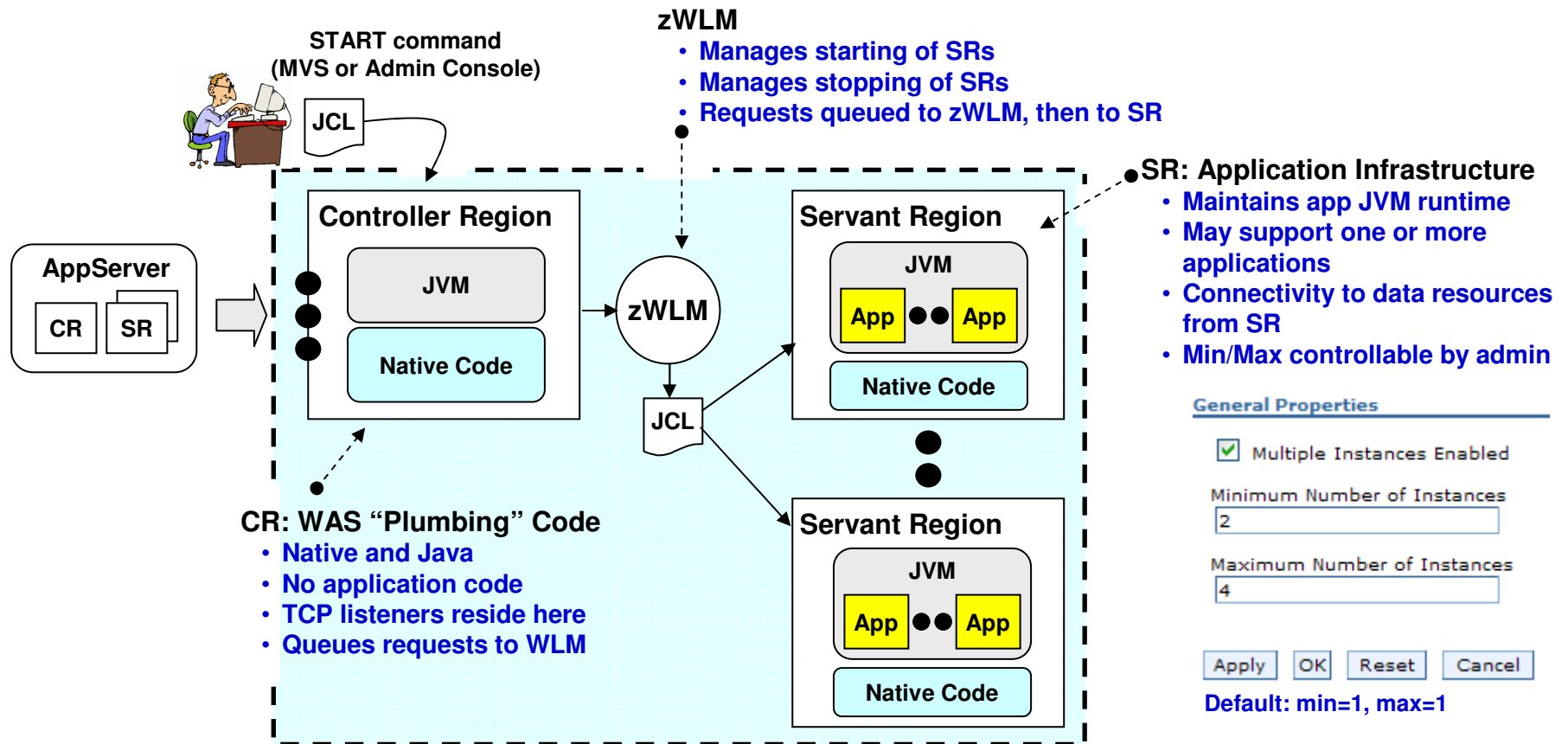
### Intelligent Workload Routing Advice

WAS z/OS using WLM to determine where best to route certain kinds of work

**The key is the controller / servant architecture ...**

## The Controller / Servant Architecture

This is a unique architectural element to the WAS z/OS design. No other platform has this design because no other platform has WLM\*\*:



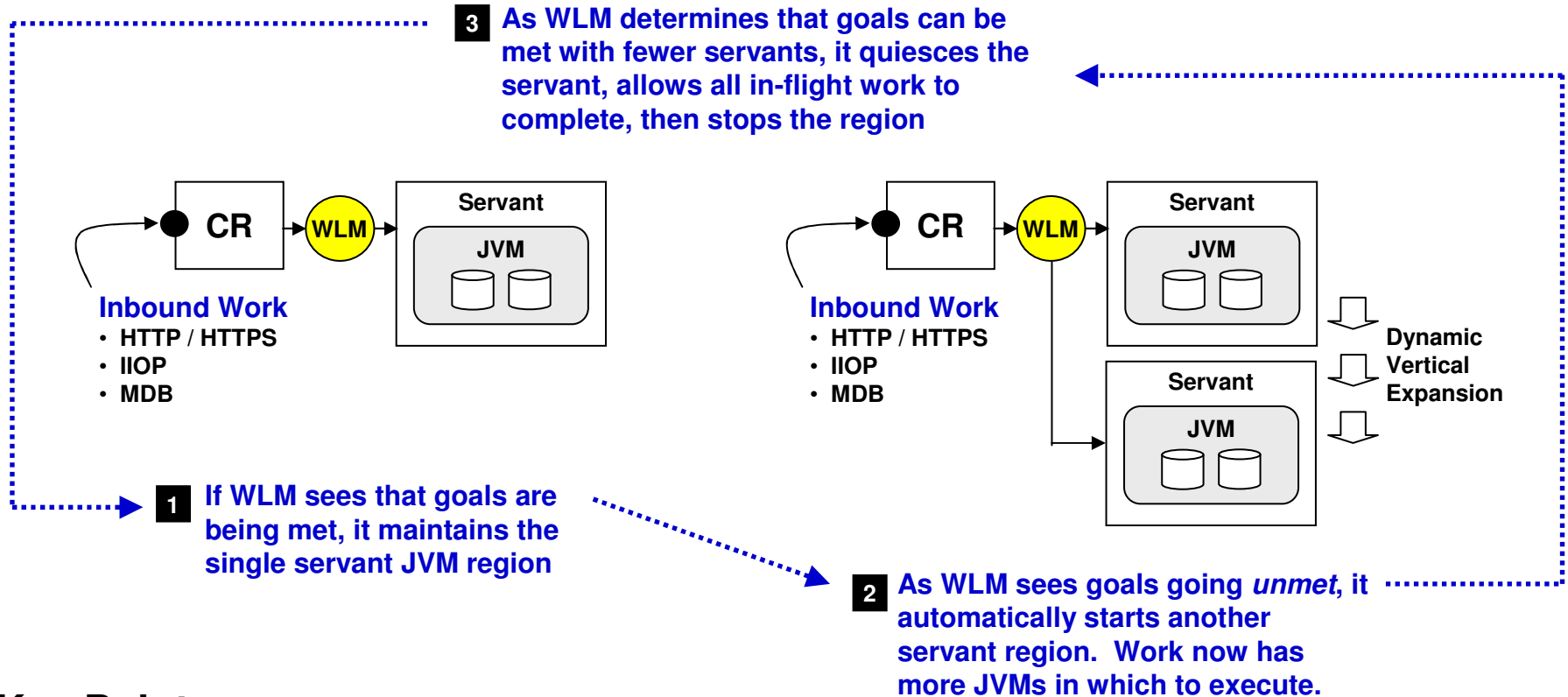
Let's now explore how this is accomplished ...

\*\* WebSphere on distributed uses the phrase "Workload Management" but it's not the same as zWLM



# Intelligent Dynamic Capacity Expansion

This is the “vertical scaling” capability of the multi-Servant structure. If allowed, WLM will start additional servant regions if it sees unmet goals:

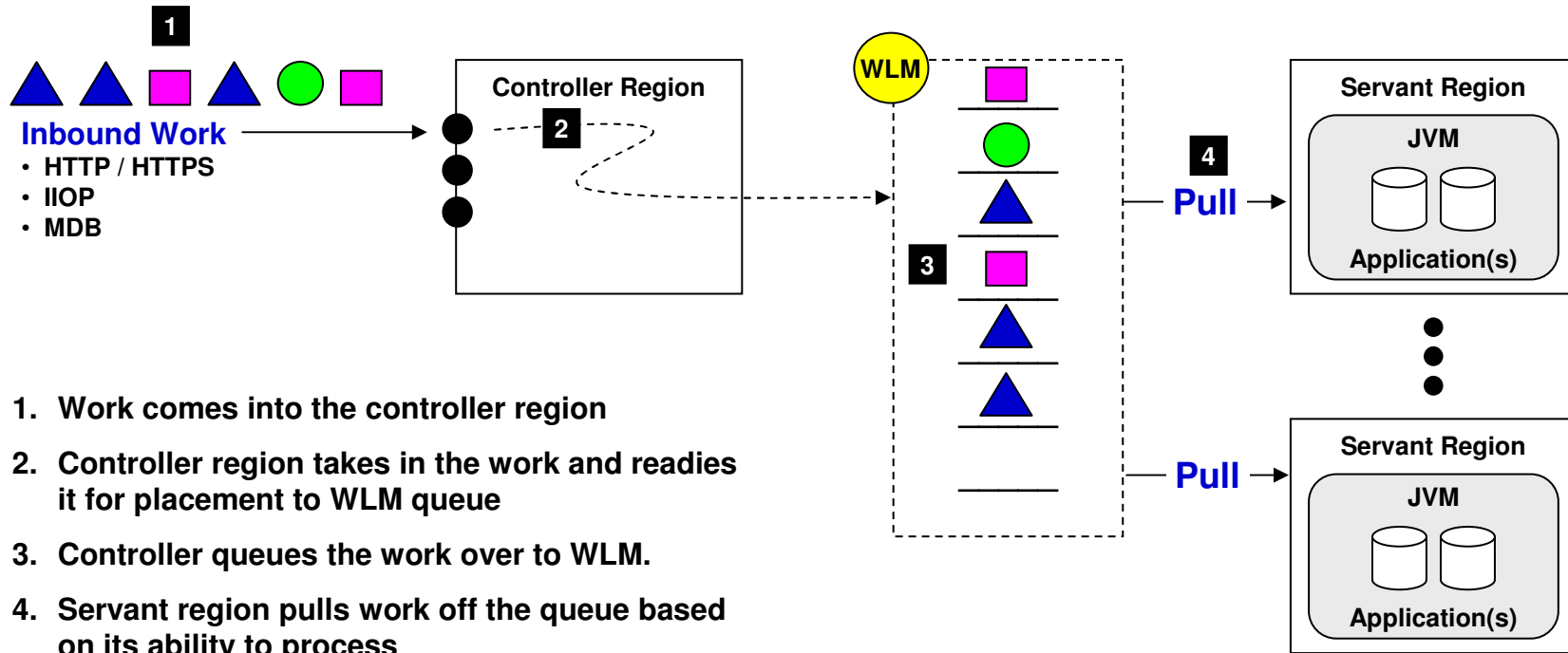


## Key Points:

- The minimum and maximum number of servants is configurable. Default: Min=1, Max=1
- We see distributed WAS users trying to do something similar by configuring a “vertical cluster” to provide duplicate JVMs on a server box. Not quite the same -- no WLM assist of that

