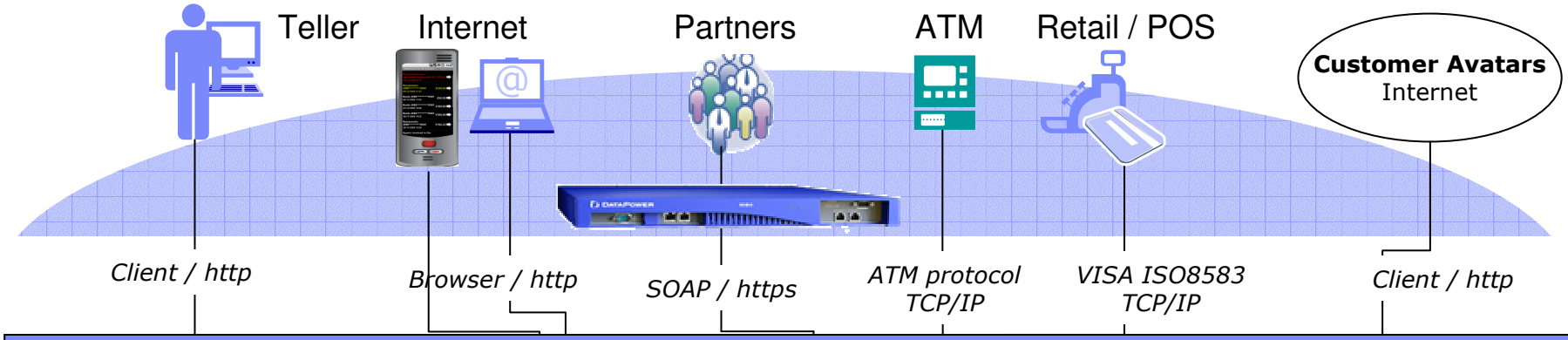
Enterprise Application Integration

Smarter Banking Showcase Team
WW Banking Centre of Excellence, IBM Montpellier

Enterprise Application Integration within Showcase



Enterprise Application Integration (EAI)
is defined as the use of software and
computer systems architectural principles
to integrate a set of enterprise computer applications.

Browser or client / http
System Operations
LOB operations
Intranet

Browser or client / http
Bank Operations
Intranet

Browser or client / http
Management/Analysts
Intranet

Websphere Application Server - Introduction

WebSphere is WebSphere at the specification line and above

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

Differentiation occurs *below* the specification line

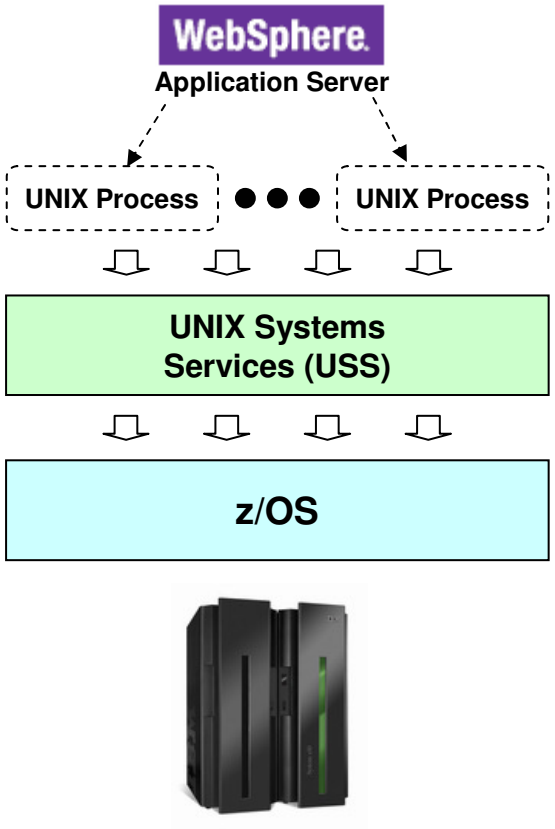
How it's *implemented* is dependent on the platform … its features, functions, attributes and qualities of service

