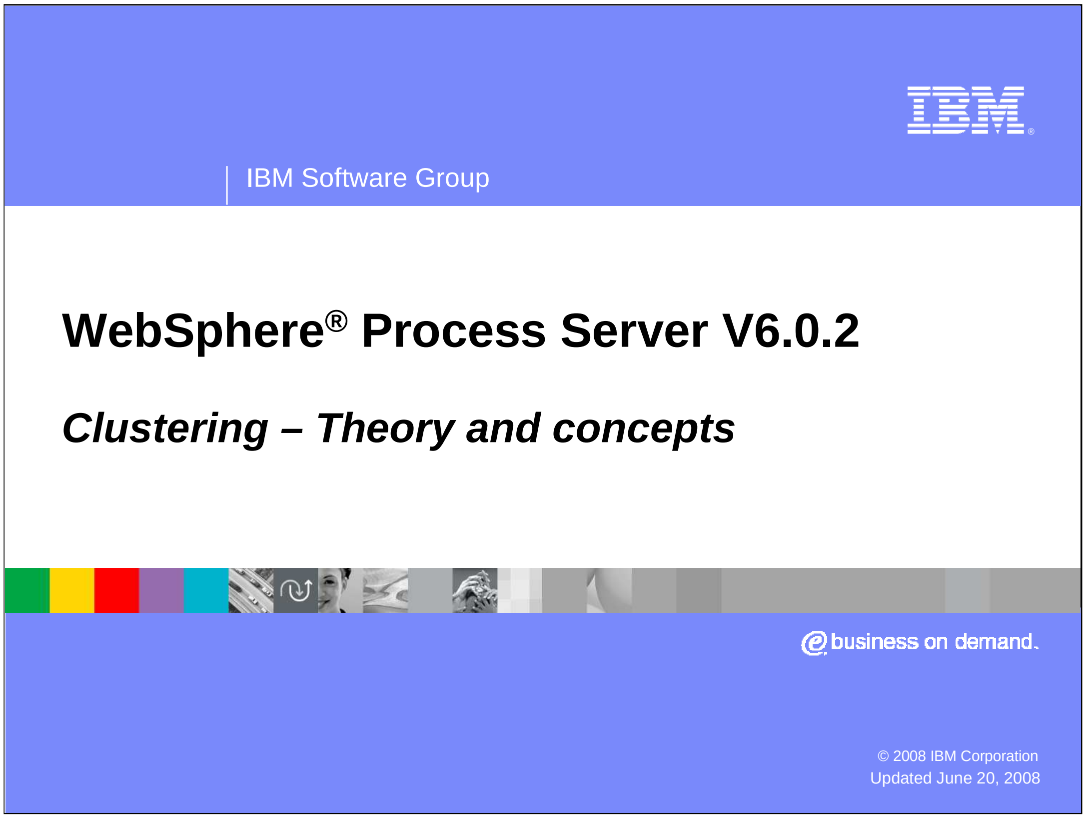
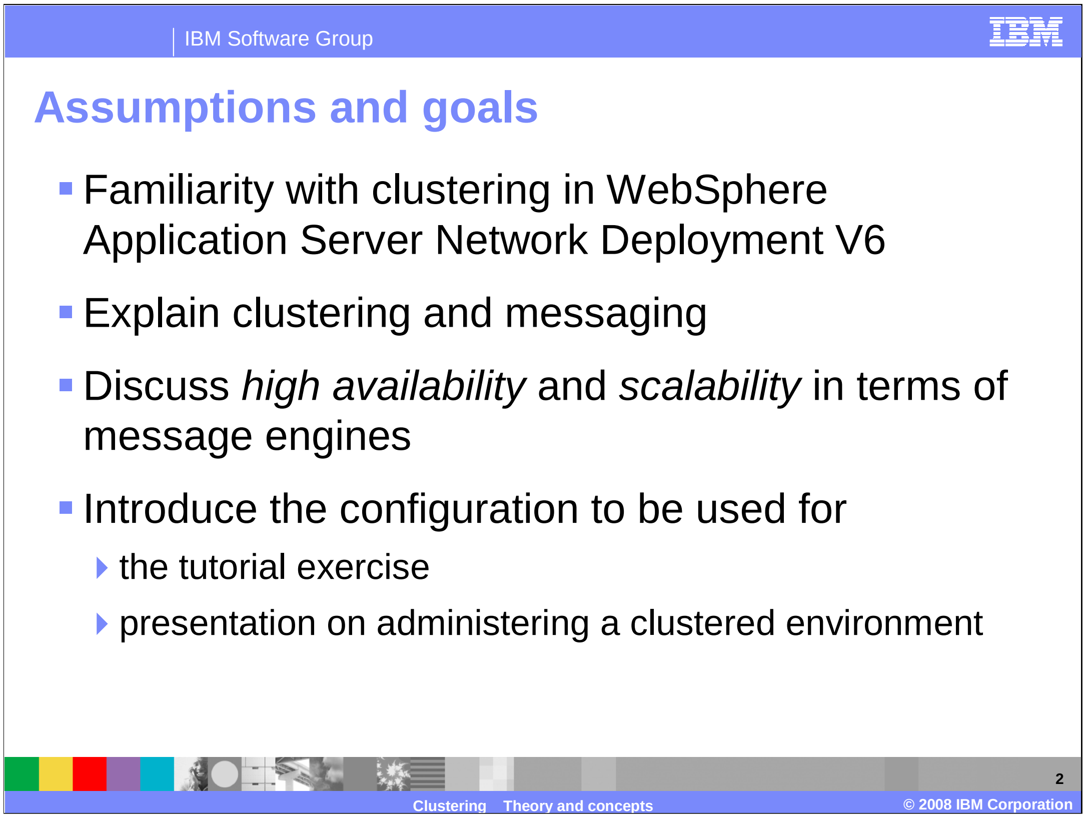


This presentation covers the theory and concepts of clustering for WebSphere Process Server V6



This presentation assumes that you are familiar with clustering in WebSphere Application Server Network Deployment V6.0. The goal is to present the underlying and motivating issues involved with clustering and messaging in a WebSphere Process Server environment. The next goal is to discuss high availability and scalability in terms of message engines. Then you will be introduced to the configuration for *high availability* that will be used for the tutorial and the presentation on administering a clustered environment.

IBM Software Group

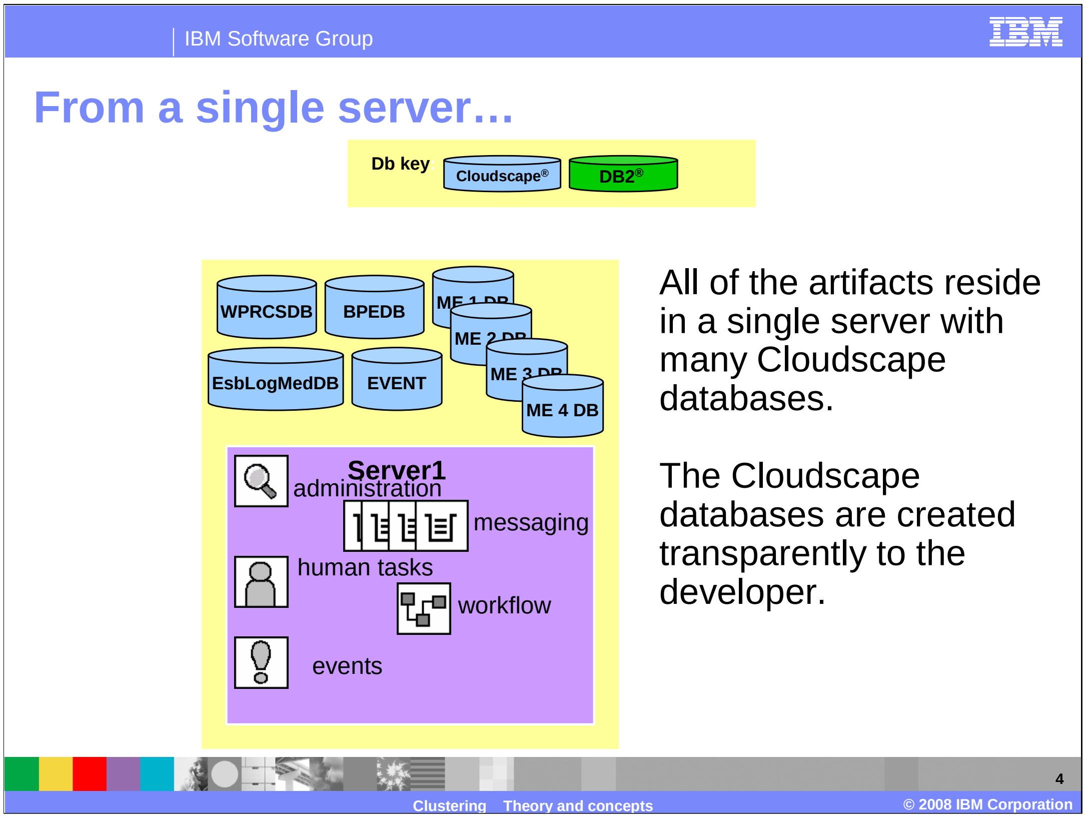
Agenda

- Overview
  - WebSphere Application Server to WebSphere Process Server
- Clustering basics
- Terms and definitions
- Messaging in WebSphere Application Server v6
- Messaging and clustering
- A deployment pattern
- Databases
- Other WebSphere Process Server components