# Intelligent Workload Flow Control

This is the WLM queuing mechanism that exists between the CR and the SR. It creates a “pull” model that prevents overwhelming an application JVM:



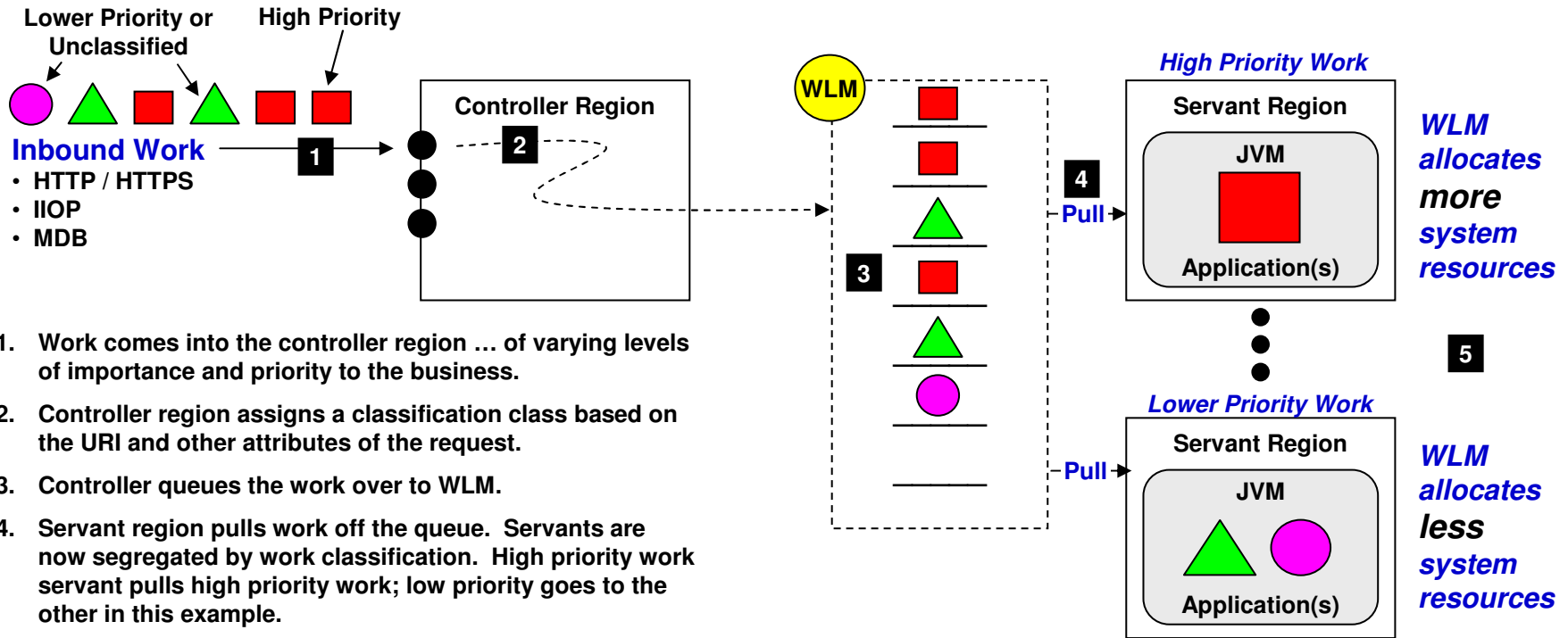
1. Work comes into the controller region
2. Controller region takes in the work and readies it for placement to WLM queue
3. Controller queues the work over to WLM.
4. Servant region pulls work off the queue based on its ability to process

## Servant can't be overwhelmed

Servant only takes what it can. Controller will take in and queue up what can't be handled immediately.

# Intelligent Management of Mixed Work in Server

This involves inbound work being given a “Transaction Classification.” With that, the CR can direct work to servants and WLM can manage:



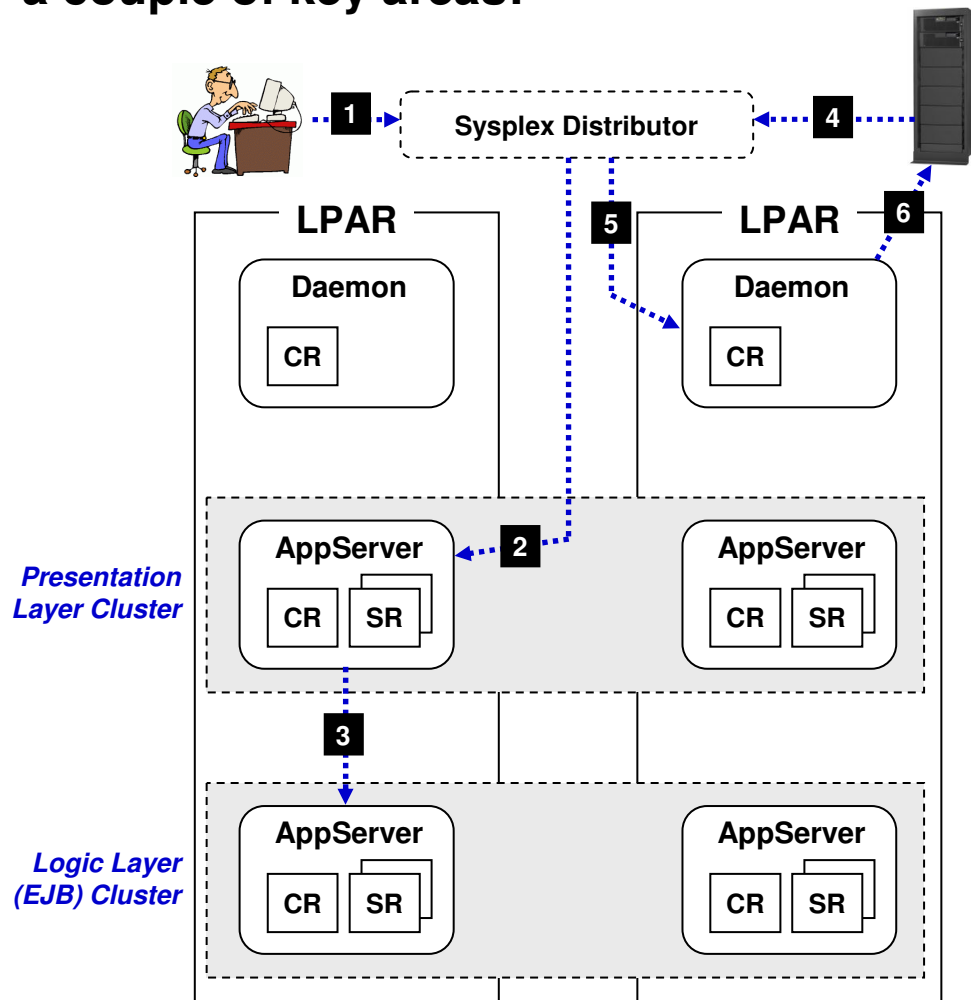
1. Work comes into the controller region ... of varying levels of importance and priority to the business.
2. Controller region assigns a classification class based on the URI and other attributes of the request.
3. Controller queues the work over to WLM.
4. Servant region pulls work off the queue. Servants are now segregated by work classification. High priority work servant pulls high priority work; low priority goes to the other in this example.
5. With work segregated by servant region, WLM can now manage the system resources given to each servant. High priority work gets more, lower priority less, all according to defined WLM goals.

## Sophisticated Work Prioritization

On other platforms this can only be done by allocating work to separate servers. No WLM there to manage at this level.

# Intelligent Workload Routing Advice

WAS z/OS relies on WLM to make routing decisions. We see this exercised in a couple of key areas:



## Inbound HTTP or IIOp Work

1. Work comes in over network aimed at DVIPA and Sysplex Distributor
2. WLM advises Sysplex Distributor of presentation cluster member to place TCP request to
3. For servlet-to-EJB flow (IIOp) WLM advises server, which then places the IIOp request to one of the members in the EJB cluster

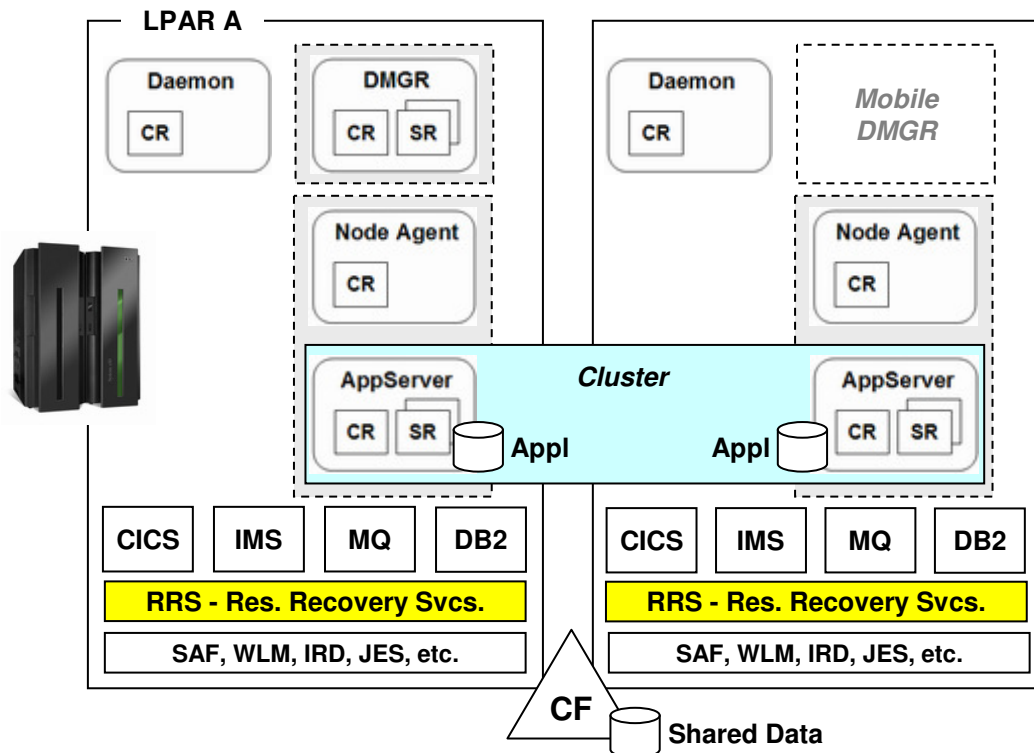
## IOR Resolution

4. External client seeks to use EJB deployed in WAS. It addresses itself to DVIPA/Sysplex Distributor for bootstrap (not shown on picture), then gets DVIPA host for Daemon location service.
5. External client then comes back to DVIPA/SD for Daemon location service. WLM advises Sysplex Distributor for best Daemon to place request.
6. Daemon consults WLM for object location best able to service external client at that time. External client provided with host:port for EJB in the cluster.