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?**

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"**

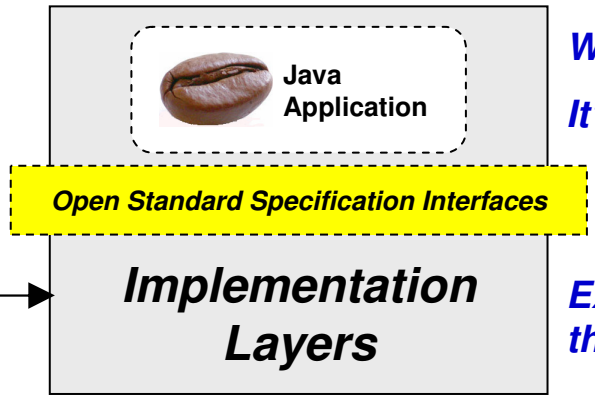
Let's expand on that a bit …

Active Exploitation Features

1. Exploitation of SMP/E
2. Exploitation of JES and common z/OS facilities
3. Exploitation of zAAP specialty engines
4. Exploitation of WLM
5. Exploitation of RRS
6. Exploitation of SAF and Crypto
7. Exploitation of SMF
8. 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



WebSphere is WebSphere up here

It's all based on accepted standards

Exploitation taking place below the open standard interface line

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.