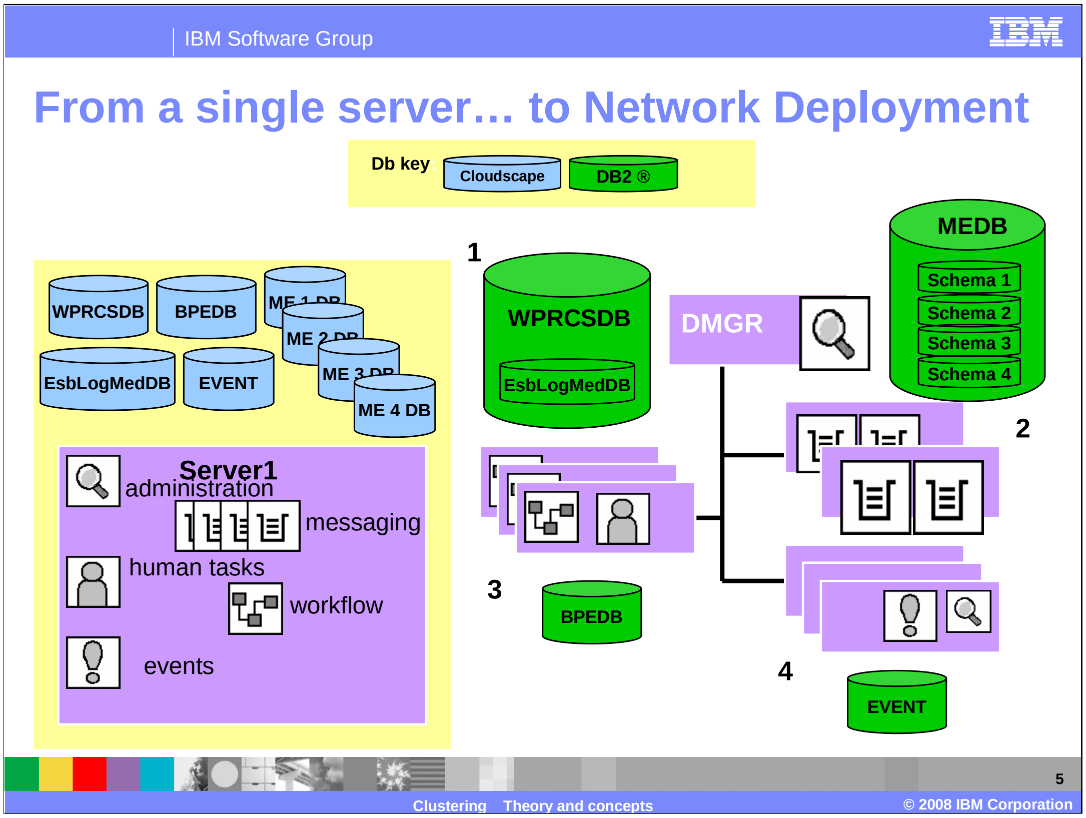
3

Clustering Theory and concepts

The agenda for this presentation begins with an overview of the differences between WebSphere Application Server and WebSphere Process Server. Then it will cover clustering basics, including some terms and definitions, before moving into Messaging topics. Finally, deployment options, database options, and other components will be discussed.



This slide shows the artifacts in a stand-alone server configuration. If using the WebSphere Portal Server test environment in WebSphere Integration Developer, all the artifacts are created for you automatically using the Cloudscape database. Outside of WebSphere Integration Developer, you have the possibility of using DB2 as your database of choice. Working with a network deployment configuration, Cloudscape is not supported and you MUST move to a DB2 database. The next slide will look at the network deployment configuration.



To get to the fully network deployed configuration….

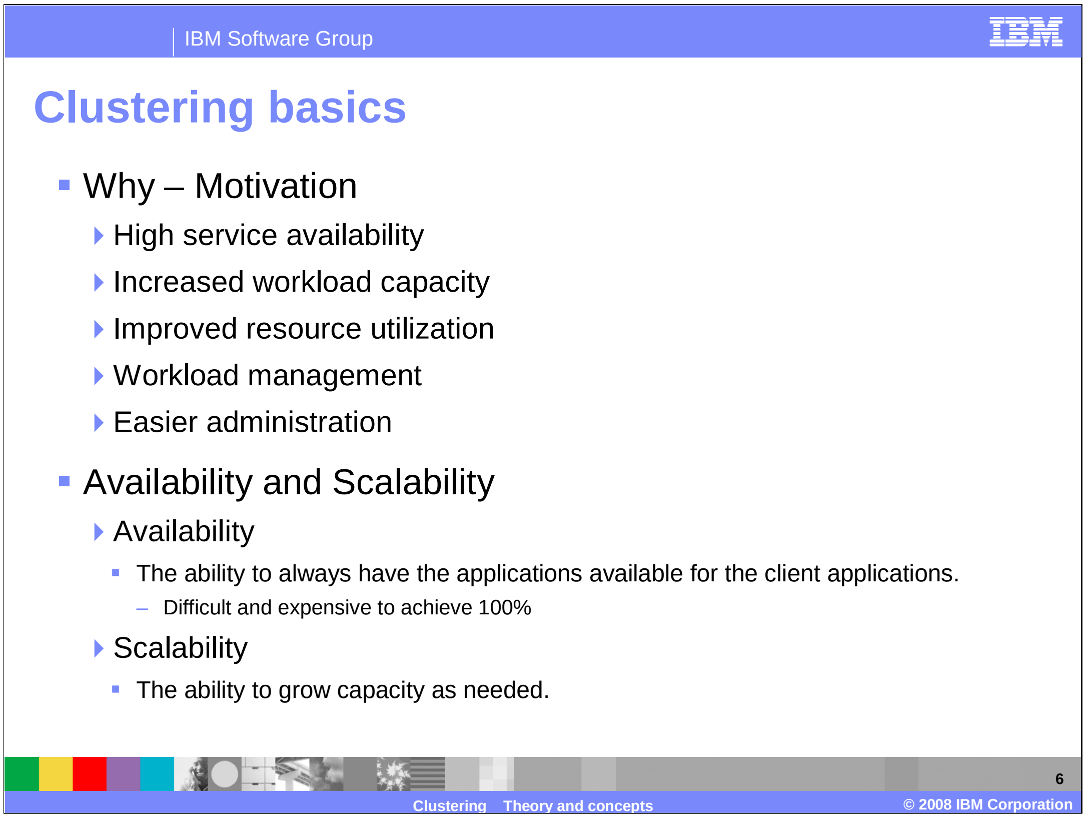
1.Create the WPRCSDB database and incorporate the tables required for the Enterprise Service Bus mediations. You need to create the tables and indexes for the common database (WPRCSDB) in DB2 rather than Cloudscape. This includes the tables required for the Enterprise Service Bus mediations.

2.When creating the messaging engines in the network deployment environment, the data stores for the message engines can be consolidated to a centrally located, remote database (except on z/OS® where the database is not remote). The JMS resources are established on numerous servers.

3.When business processes are used in the network deployment configuration, the BPEDB database is created in DB2. Note that numerous servers can host business processes in the network deployment scenario. Ideally, the servers would be grouped together into clusters but each cluster will require its own BPEDB instance.

4.The Common Event Infrastructure component running in the network deployment configuration also requires that a database be created. It will be used specifically for the Common Event Infrastructure events. These events can be generated from any of the servers, either from the Business Process Choreographer infrastructure or from applications directly.

The picture here provides a sense of what it takes to move from the development environment, to a fully distributed Network Deployment environment for either test or production. There are several databases to be created and managed and decisions about how to distribute the infrastructure and application components.



As an architect or system administrator using WebSphere Process Server to implement your business processes, you expect to have a system that will tolerate failure and allow for maintenance without the loss of data or loss of service. You also need to be able to add processing capacity, to grow your systems to meet increased user demand.

WebSphere Network Deployment provides the ability to create logical groups of servers, with the servers being distributed across one or more machines. This capability provides a mechanism to tolerate failover, apply maintenance to some servers while others keep running, grow or shrink capacity by adding and removing servers from the group, all with a single point of administration.

WebSphere Application Server Network Deployment - Key elements

- Cell
- Nodes
  - Deployment manager node – “DMGR”
  - Node – “NodeXX”
  - Node agent – “NAXX”
- Profiles
- Clusters
  - Cluster members
- Data sources
- Java™ messaging Service (JMS) resources
  - Service integration bus destinations
    - Queues, topics, factories
- Host machine – “the box”

Shown here are the key elements of a WebSphere Network Deployment Cell which are necessary for a discussion on clustering.