The point here is that WLM plays a key role in the routing decisions made by clients and WAS itself for real-time routing

## Exploitation of Resource Recovery Services (RRS)

Two-phase commit processing involves coordination of participants to make sure all are ready to commit. RRS plays that role in Parallel Sysplex:

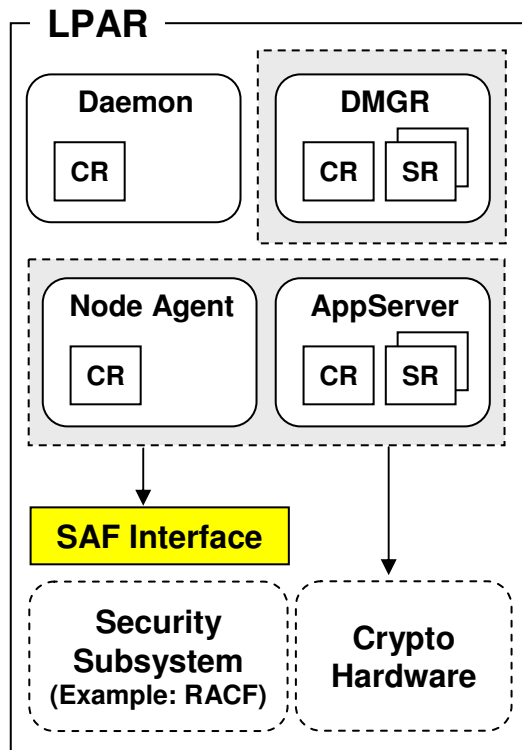


- We'll see this picture later when we discuss high availability
- WebSphere Application Server is a transaction manager ... it is able to initiate a transaction and have other resource managers (DB2, CICS, IMS) participate in the unit of work
- For two phase commit processing, *someone* has to play the role of syncpoint coordinator
- On z/OS and Parallel Sysplex that someone is RRS, which uses Coupling Facility data structures and patented recovery algorithms to provide very efficient failed transaction recovery
- WAS z/OS registers with RRS, as do resource managers. RRS handles the two-phase commit coordination

Another case of “below the specification line” exploitation of existing z/OS and Parallel Sysplex technology to perform a task in an optimized manner for the platform

## Exploitation of SAF and Crypto

**SAF is a security interface; Crypto is a hardware-assist processor for encryption and key storage on the System z and z/OS platform**



### SAF Security Subsystem

- Sysplex-wide integrated security repository
- Single location for security artifacts rather than scattered model
- IDs, groups, keyrings, certificates, EJB role enforcement
- Local access ... unlike LDAP, do not need to traverse network
- Extremely robust security model

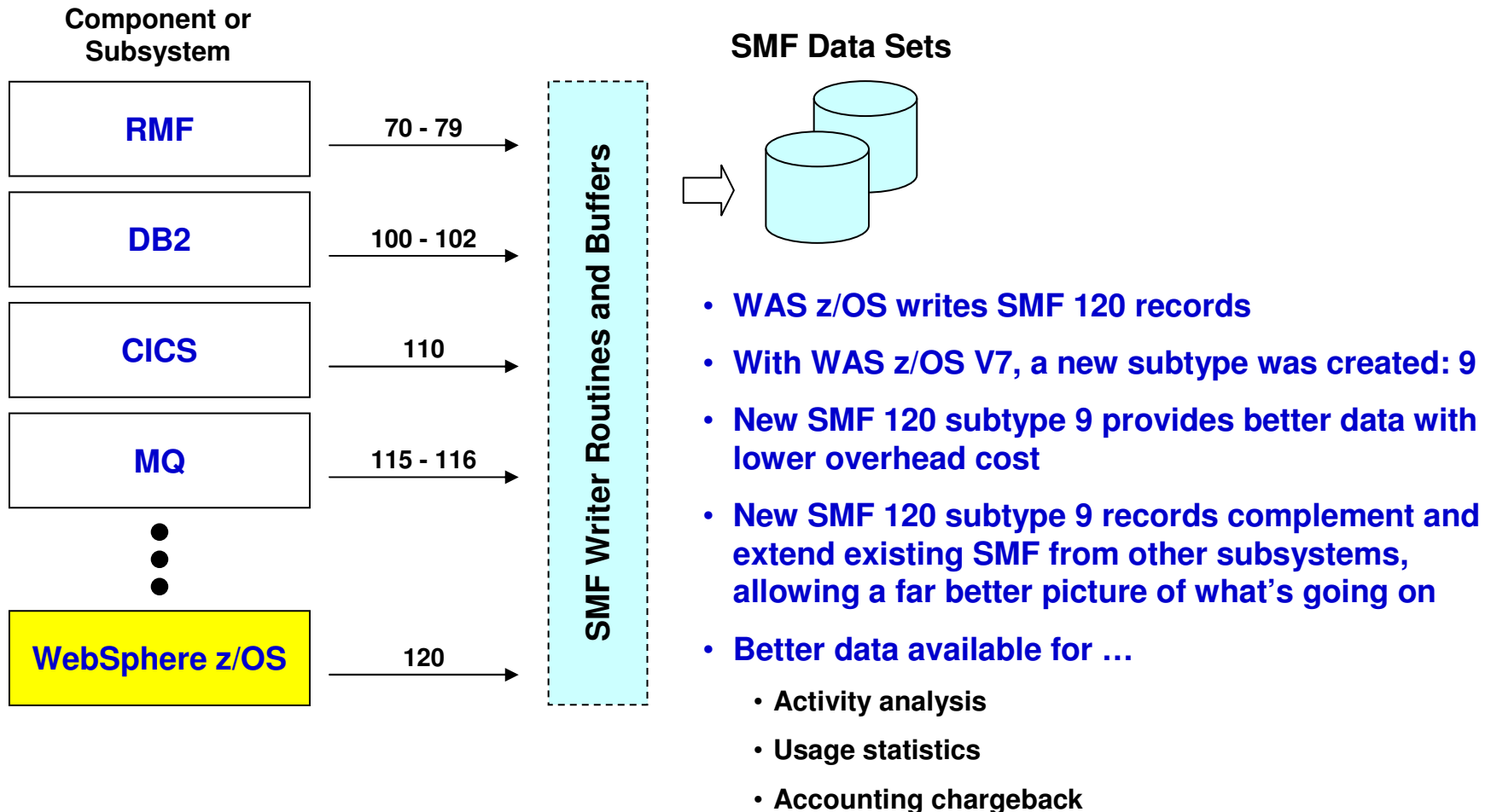
### Crypto Hardware

- Hardware-assisted cryptographic encryption and de-encryption
- Extremely secure private key store management

**Properly configured, z/OS provides an extremely secure environment ... many say the most secure available**

## Exploitation of SMF

SMF is an activity recording facility of z/OS that allows subsystems to record key activity for analysis, management and accounting chargeback



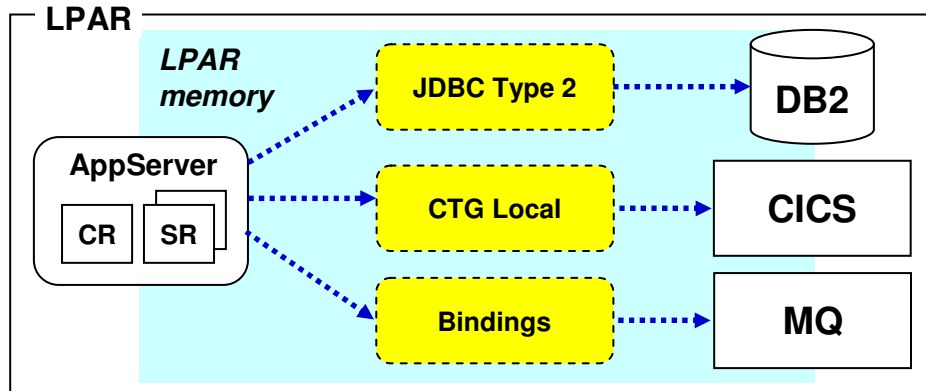
<http://www.ibm.com/support/techdocs/atmsastr.nsf/WebIndex/WP101342>



## Exploitation of Cross-Memory Communications

Any time client and target are in the same LPAR, there's an opportunity for cross-memory exploitation. Let's look at a few examples:

### Data Access

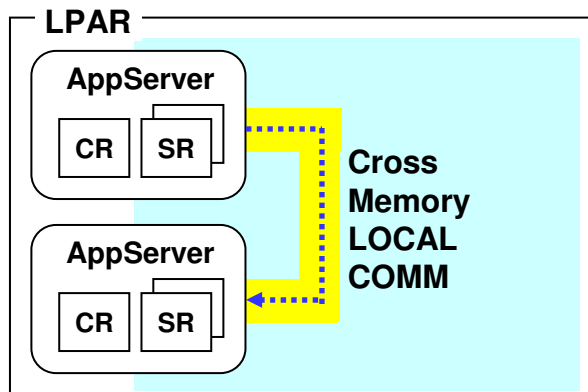


#### Benefits:

- Cross memory speed
- Security ID propagation (no alias)
- Exploitation of RRS
- Avoid serialization of parameters
- Avoids SSL overhead
- Single thread of execution

### LOCAL COMM

Used for IIOP flows between servers on the same LPAR.



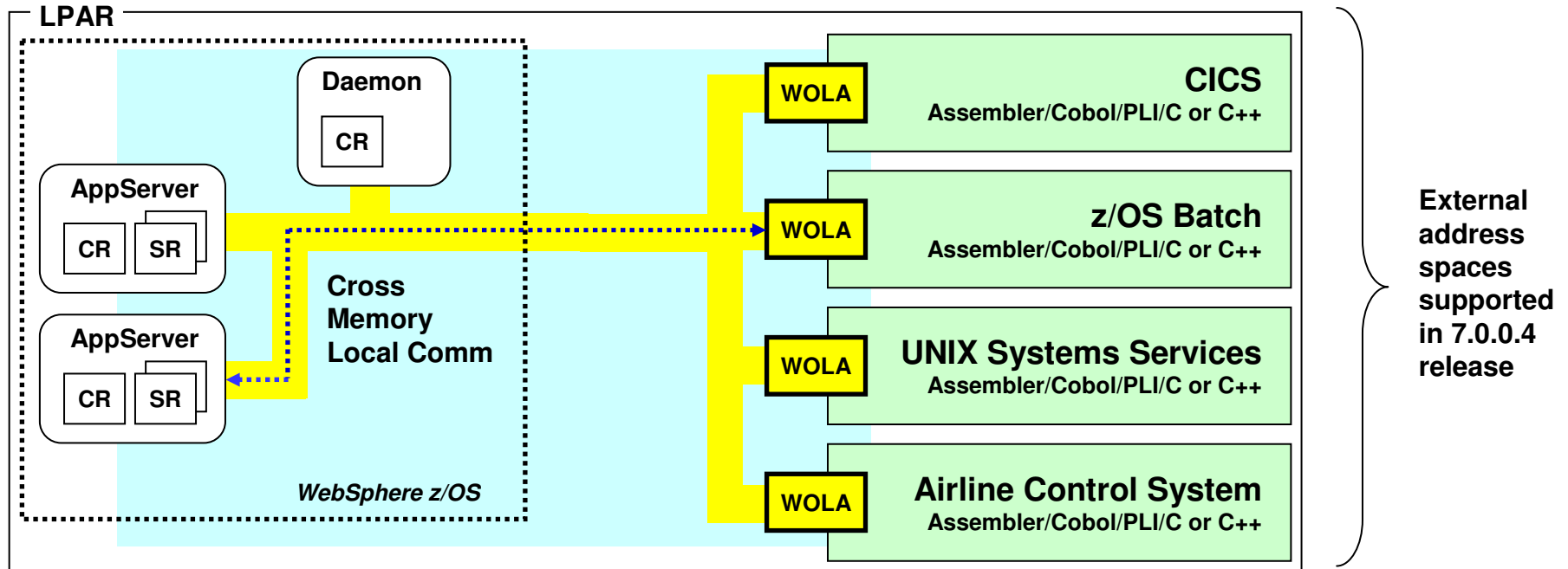
#### Benefits:

- Avoids IP stack entirely
- Avoids SSL overhead
- Very fast, very secure

**Extension to  
Local Comm: new  
Optimized Local  
Adapters ...**

## Cross-Memory: New Optimized Local Adapters (WOLA)

New in V7.0.0.4, this new function allows external address spaces to participate in a cell's Local Comm communications.