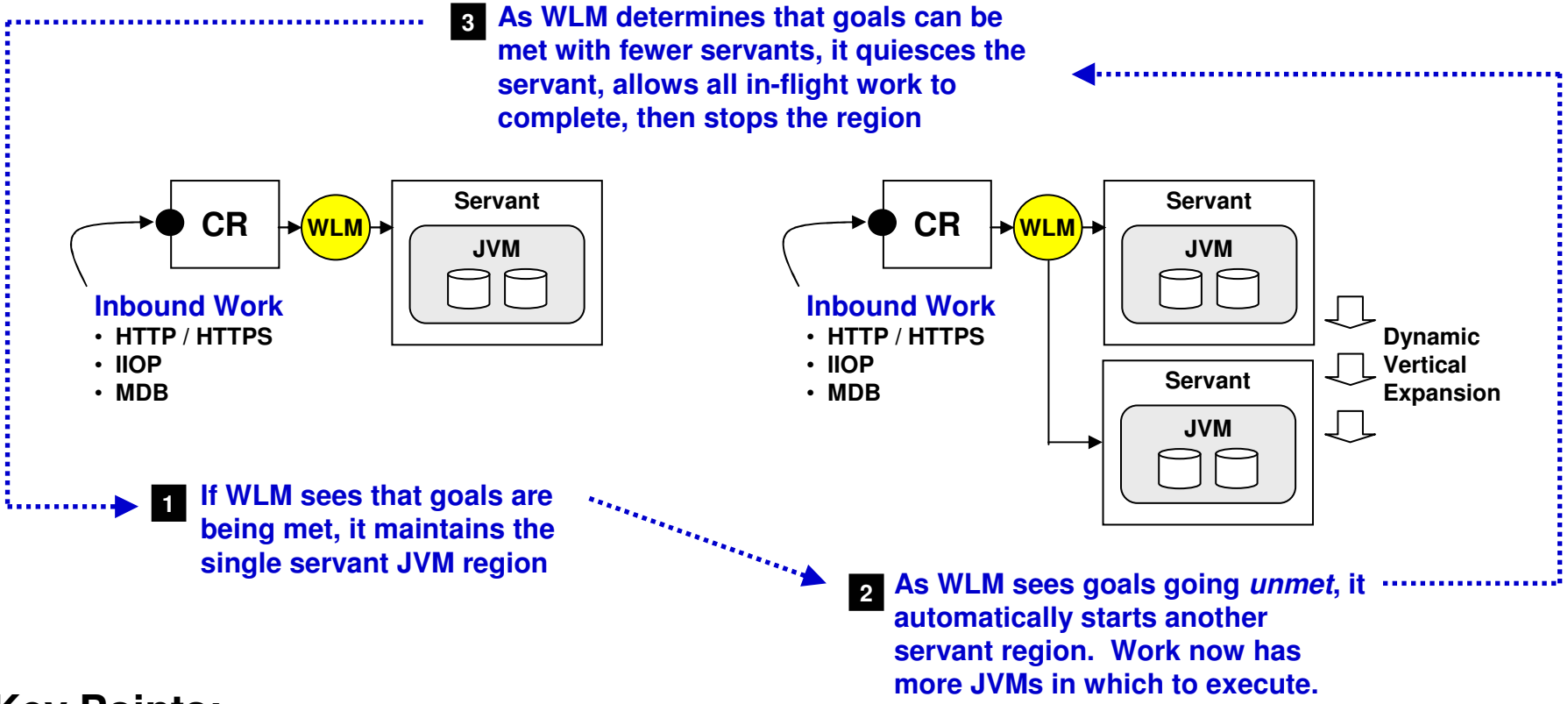
Intelligent Workload Routing Advice

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

The key is the controller / servant architecture …

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:

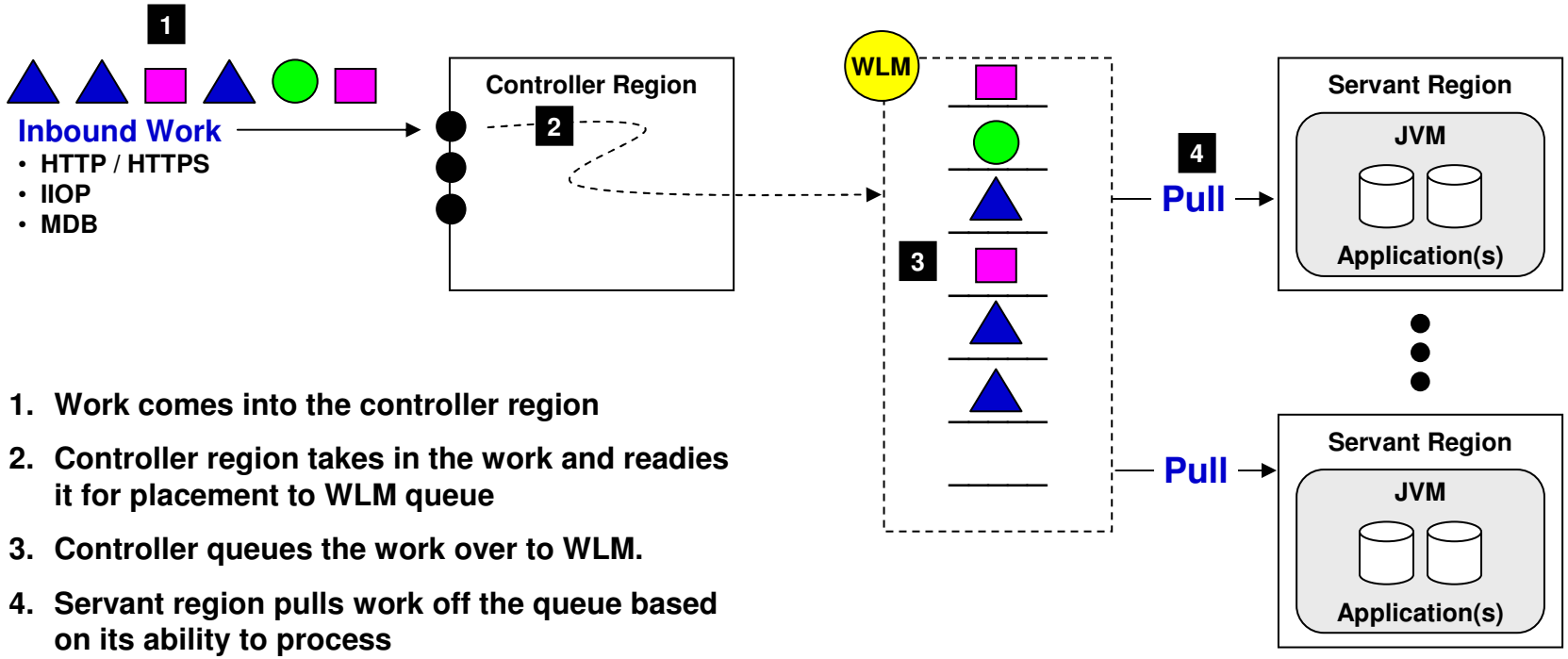


Key Points:

- 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 queueing mechanism that exists between the CR and the SR. It creates a "pull" model that prevents overwhelming an application JVM:



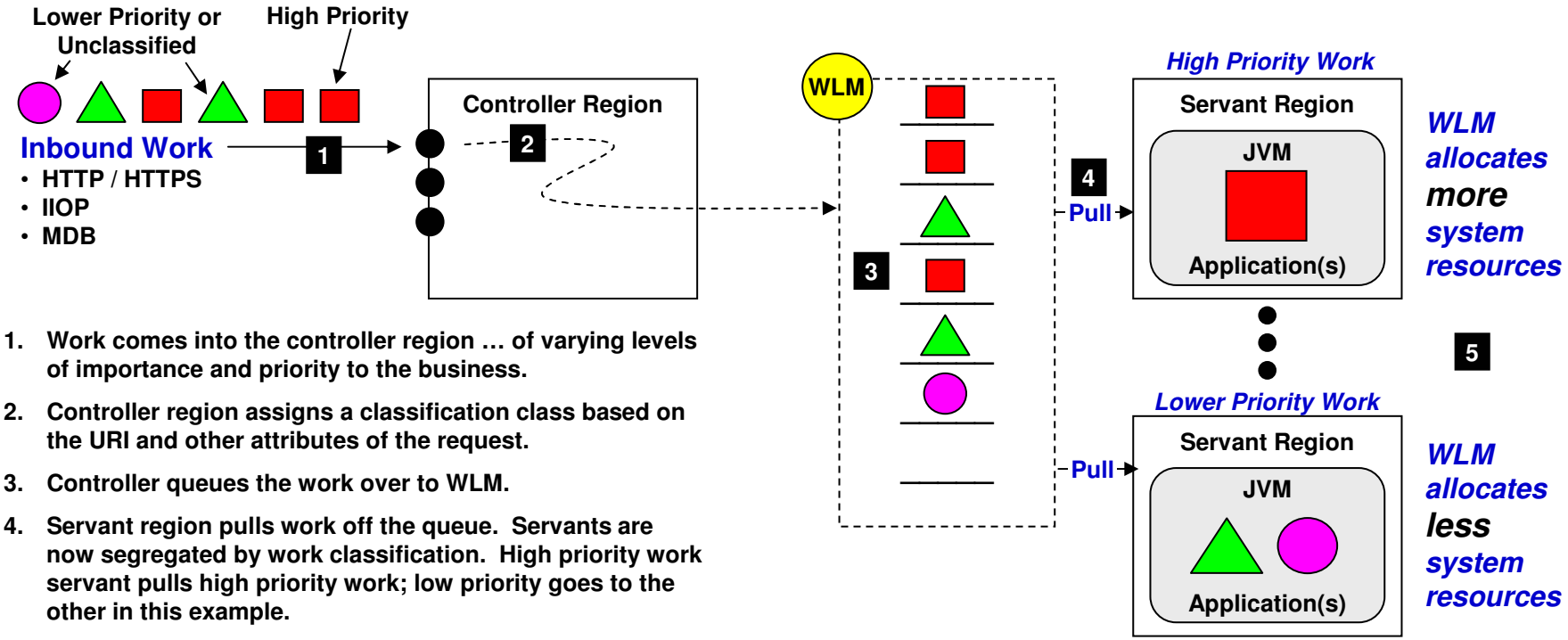
1. Work comes into the controller region
2. Controller region takes in the work and readies it for placement to WLM queue
3. Controller queues the work over to WLM.
4. Servant region pulls work off the queue based on its ability to process

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.