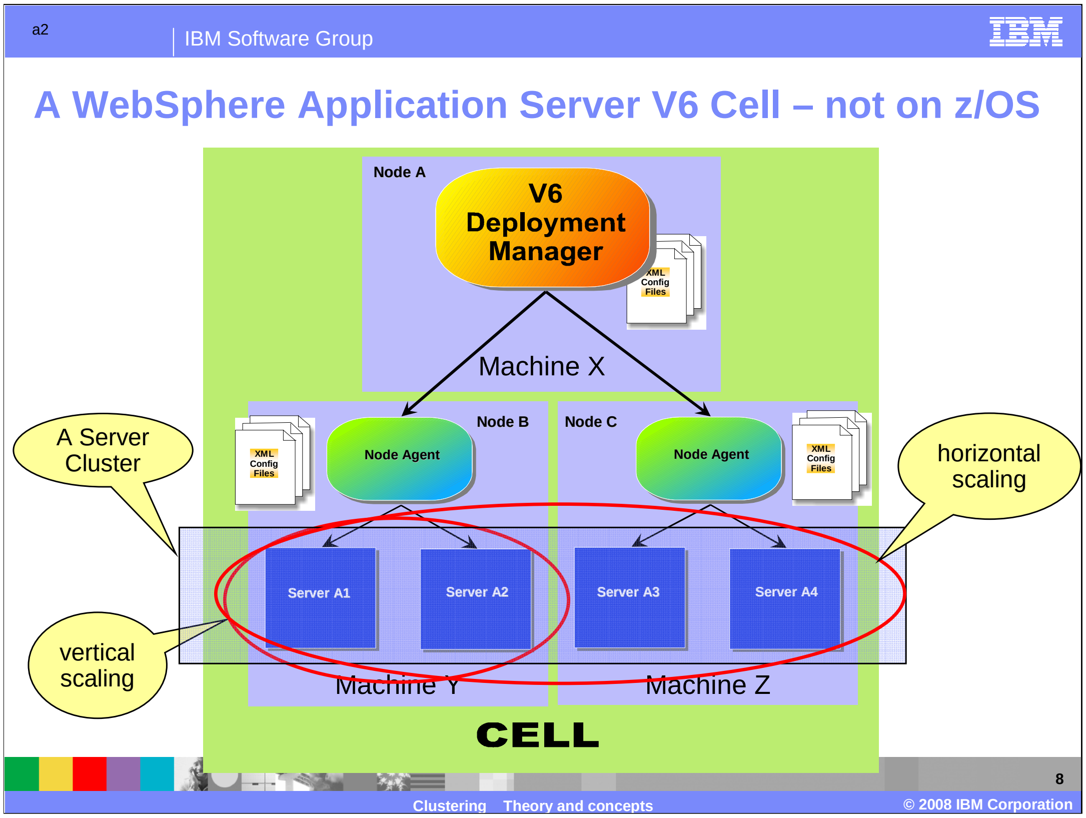
A **node** is a logical grouping of managed servers. A node typically corresponds to a logical or physical computer system with a distinct IP host address. Nodes cannot span multiple computers.

The WebSphere Application Server **profile** defines the runtime environment for a node. To create a node, that is to say, the runtime environment to host an application server on a given host machine, you run the Profile Wizard. One of the primary purposes of the profile is to provide separation between the WebSphere runtime artifacts specific for a given configuration, and the WebSphere Application Server Binaries, which are common to all configurations.

Clusters are sets of identical servers that are managed together and participate in workload management. The servers that are members of a cluster can be on different host machines.

A **Bus destination** is a virtual location within a service integration bus, to which applications attach as producers, consumers, or both to exchange messages.

And then there is “the box” or “the host machine” which often gets confused with the server. A server is the Application Server, which is a software entity that hosts applications, whereas the host machine is the physical hardware that the server or servers are running on.



The nodes may be on separate boxes or they may be on the same box. They are shown here on separate boxes and the deployment manager is deployed on a separate machine as well.

All four servers are identical.

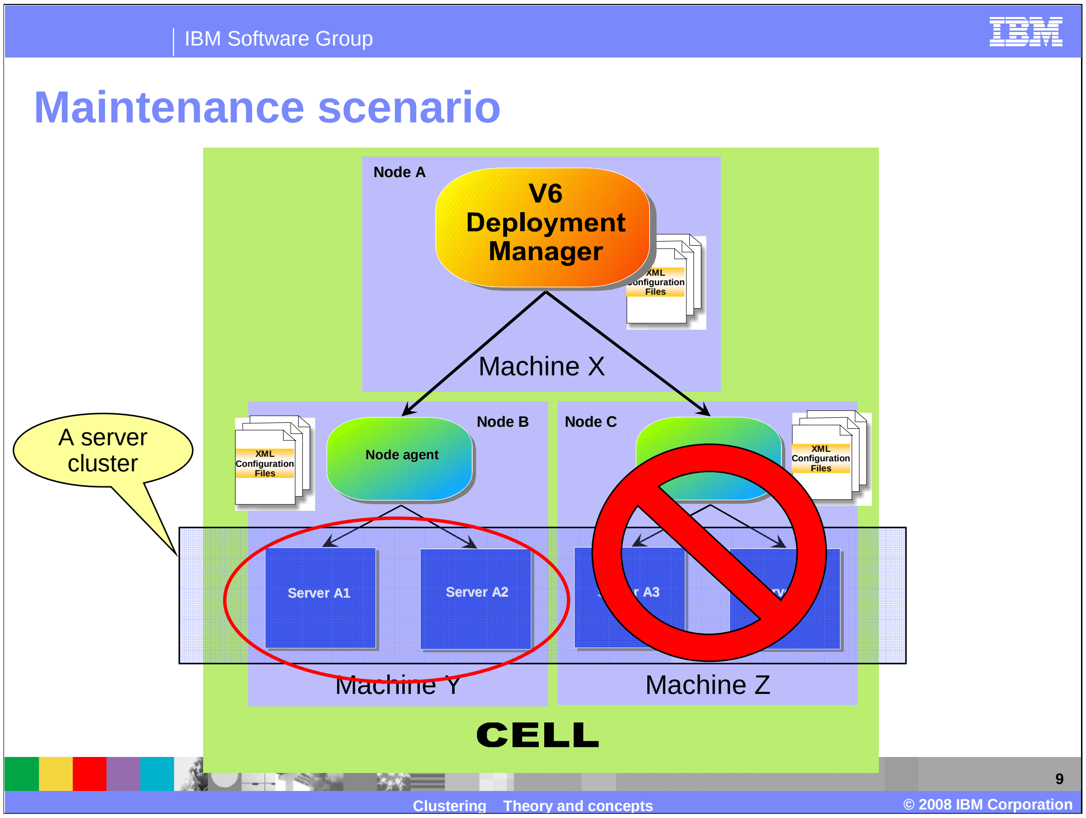
1.If Machine Z needs to be taken off-line for maintenance then Machine Y will still have 2 servers available to service requests.

2.If any one of the server experiences a problem and depending on how the system is configured, it will be possible for one of the remaining 3 to recover the work in progress.

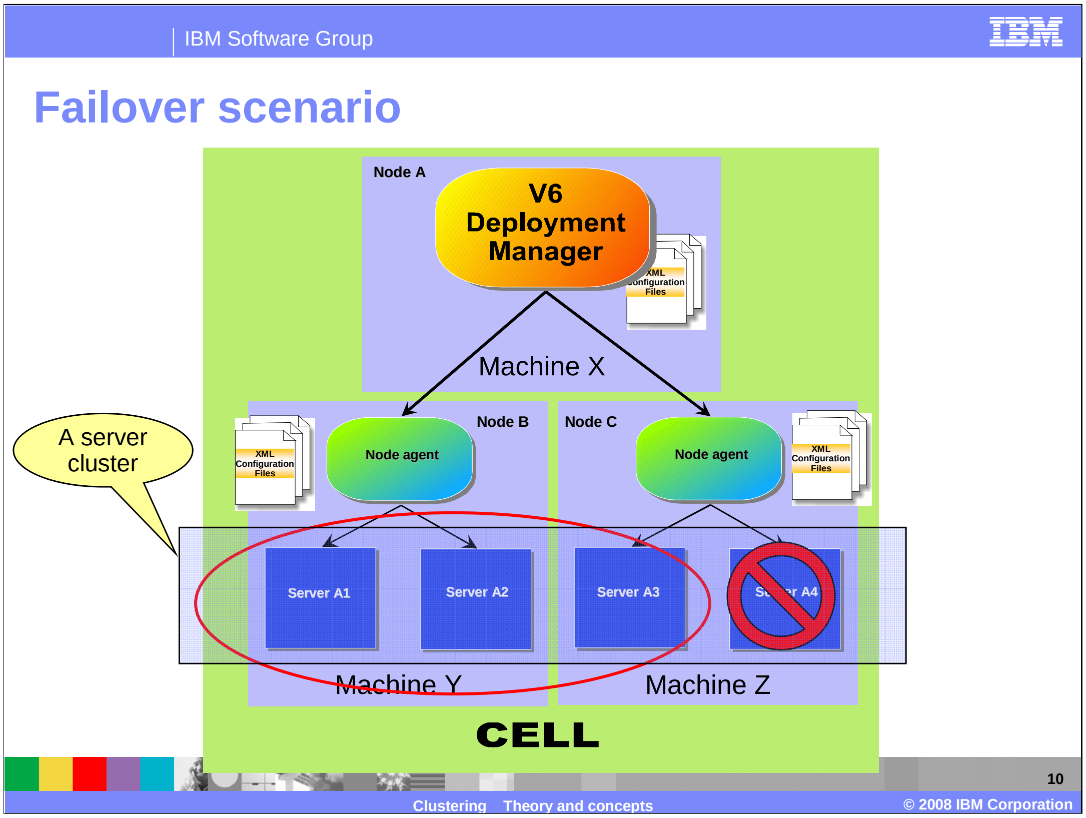
3.To add capacity more servers can be added directly to the cluster or another machine can be brought online and the new node can be federated into the Cell and more servers can be added to the cluster.