### Benefits:

- Based on Local Comm (z/OS exclusive)
- **Bi-directional** ... WAS outbound or inbound to WAS (WOLA exclusive)
- CICS Security and Transaction propagation (some restrictions apply)
- Faster than other local solutions

WP101490 on  
[ibm.com/support/techdocs](http://ibm.com/support/techdocs)  
 for more

## Summary of the Business Value Benefits of Active Exploitation

### Broad Guiding Principles of Business Value:

- **Available and Reliable**

The active exploitation elements take direct advantage of Parallel Sysplex and the shared data clustering capabilities of the technology. Properly configured, no other platform offers the same degree of availability and reliability.

- **Manageable**

The active exploitation of system components such as WLM, SAF, RRS and SMF allow for management using proven and mature technologies.

*Key point: WAS z/OS uses and exploits well known and understood system services. There's no need to develop new tools for operations, capacity planning, performance management, etc. This is building on existing knowledge.*

- **Flexible**

The active exploitation of z/OS interfaces such as MODIFY provide dynamic operations at the OS level against the operating WAS servers.

- **Affordable**

One element of contribution to the expense question is the zAAP specialty engine for Java offload.

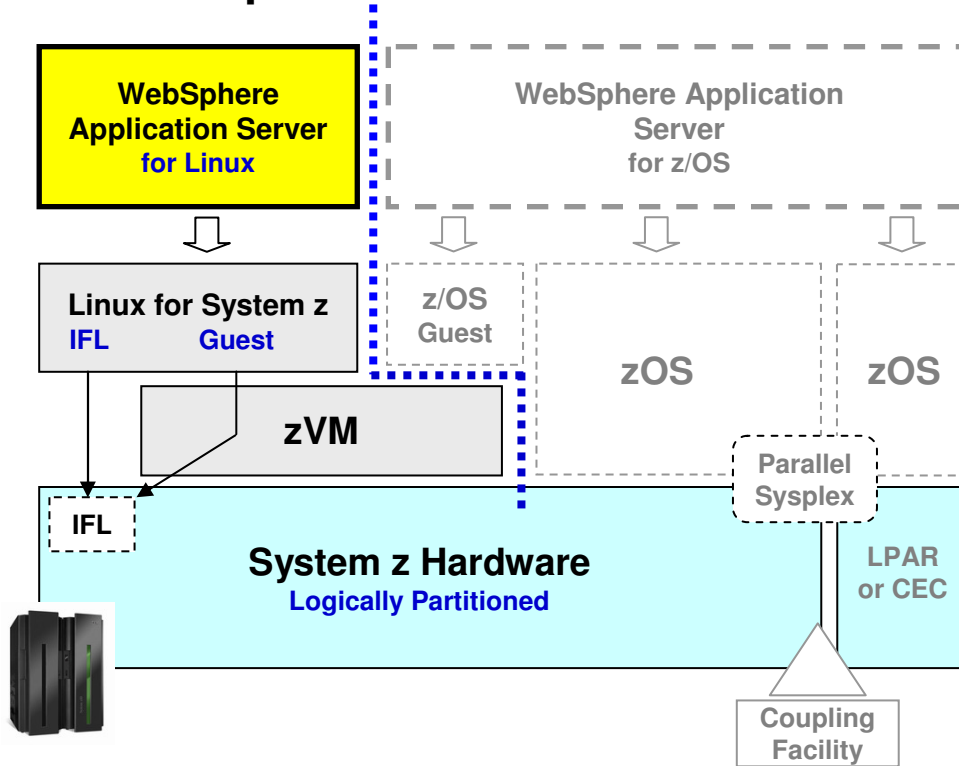
Again, the value proposition of System z and z/OS is centered around efficient sharing of resources. We cover the cost discussion at end of presentation.

# WAS and Linux for System z

*To understand the key differences and position the two System z offerings*

## Our Earlier Picture

We saw this picture earlier ...



All the WAS z/OS pieces of that picture have been grayed-out

So too has the Parallel Sysplex and Coupling Facility

WAS for Linux on System z can not *directly* participate in Parallel Sysplex or the attributes of the z/OS operating system.

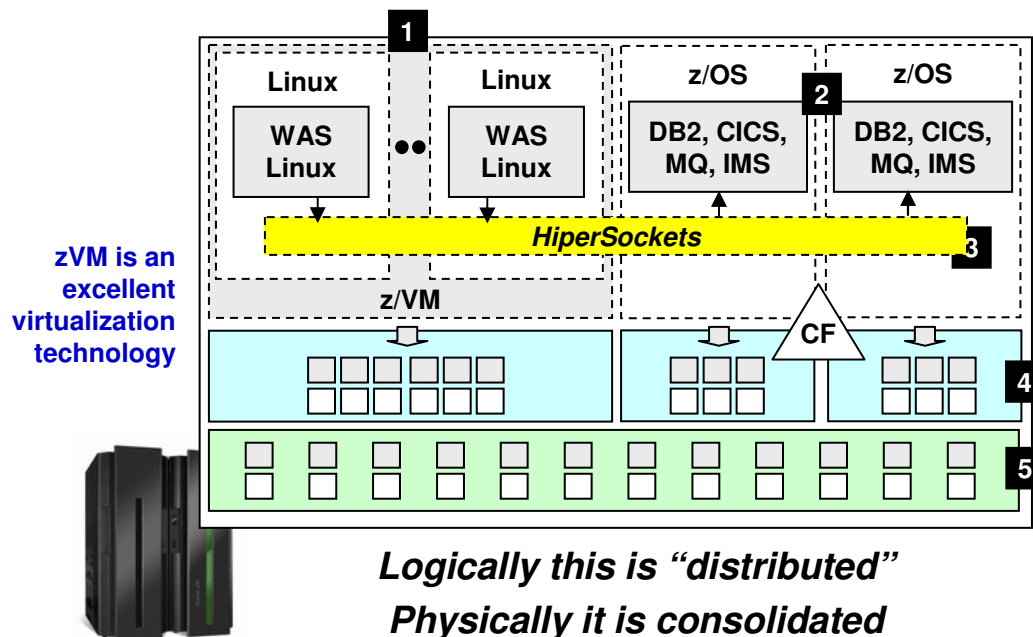
*Active* exploitation of hardware virtualization through zVM is realized

*Passive* benefit from the System z hardware is realized.

*Indirect* participate in Parallel Sysplex is possible ...

## Common Use of WAS Linux System z and Parallel Sysplex

Many access data on z/OS Parallel Sysplex from Linux LPARs in the same CEC, using HiperSockets for TCP access:



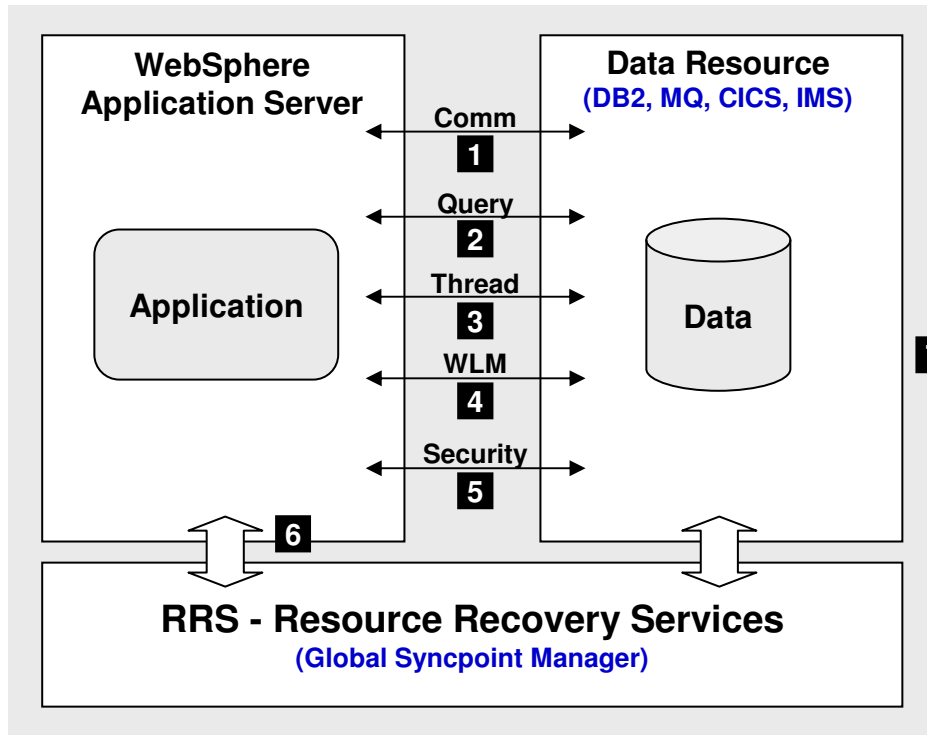
1. **Linux LPARs running WAS**  
Exploiting virtualization of z/VM to realize server consolidation to single HW footprint
2. **Data subsystems in Parallel Sysplex**  
Exploiting data sharing
3. **HiperSockets**  
Optimized cross-LPAR virtual network
4. **Virtualized resources**  
Exploiting LPAR technology of System z
5. **Real resources**  
Efficient sharing of real resources through HiperVisor allocation of real to virtual

This is primarily used for server consolidation. Advantages over server farms include reduced power, cooling, square footage, administration and potentially software

Very good, but it does have drawbacks compared to co-location on z/OS ...

## Linux vs. Co-Location on z/OS

This gets to the “Value of Co-Location” as outlined in WP101476 paper on [ibm.com/support/techdocs](http://ibm.com/support/techdocs):



Important point here ... solution architectures that span multiple operating systems environments implies different monitoring and management capabilities. Correlating information between those different tools can be challenging. Co-location helps reduce that complexity by bringing it all under a single operating system environment.