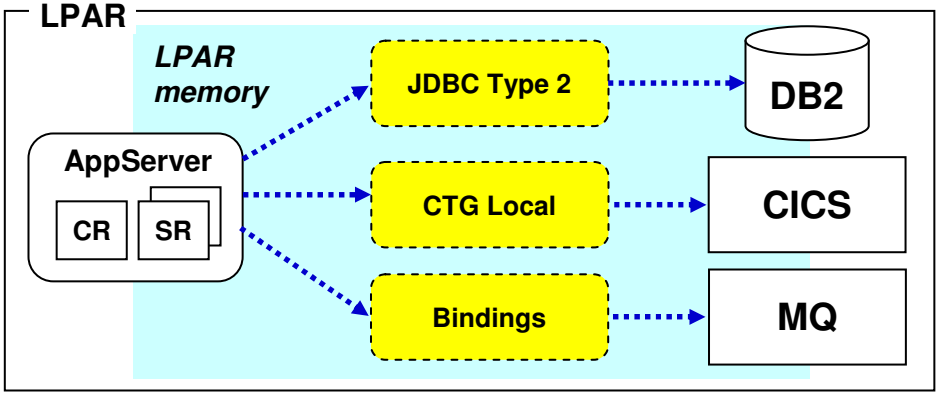
Sophisticated Work Prioritization

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

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

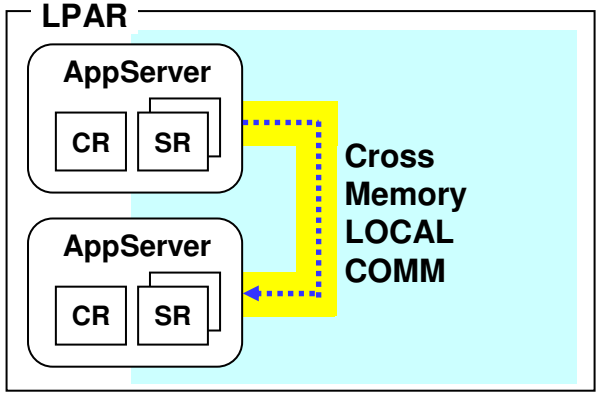


Benefits:

- 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

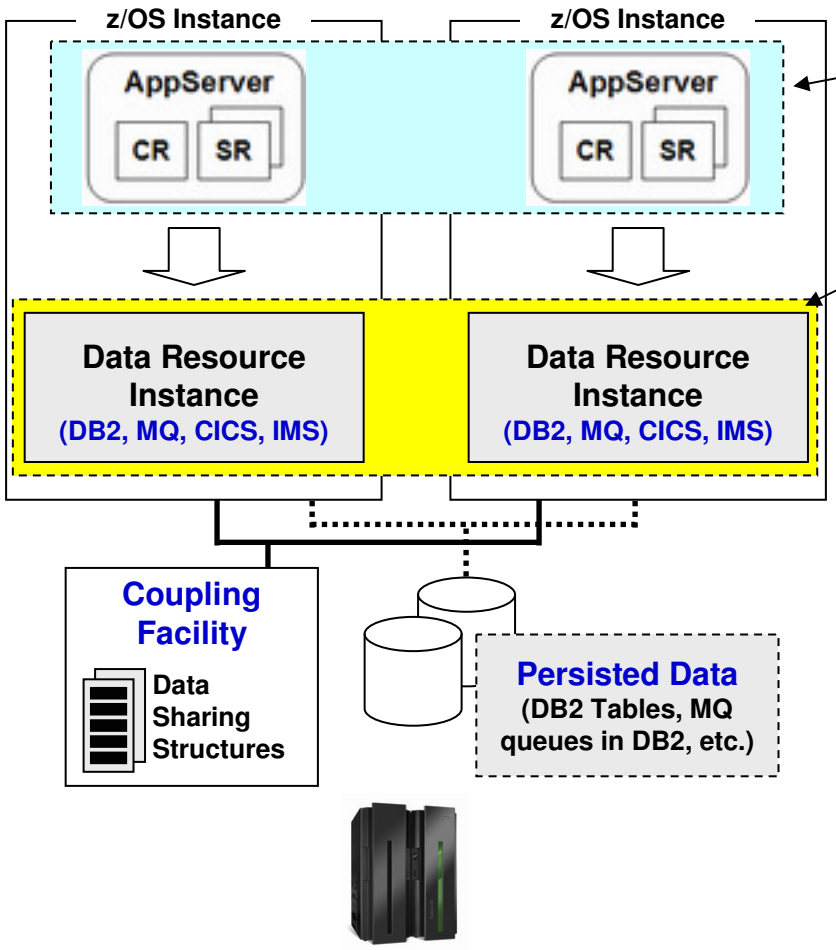
Extension to Local Comm: new Optimized Local Adapters ...