4.Adding additional application servers to a server cluster on the same machine is called *vertical* scaling. Adding additional applications to the server cluster on a different machine is called *horizontal* scaling.

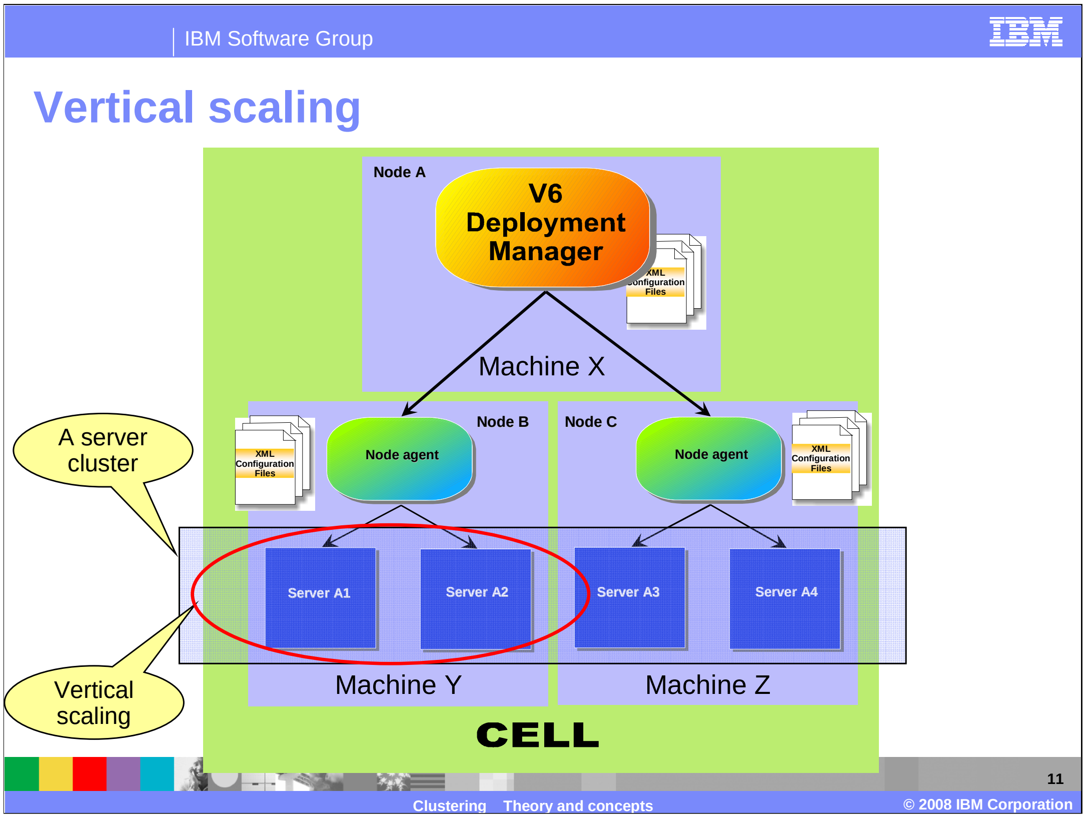
5. On z/OS, within an application server, the WorkLoad Manager will start as many servant regions as required (within imposed restraints) to process the workload and meet the defined goals. If a given server is overloaded, it is temporarily bypassed in favor of less busy servers. If a server fails, other servers take over the work and the server is recovered. When the servers are no longer needed, they are automatically stopped. This kind of automatic vertical scaling allows the full capabilities of z/OS workload management to be enabled. In the non-z/OS environment, the clones need to be preconfigured and are not dynamically created.



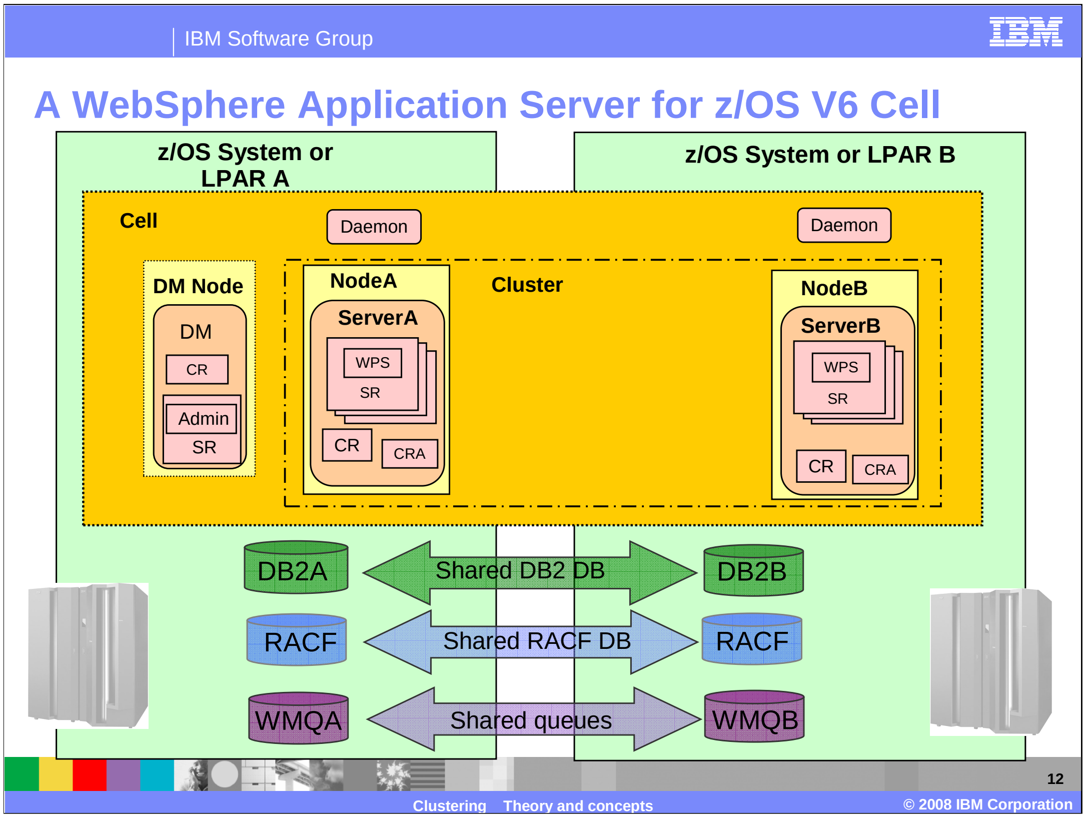
If Machine Z needs to be taken off-line for maintenance then Machine Y will still have 2 servers available to service requests.



If any one of the servers experiences a problem and depending on how the system is configured, it will be possible for one of the remaining three to recover the work in progress, that is, **failover** to one of the remaining servers in the server cluster.