1. **Cross-memory data transfer**  
Better than even HiperSockets
2. **Avoid data and parameter serialization**  
Since not passing across network, do not need to serialize. Avoids SSL as well.
3. **Single thread of execution**  
Avoid switching threads, which means even greater efficiency
4. **Manage to a single WLM goal**  
Easier goal definition and management
5. **Passing security context**  
More options for security identity propagation: servant ID, client ID, application role vs. alias for remote T4
6. **Sysplex-wide RRS**  
Extremely efficient transaction recovery processing
7. **Reduced complexity**  
Single OS, more focused problem determination



## Summary of the Business Value Benefits of Linux on System z

### Broad Guiding Principles of Business Value:

- **Available and Reliable**

z/VM and IBM's Linux kernel are proven and reliable. No direct exploitation of Parallel Sysplex, but access to data elements on z/OS provide for a degree of that at the data layer.

- **Manageable**

Linux is Linux, but the z/VM tools at the virtualization layer are extensive and mature.

- **Flexible**

This is one of the key value statements of Linux on System z. The virtualization capabilities translate directly into rapid flexibility.

Software vendors often limit their support to a few operating systems, Linux often being one. Linux on System z allows you to take advantage of the platform and maintain ISV support.

- **Affordable**

Consolidation of servers from distributed to virtualized Linux serves on z/VM provides reduced environmental costs, potentially reduced administrative costs, and potentially reduced software licensing costs.

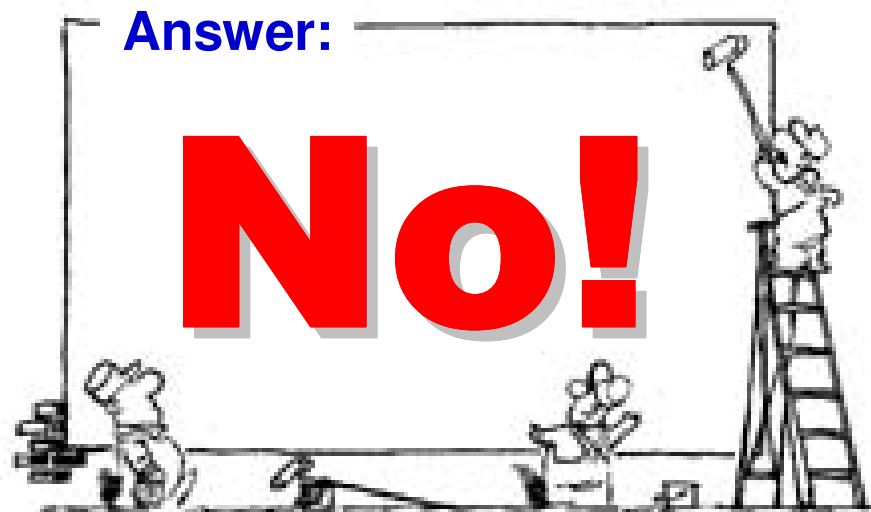
Every situation is different. An analysis of cost savings is recommended.

# Application Design Considerations

*Exploring what makes a Java application “z ready”*

## Is Java on System z Different?

Is there anything special that needs to be considered when designing and writing Java code to run on System z or z/OS?



Clear enough? 😊

## Java is Java

The point of an open standard application platform is to eliminate platform dependencies

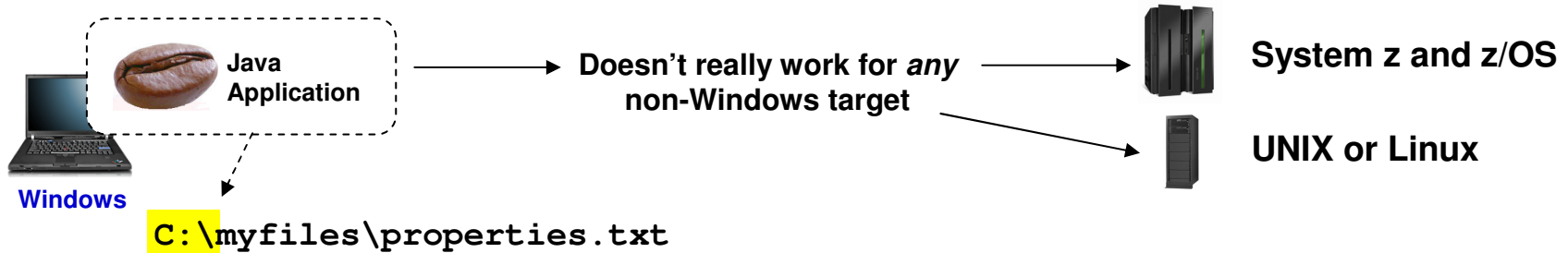
### Two points:

1. That wasn't always the case ... earlier we had ASCII / EBCDIC issues. No more.
2. There are poor coding practices that make bringing applications to z/OS problematic ...

## Compatibility -- Two Common Problems

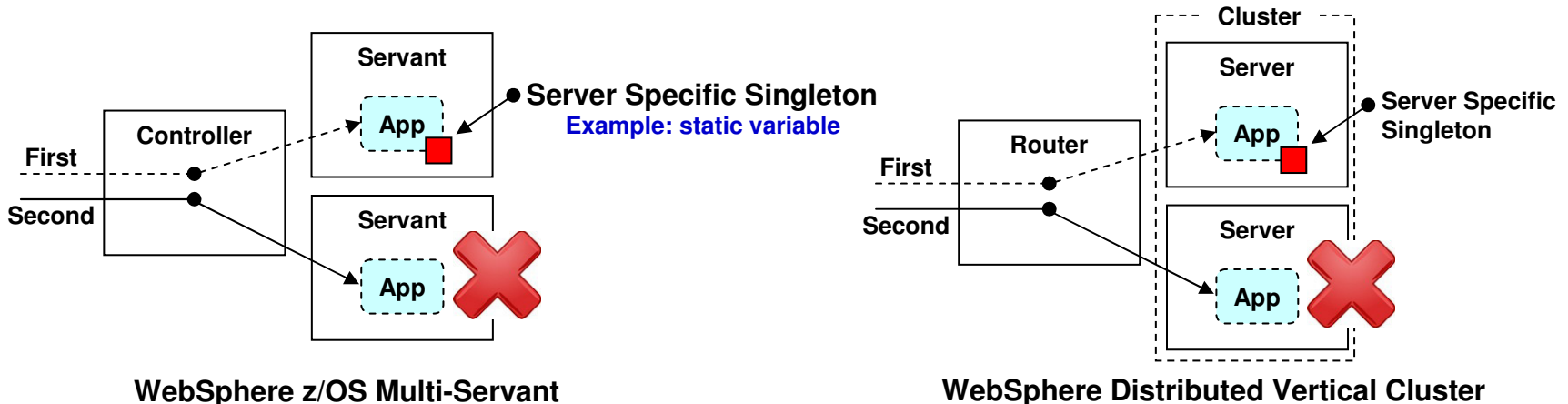
And neither is a “problem” with the platform, but rather a problem with Java code that makes improper assumptions

### Hard-coded references to properties or output files



## Assuming a single JVM instance

This is different from the use of HTTP session objects. That is a container-managed mechanism and can be handled in multi-JVM cases

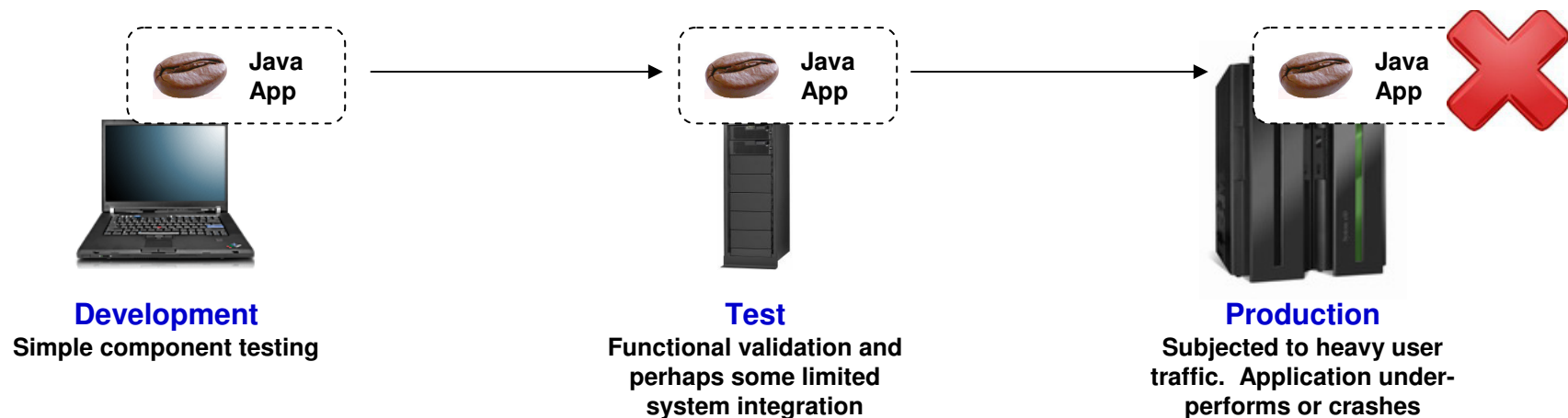


**Both not recommended since the underlying assumption is incorrect**

## Efficiency -- Write Smart, Tight Server Side Code

An application expected to scale up to heavy user rates must be sure to use very efficient server side code

We've seen this often in the benchmark center ...



In every case an investigation revealed inefficient coding practices on heavily used components. Once fixed, the application scaled to the desired levels.

There are many different examples of inefficient coding practices. Going into all of them is outside the scope of this presentation. Key message is: code expected to scale must be efficient.

## Efficient Code Advice

Some broad pieces of advice ...

- **Have a performance expectation documented**  
Called a **“performance budget”** ... this guides development efforts and helps set proper expectations
- **Understand where time is spent**  
Tools like JINSight help you analyze -- “profile” -- code
- **Write heavily used code as tight as possible**  
Applications typically have some components more used than others -- optimize those
- **Consider JVM co-location and local methods**  
This is a way to avoid “expensive” remote EJB calls
- **Take advantage of caching where possible**  
Caching within application, or use WebSphere dynamic caching
- **Use statefulness wisely**  
Creating statefulness creates affinities, which hinder scalability and availability
- **Trust But Verify**  
Popularity in open source communities does not guarantee the package is robust, efficient and scalable.

## Summary of the Business Value Benefits of Good Java Code

### Broad Guiding Principles of Business Value:

- **Available and Reliable**

Unexpected outages due to inefficient code designs are avoided

- **Manageable**

Properly architected Java solutions are more easily managed. Consistent design practices allow for consistent management approaches.

- **Flexible**

Properly architected Java solutions allow for easier integration of new features and functions. Poorly designed solutions require more time and effort to incorporate changes or expose as services.

- **Affordable**

Classic case of “pay me now or pay me later” ... spending time up front to design and craft good code pays dividends down the road in terms of reduced administrative costs, reduced resource consumption, and reduced cost to the business from outages or insufficient service to the customers.