Availability and Scalability

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

Availability - 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

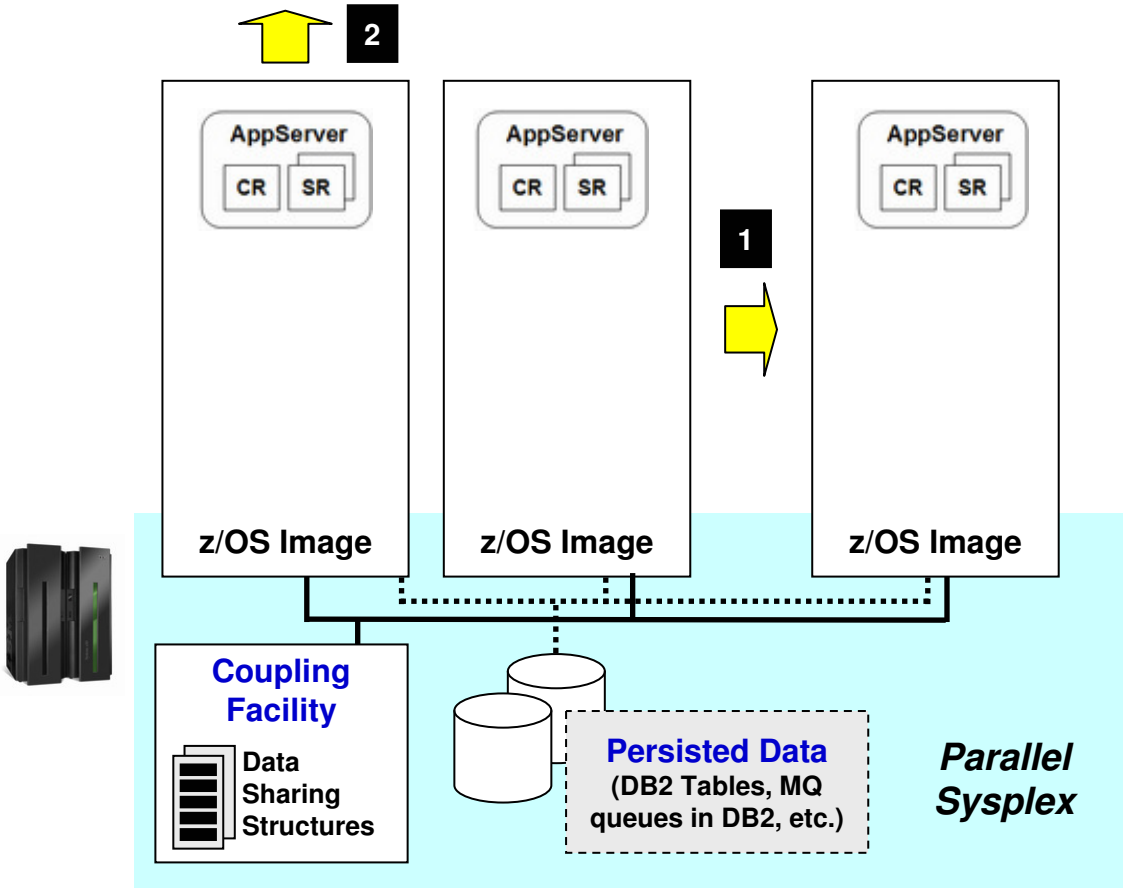


- **WebSphere "Cluster" consists of multiple physical application servers**
  - They are physically separate in most ways. Together it represents a logical one.
- **Sharing Group**
  - Physically separate instances but organized so they understand participants in group and have defined sharing relationship
- **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