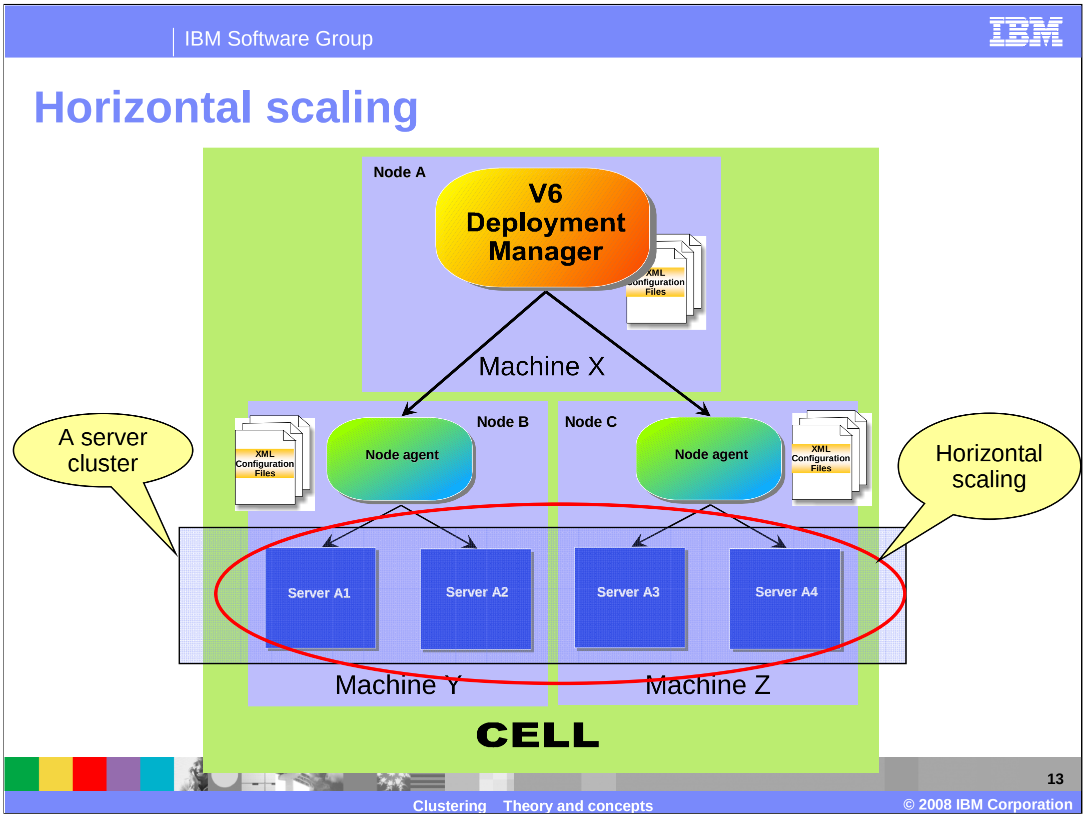


Adding additional application servers to a server cluster on the same machine is called ***vertical*** scaling

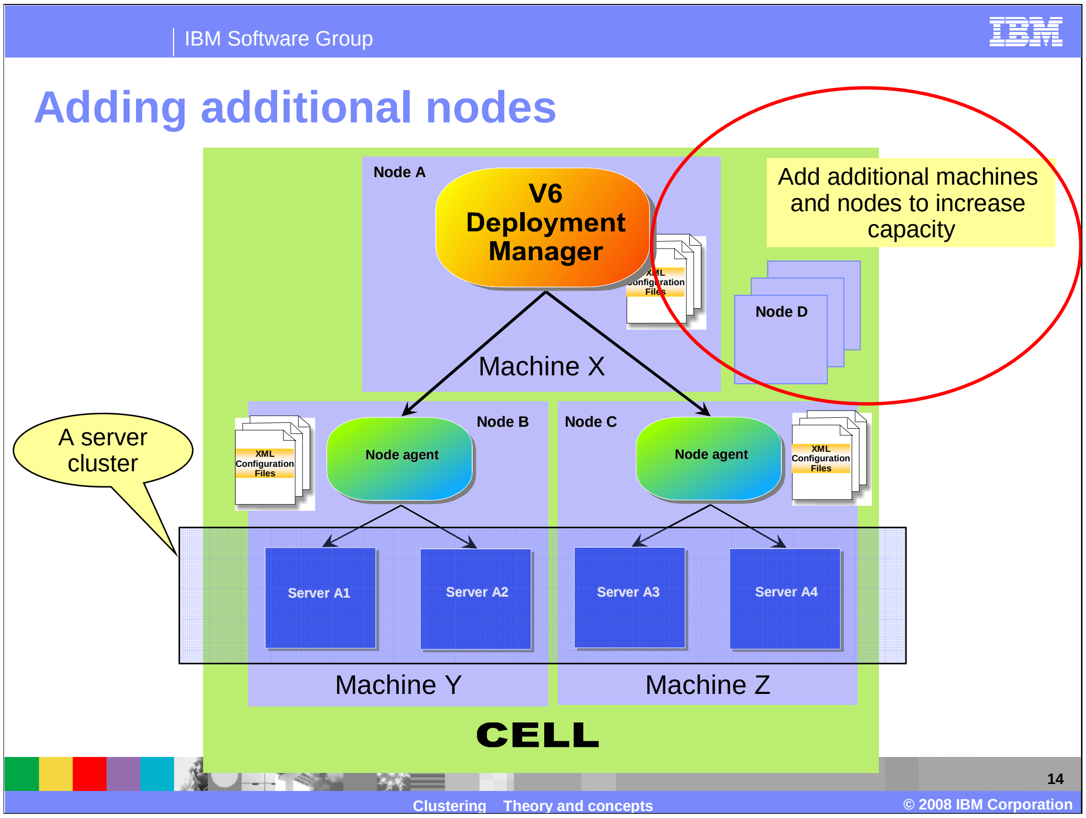


In a z/OS environment, the server instances belonging to a cluster can run on the same z/OS image, or can span more than one image in a sysplex.

Vertical clustering is actually built-in to the WebSphere z/OS Server. On z/OS, within an application server, the WorkLoad Manager will start as many servant regions as required (within imposed restraints) to process the workload and meet the defined goals. If a given server is overloaded, it is temporarily bypassed in favor of less busy servers. If a server fails, other servers take over the work and the server is recovered. When the servers are no longer needed, they are automatically stopped. This kind of automatic vertical scaling allows the full capabilities of z/OS workload management to be enabled. In the non-z/OS environment, the clones need to be preconfigured and are not dynamically created.



Adding additional application servers to the server cluster on a different machine is called ***horizontal*** scaling.



To **scale up** the capacity, one or more machines can be brought online and the new nodes can be federated into the Cell and more servers can be added to the cluster

WebSphere Process Server - Key elements

- Service integration buses
- Messaging
  - Message engine
  - Bus destinations
- Applications
- Common event infrastructure
- WebSphere process server components
  - Process choreographer, business rules, selectors, relationships
- Databases
- Enterprise service bus

Although Service Integration Buses and Messaging are part of WebSphere Application Server Network Deployment V6, they are listed here because they are key elements for WebSphere Process Server solutions.

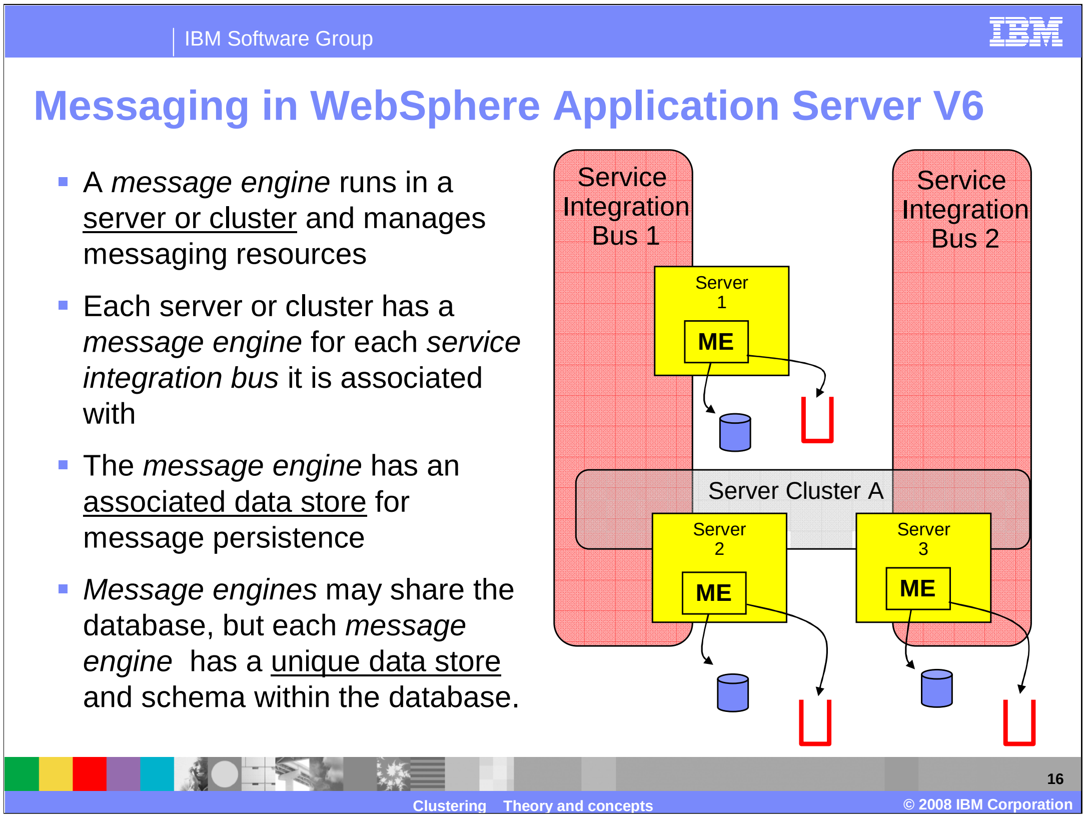
A ***service integration bus*** supports applications using message-based and service-oriented architectures. A bus is a group of one or more interconnected servers or server clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

A ***messaging engine*** is a server component that provides the core messaging functionality of a service integration bus. A messaging engine manages bus resources and provides a connection point for applications.

JMS destination is the queue

The ***Enterprise Service Bus*** is a new feature provided with WebSphere Process Server V6.0.1 that adds additional messaging and service oriented features such as complex transformations and mediations.