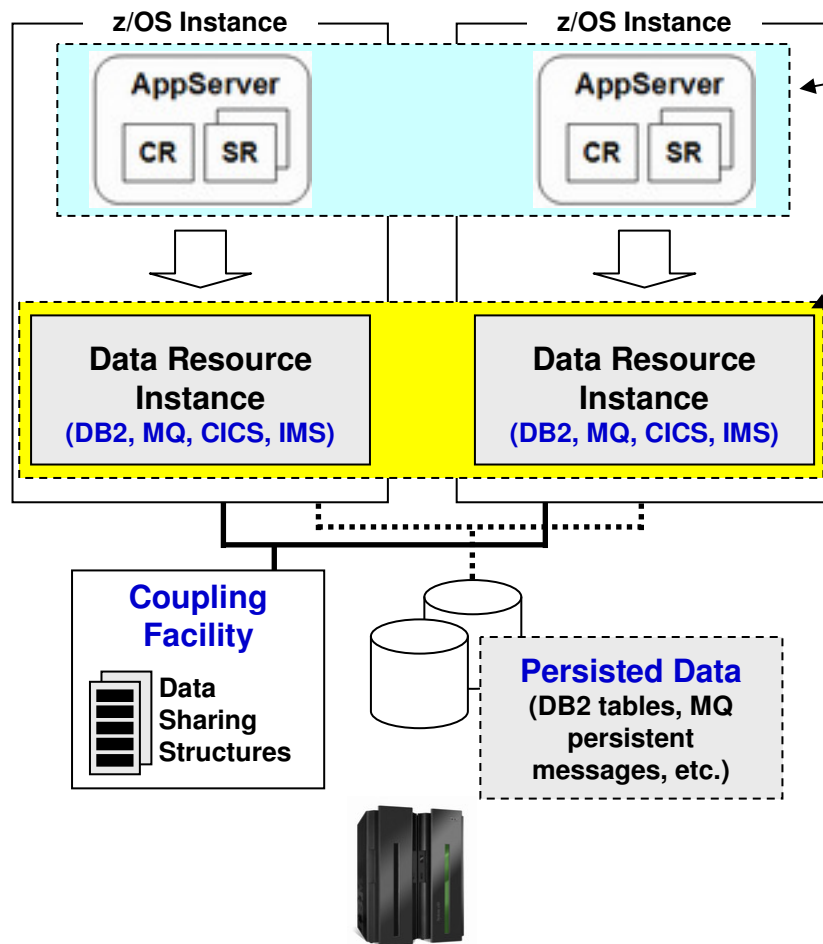


# Availability and Scalability

*Two of the more common business drivers for System z and z/OS*

## At The Heart -- Sysplex Data Sharing

Parallel Sysplex data sharing provides duplicated access to the same data. Data access and locking issues provided by Coupling Facility and Subsystems



- WebSphere “Cluster” consists of multiple physical application servers  
They are physically separate in most ways. Together it represents a logical one.

- Sharing Group  
Physically separate instances but organized so they understand participants in group and have defined sharing relationship

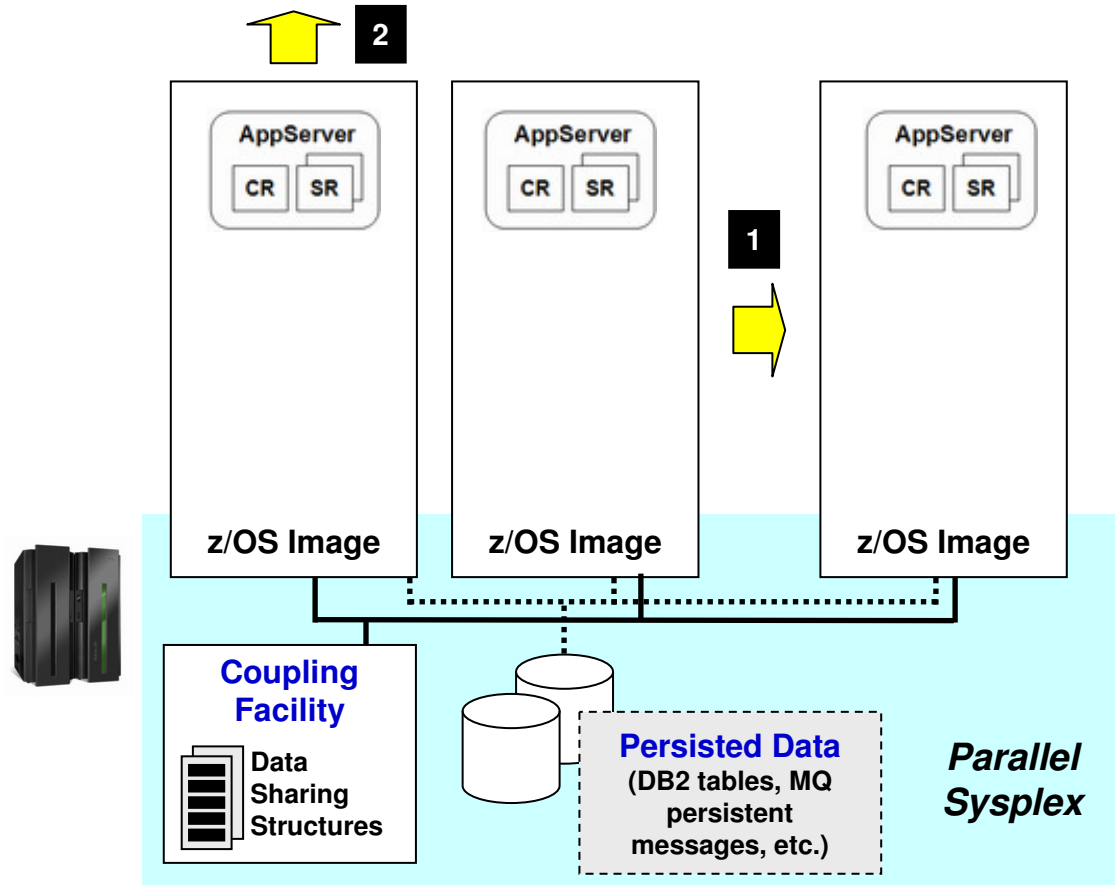
### AppServers and Resource Instances

An application in an AppServer interacts with a data resource *instance*. Sharing conflicts are resolved by the data resource instances working in concert with each other with the help of z/OS and the Coupling Facility.

Parallel Sysplex and Data Sharing has been around for a decade and more. The technology is mature and proven and in use by large customers the world around.

# Scalability

## Two kinds of scalability -- Horizontal and Vertical



### 1. Horizontal Scaling

This is what people most often think about when they think of scalability.

It can work, but it gets increasingly difficult unless you have an effective shared resource (data) clustering mechanism.

Parallel Sysplex is just such a mechanism.

Shared disk storage systems, proven locking mechanisms, in-memory data structures and caching (CF) all make for effective horizontal scaling.

### 2. Vertical Scaling

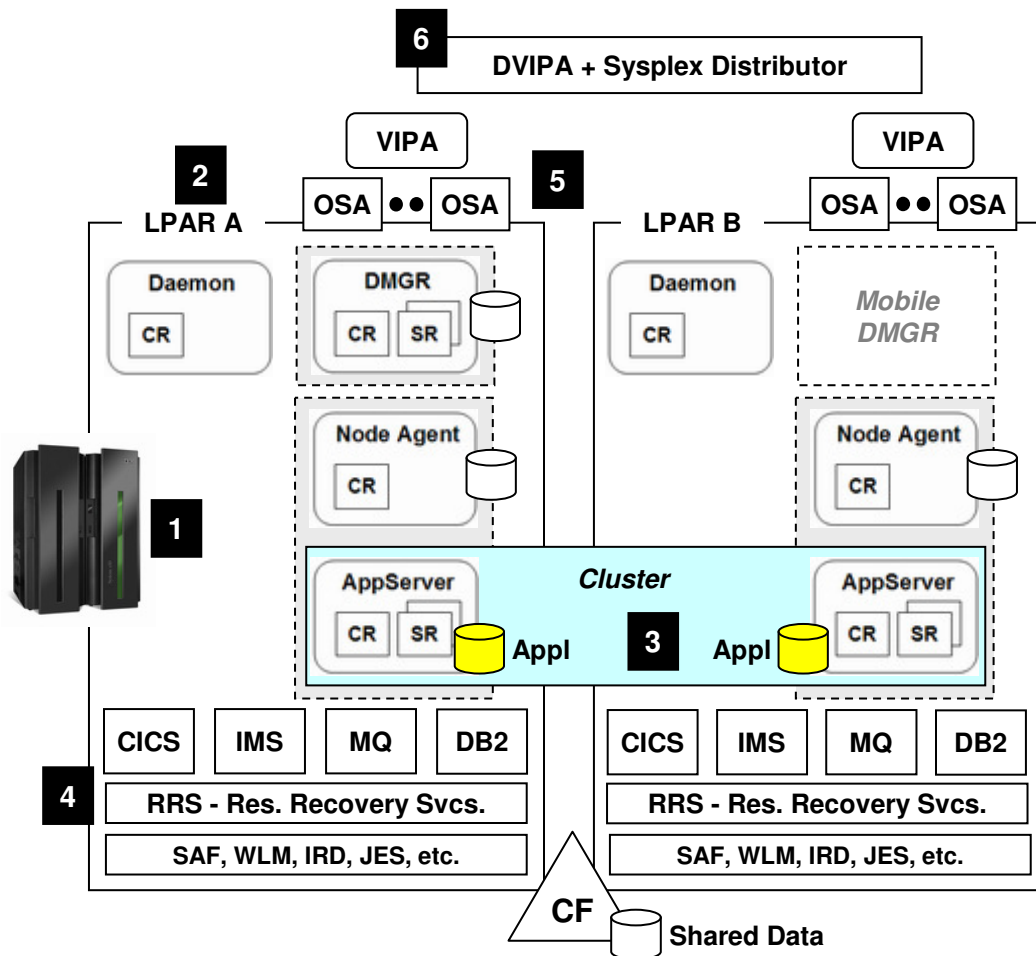
Vertical scaling is often overlooked. The result is massive horizontal scaling with all the attendant issues of manageability.

z/OS is designed for high degrees of utilization and has the capability to scale very high per system image. The balanced architecture (CPU, memory, cache, I/O) allow for this.