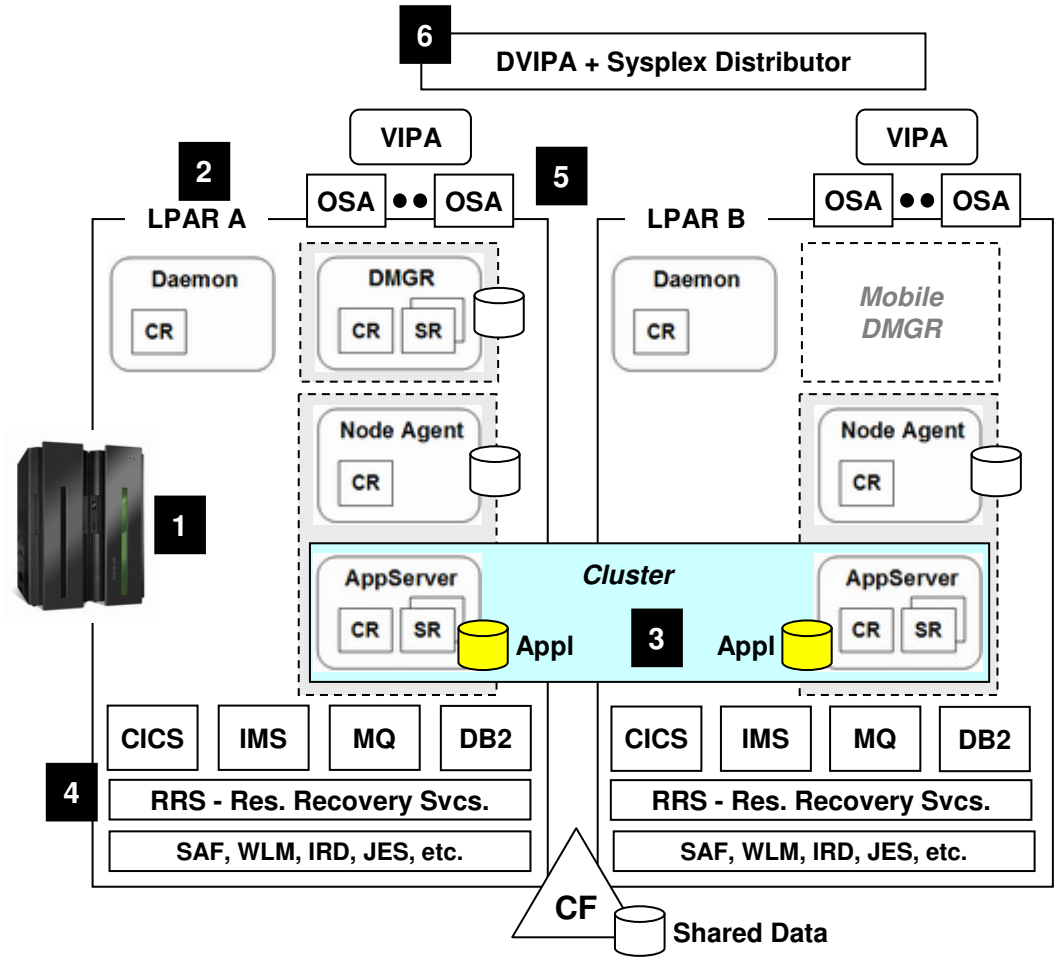
Vertical scaling is often overlooked. The result is massive horizontal scaling with all the attendant issues of manageability.

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.

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

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



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

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