Messaging is at the heart of the WebSphere Process Server functionality. Its used for the common event infrastructure, business process choreography and for the asynchronous SCA invocations. This is why you need a thorough understanding of WebSphere Application Server Network Deployment V6 messaging and clustering in order to understand clustering in WebSphere Process Server. The next few slides will discuss the fundamentals of WebSphere Application Server Network Deployment V6 messaging and clustering.



Server 1 is a member of the Service Integration Bus **Service Integration Bus 1**

Server 2 and 3 are both members Server Cluster A which is also a member of the **Service Integration Bus 1** and **Services Integration Bus 2**.

A ***service integration bus*** supports applications using message-based and service-oriented architectures. A bus is a group of one or more interconnected servers or server clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

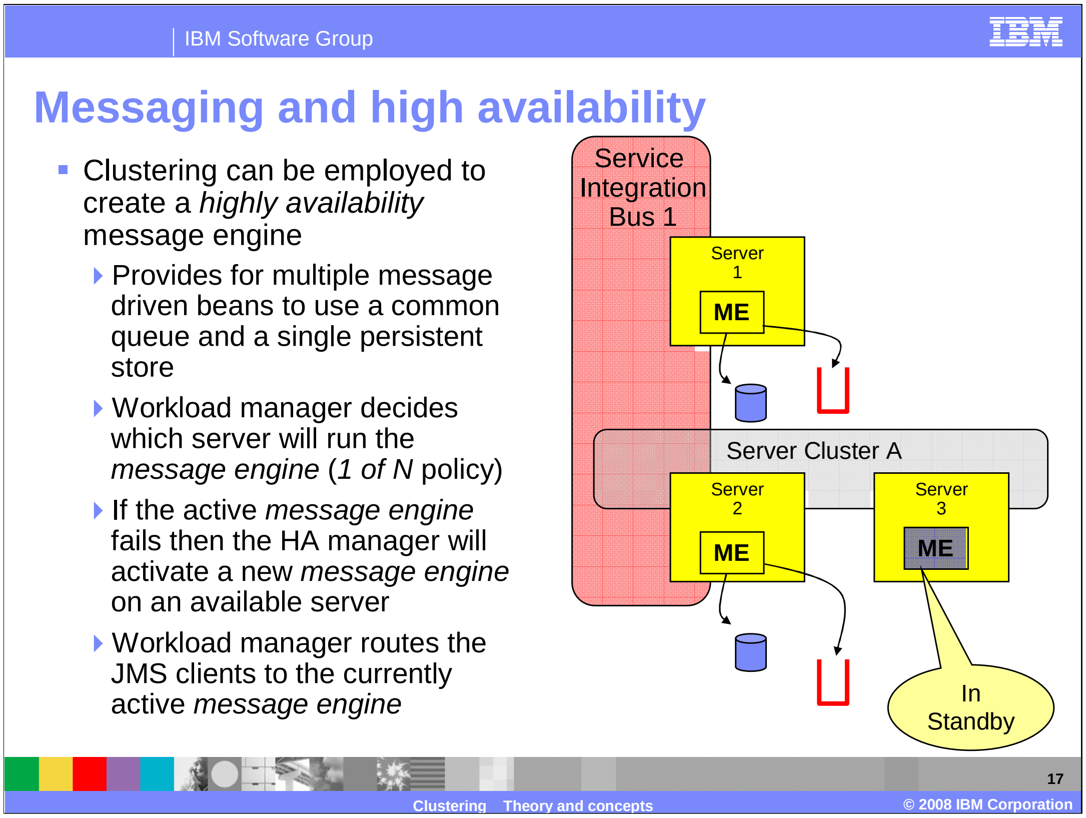
Each ***messaging engine*** is associated with a server or a server cluster that has been added as a member of a bus. When you add an application server or a server cluster as a bus member, a messaging engine is automatically created for this new member. If you add the same server as a member of multiple buses, the server is associated with multiple messaging engines (one messaging engine for each bus).

Bus members

The members of a service integration bus are the application servers and server clusters within which messaging engines for that bus can run.

Note:

The are two different kind of aggregations, the collection of servers (and server-clusters) that are part of (members) the Service Integration Bus and then there is another collection of servers that comprise the server cluster. A ***cluster member*** will refer to the relationship between a server and the server cluster and a ***bus member*** will refer to the relationship between the server or server-cluster and the service integration bus.



When an application server cluster is added as a member of the service integration bus the message engine is created for the cluster using the active/standby pattern by default.

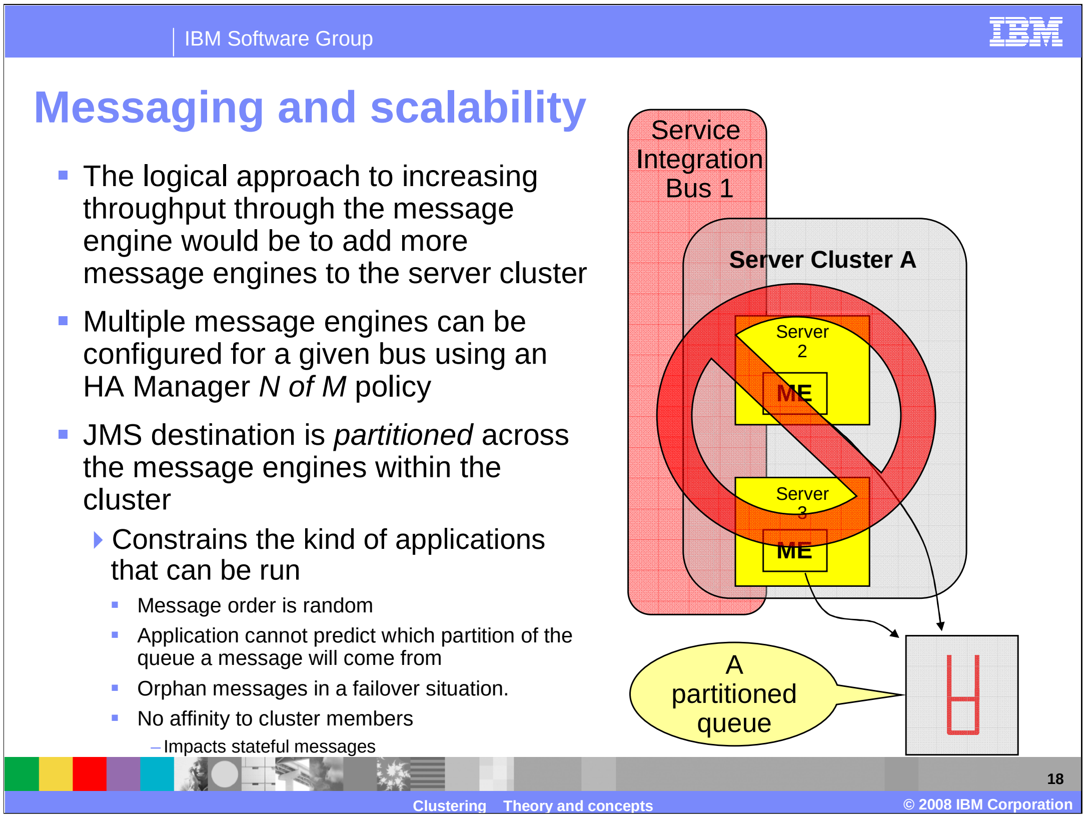
This configuration is appropriate when the goal is *high availability*, and when there must always be a messaging engine available. This capability is also referred to as *failover*.

The drawback with this configuration is that there is only one queue and database table for the cluster. This could become a bottleneck. If this happens, the only way to get more throughout is to add a server on a faster computer with more memory.

It is possible to create a server cluster that has computers with different levels of service, some may be faster than others. In this situation the server with the highest capacity can be designated as the primary and the other servers can be the standbys.

Sidebar:

When a cluster has servers that have different capacities, this is referred to as a mixed configuration. The applications running on the servers must all be the same but the configuration information can be unique for each server.



When you consider how to increase messaging throughout, it might seem that you should add more message engines to the server cluster.

The drawback with this approach is that it places constraints on the kind of applications that can be run.

•When a user of the queue reconnects to the queue it may be routed to a different partition, so messages may not be processed in the order that they were placed on the queue.

•The message consumer will not be able to determine which partition a message is coming from, therefore it will need to retrieve the messages from all of the servers.

•The order of the messages is indeterminant.

•An application may not rely on the assumption that a message that has been put to the queue will still be there on the next connection.

•When a cluster member fails, the messages in the associated queue will not be available until that server comes back online; they are orphaned.

Because of the constraints imposed on the application, this topology is not recommended for general use.

Note: the pub/sub configuration is a special case where affinity to the queue can be obtained, but this is not applicable in the more general case used by the WebSphere Process Server components.

IBM Software Group

Messaging and clustering

- WebSphere Process Server V6 requires the use of the messaging infrastructure
  - Recommended practice is to use the *default JMS message provider* for all the messaging needs in WebSphere Process Server
    - SCA, business process choreographer and common event infrastructure
- The next issue to consider is the relationship between the messaging engines and the applications that will be using them.
  - On z/OS, applications and messaging engines in the <u>same cluster</u>
    - Two possible configurations
      - Single MDB active at a time, (Message engine in active/standby)
      - Multiple message engines, 1 per server (active/active) results in the partitioned queues
  - Other platforms, applications and messaging engines in <u>separate clusters</u>
    - Highly available message server

19

Clustering   Theory and concepts

The default JMS message provider is the one that comes with WebSphere Application Server V6.