**System z and Parallel Sysplex provides both. That's the design point of the platform. That's how it's used in many large customer installations.**

# The Big Picture of WAS z/OS and Parallel Sysplex HA

It's all about redundancy *and* integration with platform HA function

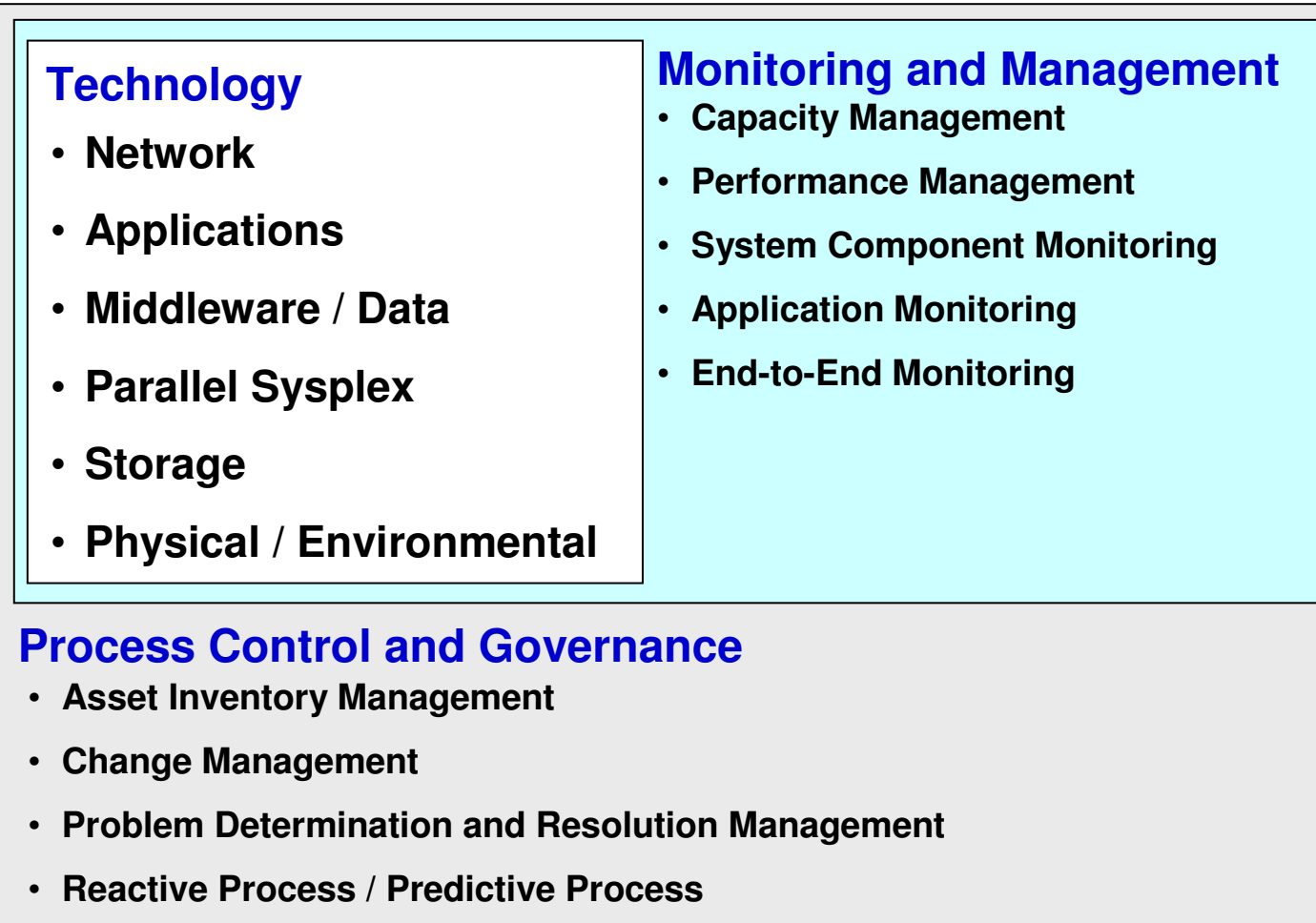


1. **Redundant and fault-tolerant hardware**  
System z hardware design has many layers of fault tolerance and redundancy.
2. **Redundant z/OS instances**  
Either through logical partitioning (LPAR) or separate physical machines.
3. **Clustered WebSphere z/OS servers**  
Multiple application servers grouped into a logical unit for application deployment and management  
z/OS exclusive: dynamic SR expansion (more coming up)
4. **Redundant data resource managers with Sysplex shared data**  
Multiple resource managers instances with shared data in CF and a global syncpoint manager (RRS)
5. **Redundant network adapters hidden behind Virtual IP address**  
On the front end, multiple network interfaces with a moveable virtual IP address protecting against outage
6. **Workload distribution hidden behind distributed virtual IP and Sysplex Distributor**  
Further abstraction of real IP addresses behind a virtual IP that can be swapped across images in a Sysplex, with Sysplex Distributor providing TCP connection distribution based on WLM

We show two operating system instances. That can be higher for greater availability and more manageable failover

# The Strategic Picture of High Availability

It's *more* than just technology ...



HA is a big topic

It's a total system thing ... focusing on one element of the design does not assure HA

It's as much a question of process as it is technology

An important first step is to determine what level of HA the business really requires

## Summary of the Business Value Benefits of Scaling and HA

### Broad Guiding Principles of Business Value:

- **Available and Reliable**

This is the heart of this topic. The System z hardware design is inherently stable and mature (passive benefit). The z/OS operating system is inherently stable and mature (passive and active benefit). The design of Parallel Sysplex provides for both scalability and availability. The key value attribute of the platform.

- **Manageable**

Mature systems management tools make managing a Sysplex effective.

- **Flexible**

WLM and IRD represent extraordinary flexibility of resource allocation across a Sysplex within a CEC. Many elements of the hardware and system software design provide hot-pluggability and dynamic vary online and off.

- **Affordable**

Higher initial acquisition with lower overall, depending on a number of costing factors. We cover this at the end of the presentation.

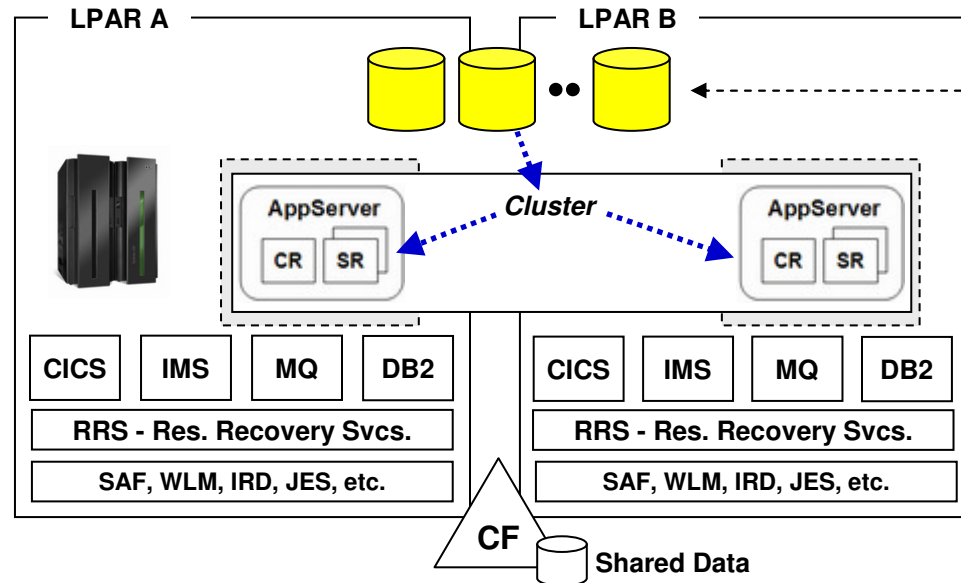
# Other WAS-Based Solutions

*A quick note about IBM's strategy to leverage the WAS z/OS solution*



## Other WAS-Based IBM Solutions

We see many IBM z/OS solutions being built as Java EE applications that leverage the existing foundation of WAS z/OS:



### Examples:

- WebSphere Process Server (WPS)
- WebSphere Enterprise Service Bus (WESB)
- WebSphere Service Registry and Repository (WSRR)
- WebSphere Extended Deployment
  - Virtual Enterprise
  - Compute Grid
  - Extreme Scale

Why reinvent the wheel? IBM has a robust, platform-exploiting Java EE runtime environment in WAS z/OS. It's proven and it works well.

Why not just piggyback on all the key services it provides (security, transaction, containers, data access) that WAS z/OS provides? That's exactly what's done.

All the benefits that accrue to WAS z/OS accrue to these solutions as well

## Summary of the Business Value Benefits of Leveraging WAS

### Broad Guiding Principles of Business Value:

- **Available and Reliable**

All the availability and reliability statements made earlier apply to the solutions built on the Java EE base of WAS z/OS

- **Manageable**

All the manageability statements made earlier apply to the solutions built on the Java EE base of WAS z/OS

- **Flexible**

All the flexibility statements made earlier apply to the solutions built on the Java EE base of WAS z/OS

- **Affordable**

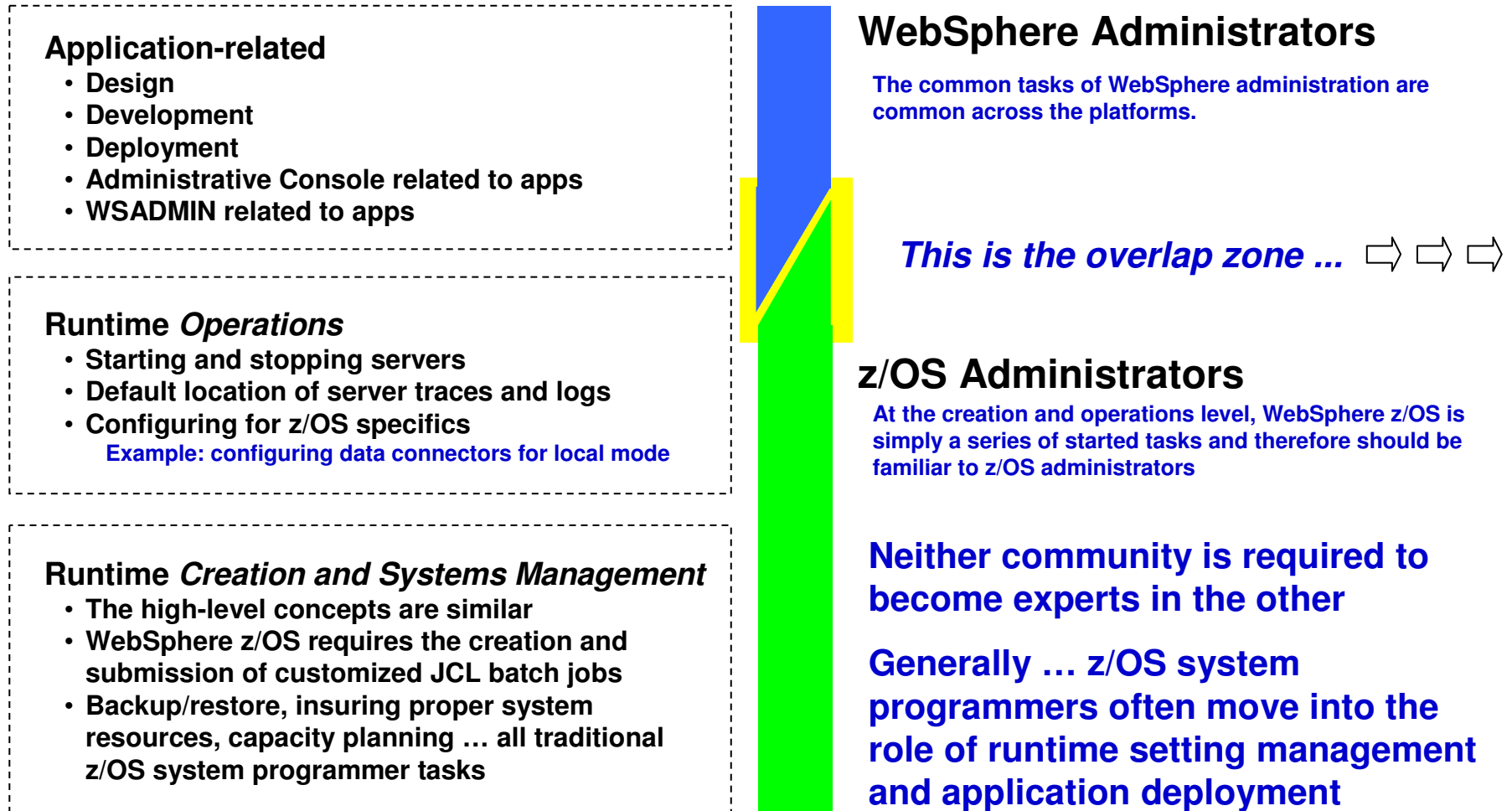
Higher initial acquisition with lower overall, depending on a number of costing factors. We cover this at the end of the presentation.