It is possible to use other JMS providers for Business Process Choreographer and the Common Event Infrastructure. Since the Service Component Architecture requires the default JMS message provider it is recommended that the default JMS message provider be used for the Service Component Architecture and the Common Event Infrastructure requirements as well.

Assuming that the applications are to be clustered.

In a distributed environment, grouping the applications and the message engine into the **separate clusters** allows for greater flexibility in configuration and tuning and is the **recommended approach**. However, on z/OS, the **recommended approach** is to group the applications and messaging engines in the **same cluster**.

When you consider the case with the applications and the message engine in the <u>same cluster</u> there are two alternatives to consider.

1.Active/Standby

- Is the default provided when a cluster is added to an service integration bus as a bus member.
- There is one message driven bean per message engine and only one message engine is active at a time

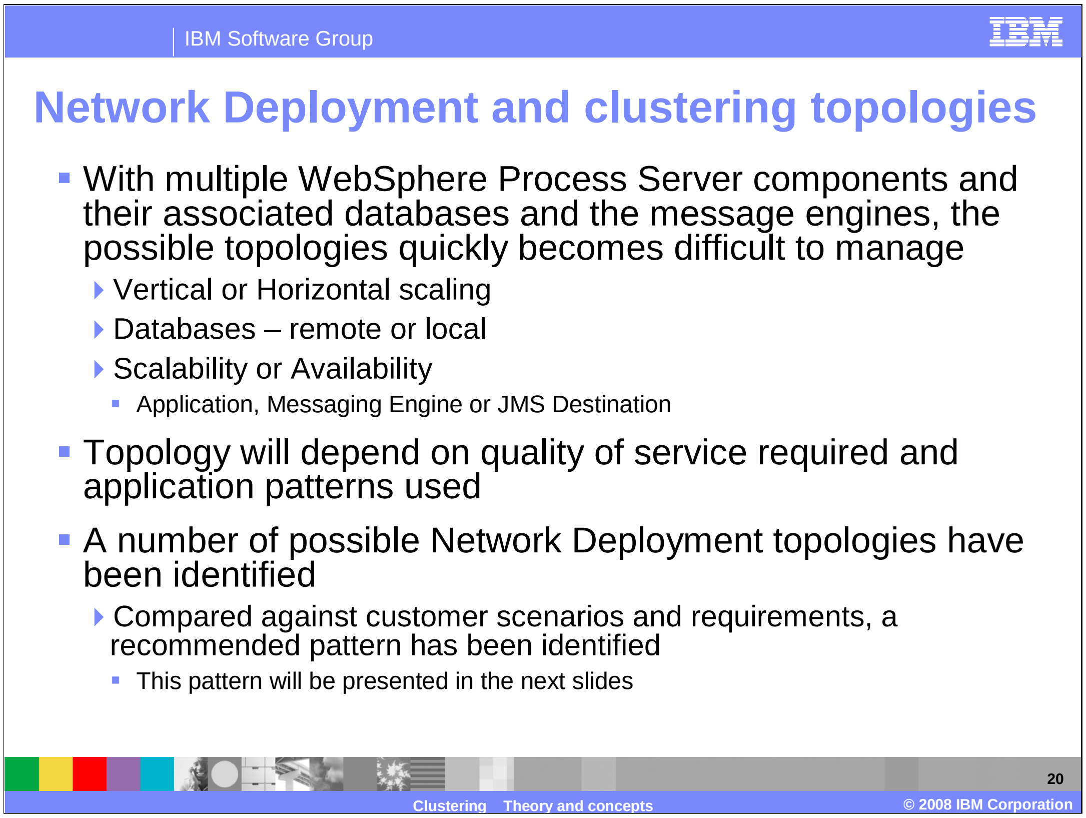
•2. Active/Active

- Having multiple message engines per cluster will result in partitioned queues

Either of these configurations work on z/OS.

On other platforms, keeping the message engine in a separate cluster will create a highly available message server.

Separating the message engine and the applications is the recommended approach for achieving *high availability*.



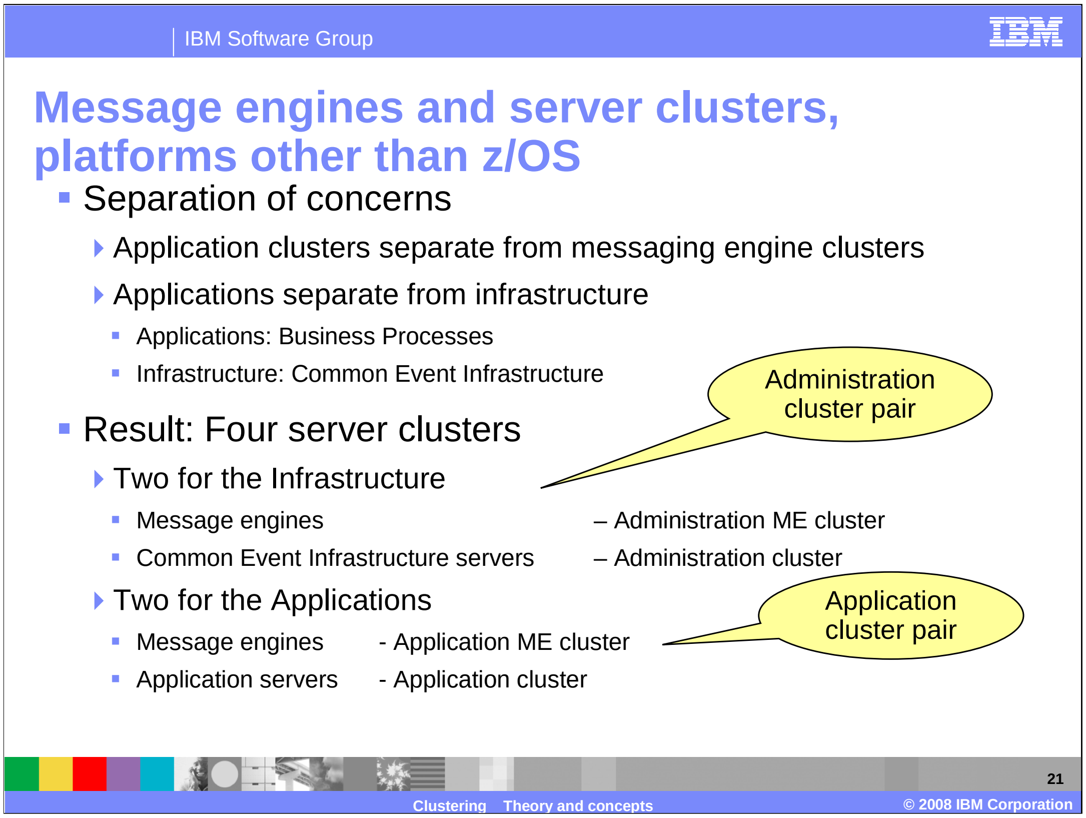
There are many facets to consider when deciding how to configure the Cell topology. Is the goal high availability or scalability or some combination of both? If so, which one is most important based on the requirements of the application?

Will the databases be remote or local?

Will there be several databases or will tables be combined into one or two databases, where it is feasible?

Ultimately it will depend on the administrative processes already in place and the quality of service required by the applications. There are many ways to configure the cell topology and WebSphere provides the flexibility to do what is needed.

Based on typical customer requirements for scalability and high availability, a ‘suggested’ topology will be presented as a vehicle for discussing how to configure a complex WebSphere Process Server topology.



The topology being presented here is recommended as the most likely topology for non-z/OS environments based on the quality of services available to the applications. It will be used to demonstrate the steps for setup and configuration of a clustered topology. Once you understand the principles and the steps, you can easily develop alternative solutions.