# Administration and Skills

*Where the z/OS skills intersect with WAS skills and how to manage the overlap*

## Skill Communities

We see two groups of people -- WebSphere Administrators and z/OS Administrators ... where do the two meet?



## Common Patterns of Skills Development

Some observations from other customers:

### WebSphere Administrators Seeking Some Knowledge of z/OS

*Not required ... WAS administrator can be effective with no z/OS knowledge if z/OS environment "hidden" from them*

Generally speaking, the objective is to provide knowledge of common activities:

- Logging into TSO, a little on using ISPF
- Becoming familiar with concept of started tasks
- Understanding how to browse held output in JES
- Knowing how to check started task status
- Starting/stopping servers (maybe)

Objective is rarely to make them z/OS administrators. **No need** to learn in-depth:

- WLM classifications
- RACF or other SAF product
- JCL, z/OS commands, etc.

We often see access to z/OS using familiar tools:

- Telnet / FTP for UNIX access
- WebSphere Admin Console (essentially common)
- WSADMIN scripting (common across platforms)

### z/OS Administrators Seeking Some Knowledge of WebSphere

*Some knowledge is helpful*

Generally speaking, the objective is to provide conceptual framework of WebSphere:

- Understanding the runtime architecture
- Construction of runtime
- Key systems management of runtime

Objective is rarely to make them deep WAS administrators or Java programmers. **No need** to learn in-depth:

- Java programming
- Developing tooling

Frequently see z/OS personnel desire to expand skill set and learn things like:

- JVM monitoring and tuning
- Application deployment techniques
- WebSphere runtime customization details

## Summary of the Business Value Benefits of Skill Communities

### Broad Guiding Principles of Business Value:

- **Available and Reliable**

The platform provides the ability to provide this; your System z and z/OS staff is what makes it work. WAS z/OS is implemented in a way where existing z/OS skills can be effectively leveraged without requiring massive re-learning of new technologies.

- **Manageable**

WAS z/OS directly exploits many standard z/OS management functions: MODIFY, SAF, SMF, RMF, WLM.

- **Flexible**

WAS z/OS is designed to directly exploit the z/OS flexibility attributes.

- **Affordable**

Existing z/OS staffs have effectively managed the addition of z/OS without additional staffing. Existing distributed WAS administrators can use WAS z/OS with little or no awareness of the different platform.

The key is effective and efficient use of *existing* staffing resources.

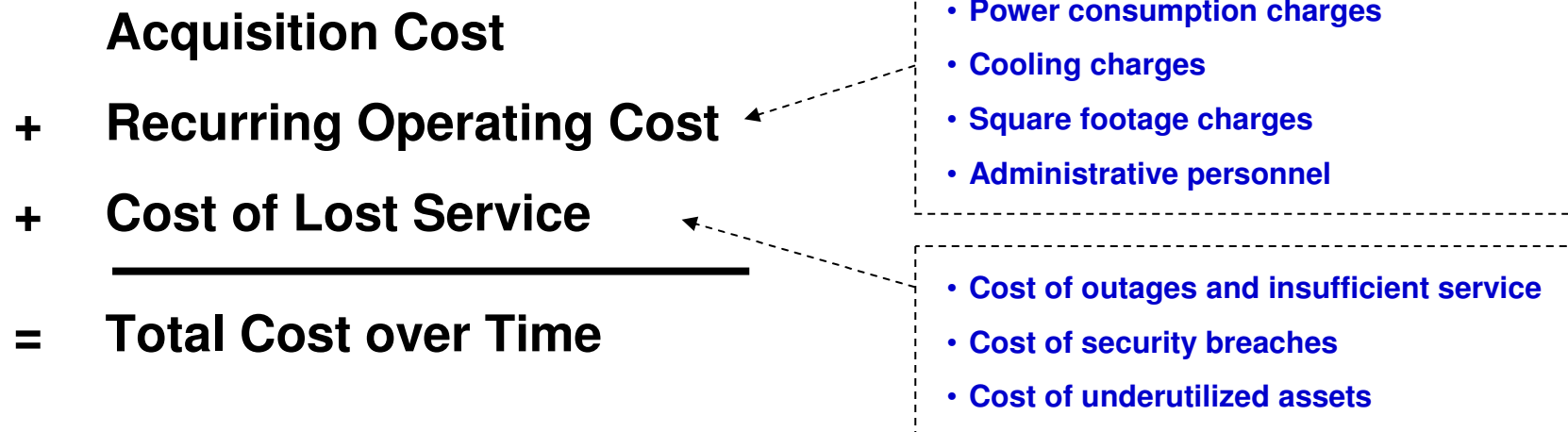
# Addressing the Issue of Cost

*Acquisition cost versus Total Cost of Ownership*



## Three Costs Go Into Total Cost

It is an often complex undertaking to calculate the true total cost



We are *not* minimizing the pressures you face to keep acquisition costs low

We are *not* suggesting determining some of these costs is easy

We are asking that give consideration to the total cost picture

## Really Quick Primer on Fixed Cost Allocation

This is at the heart of the “total cost of ownership” (TCO) issue. Let’s imagine a hypothetical household:

**Mortgage**  
\$36,000 / year

**Property Taxes**  
\$10,000 / year

**Insurance**  
\$5,000 / year

**Utilities**  
\$5,000 / year

**College Tuition**  
\$30,000 / year

**All Other Costs**  
\$14,000 / year



**\$100,000 per year**

The objective is to allocate that across some “marginal” (incremental) unit

(Reason is because businesses wish to charge operating units to cover overhead, and because they want to estimate profit per unit)



Per Person?

$$\begin{aligned} \$100K / 4 = \\ \$25K / \text{person} \end{aligned}$$



Per Square Foot?

$$\begin{aligned} \$100K / 3000 = \\ \$33 / \text{sq. foot} \end{aligned}$$



Per Pet?

$$\begin{aligned} \$100K / 2 = \\ \$50K / \text{pet} \end{aligned}$$



Per Kilowatt Hour?

$$\begin{aligned} \$100,000 / 14kWh = \\ \$7,000 / kWh \end{aligned}$$



Per Meal Consumed?

$$\begin{aligned} \$100K / (4 * 365 * 3) = \\ \$23 / \text{meal} \end{aligned}$$

This “cost” per unit can affect future purchase or activity decisions ...

“Having your mother move in would cost us \$25K extra!”

“That addition is going to cost more than I expected!”

“No, Billy can’t get a goldfish!”

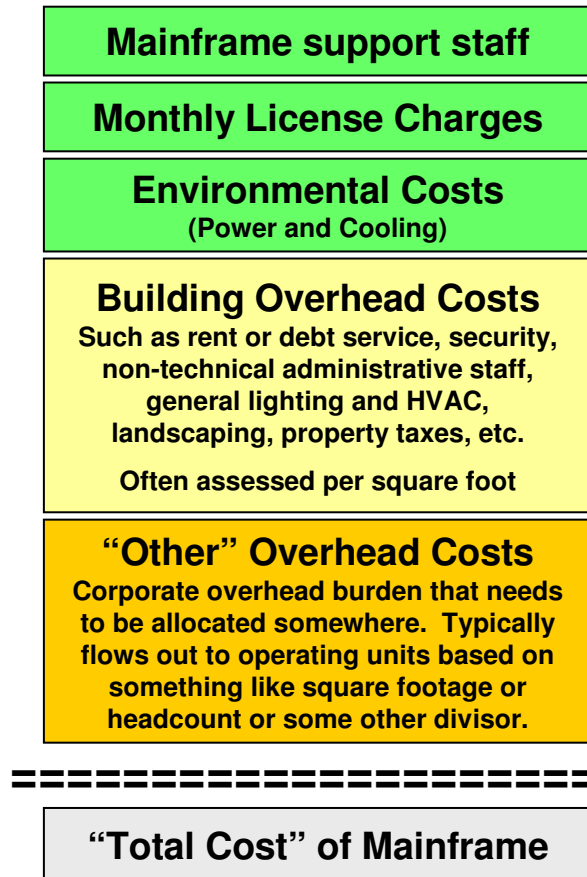
“Turn off the lights!”

“You just ate! You want something else?”

The allocated fixed cost does not reflect the *true* marginal cost ... but it can still influence perception

# The Chargeback Conundrum

Chargeback can be a complex topic. But understanding how the “mainframe” costs are charged back is essential to the TCO question.



**“Other” I/T Costs**  
 Due to lack of good accounting tools on early distributed platforms, the cost of non-mainframe I/T -- including support staff -- is put into mainframe bucket. Practice never revisited ... most of distributed server farm costs carried by mainframe

*Is this fair? No.  
 Does it happen? Yes.  
 Why? Because it's an easy and convenient cost accounting bucket*

$$\begin{aligned}
 & \text{=====} \\
 & \text{“Cost of the Mainframe” Number} \\
 & \div \text{ Mainframe CPU Cycles Used} \\
 & \text{=====} \\
 & \text{Cost per CPU Cycle To Use MF}
 \end{aligned}$$

The impression formed is ...

“Cheap acquisition, free otherwise”



“High acquisition, high usage charge”



**Neither is accurate. True TCO will account for costs attributable to the distributed server farms. Mainframe would bear only its fair share.**

Challenge is that some costs are not clearly associated with one or other. How to fairly allocate? Cost accounting is not easy.

## Summary of the Business Value Benefits for Costing

### Broad Guiding Principles of Business Value:

- **Available and Reliable**

*Addressed earlier in the presentation*

- **Manageable**

*Addressed earlier in the presentation*

- **Flexible**

Operational flexibility addressed earlier in the presentation

SMF allows for usage details down to the request, as opposed to costing based on server box

- **Affordable**

TCO studies have shown the mainframe to be an affordable solution when costs are properly taken into account

# Overall Summary

## Overall Presentation Summary

- **The System z and z/OS platform has technical qualities that are part of its design**
- **Those technical qualities can map to your business objectives**
- **Two ways to derive the benefits:**
  1. **Passively -- simply by running on the platform**
  2. **Actively -- by directly making use of the features**
- **WebSphere Application Server for z/OS is designed to actively exploit the value features of the System z and z/OS platform**
- **WebSphere Application Server for z/OS is capable of extremely high degrees of scalability and availability if that's a business objective**
- **The administrative skills to run WAS z/OS are in many ways common with WAS on other platforms.**
- **The cost of WAS z/OS is often cited as an inhibitor. We ask that the question of cost allocation be looked at objectively.**