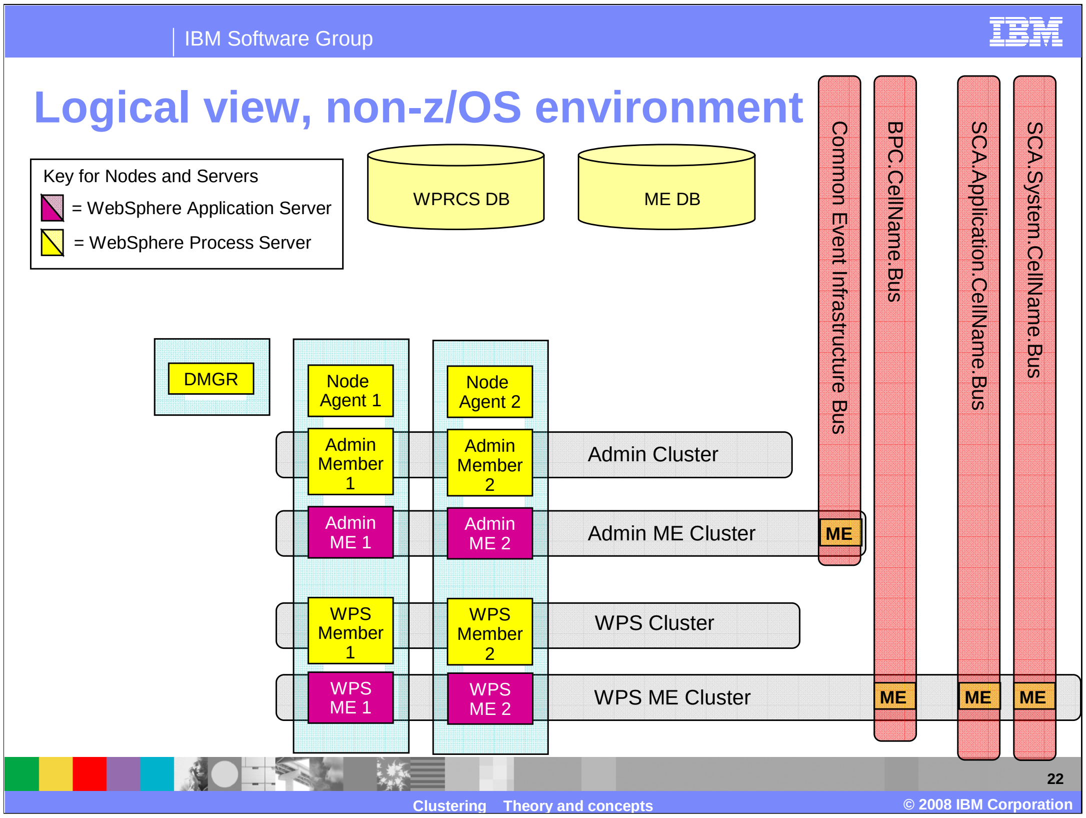
The Common Event Infrastructure will likely be an integral part of all WebSphere Process Server applications and it is therefore considered to be an infrastructural component of the system.

The principle of “Separation of Concerns” is used as a guide for deciding how to partition the system and distribute the function.

Based on the previous discussion regarding the message engine and the applications, there are message engine clusters and application clusters.

Considering functional boundaries, you can partition the system based on whether it is an administrative function or functionality associated with the end-user application. An example of an administrative function would be the Common Event Infrastructure (CEI) or the Business Rules Manager. Both of these are WebSphere Process Server components and therefore part of the overall infrastructure.

Using this criteria, the resulting topology starts with four server clusters. As the system grows there may be additional application cluster pairs, but only one administration cluster pair would be necessary.



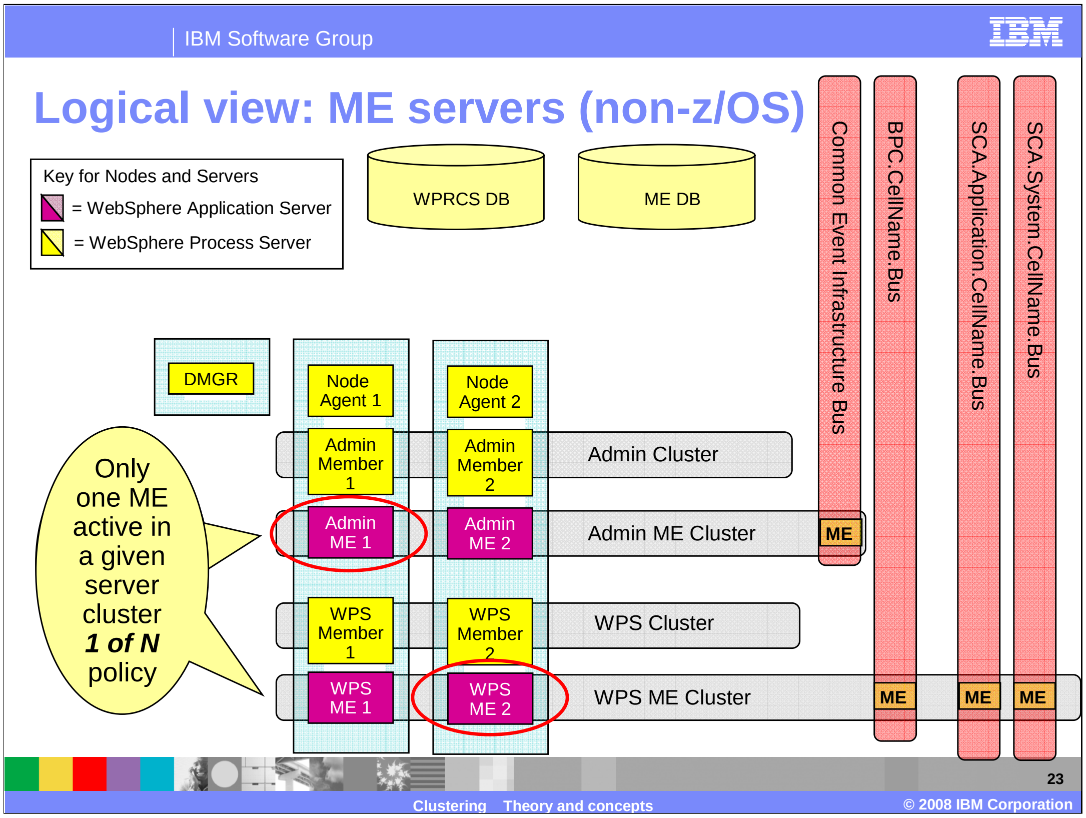
Notice the key in the upper left. There are two different kinds of servers in each node. The three nodes, deployment manager, Node 1, and Node 2, are represented by the light blue rectangles.

The service integration buses are represented by the salmon colored vertical columns on the right and the server clusters are the gray horizontal bars, enclosing the application servers and sometimes intersecting with the Service Integration Buses.

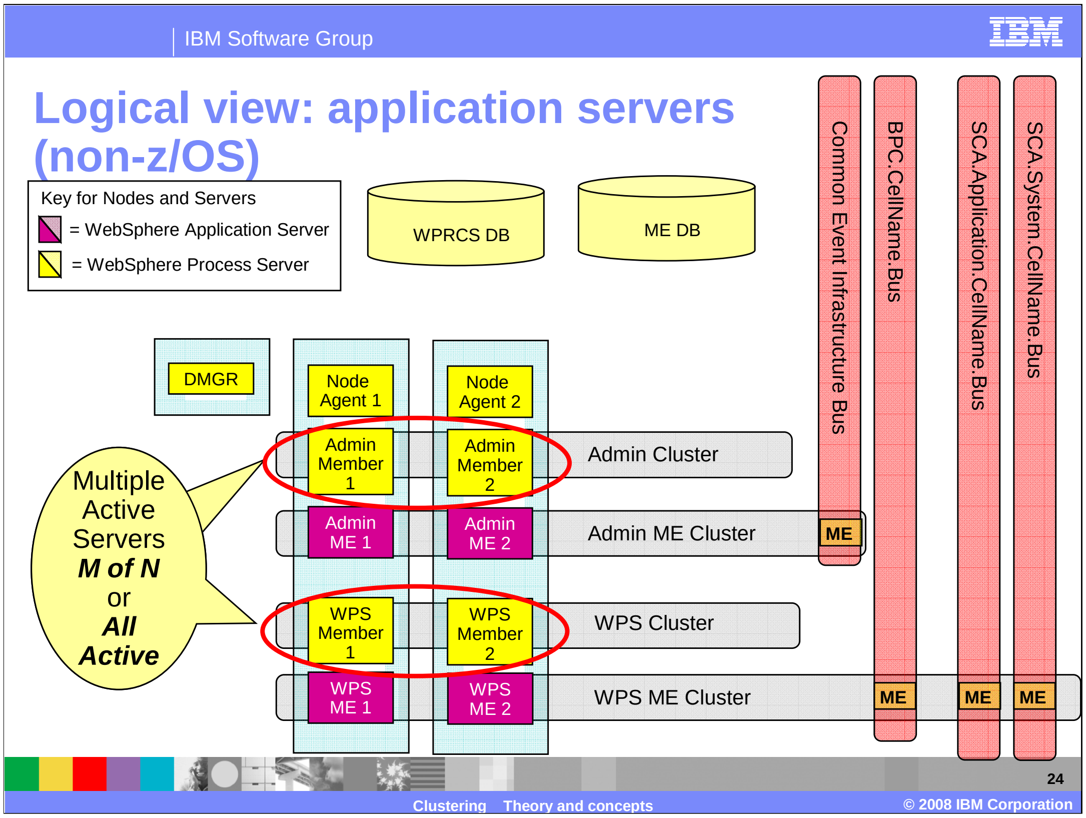
The Common Event Infrastructure Servers in the AdminCluster are associated to the message engine in the AdminMECluster through the JMS destination configurations.

Likewise the Application Servers in the WPSCluster are associated to their messaging engine in the WPSMECluster through their JMS destination configurations.

The **ME** (Message engine) shown on the Service Integration Buses represent the association between the server cluster and the Service Integration Bus.



Only one message engine is active in a server cluster at a given time. The WebSphere Workload Manager decides which message engine is active based on the HA manager policy which is configured as 1 to n.

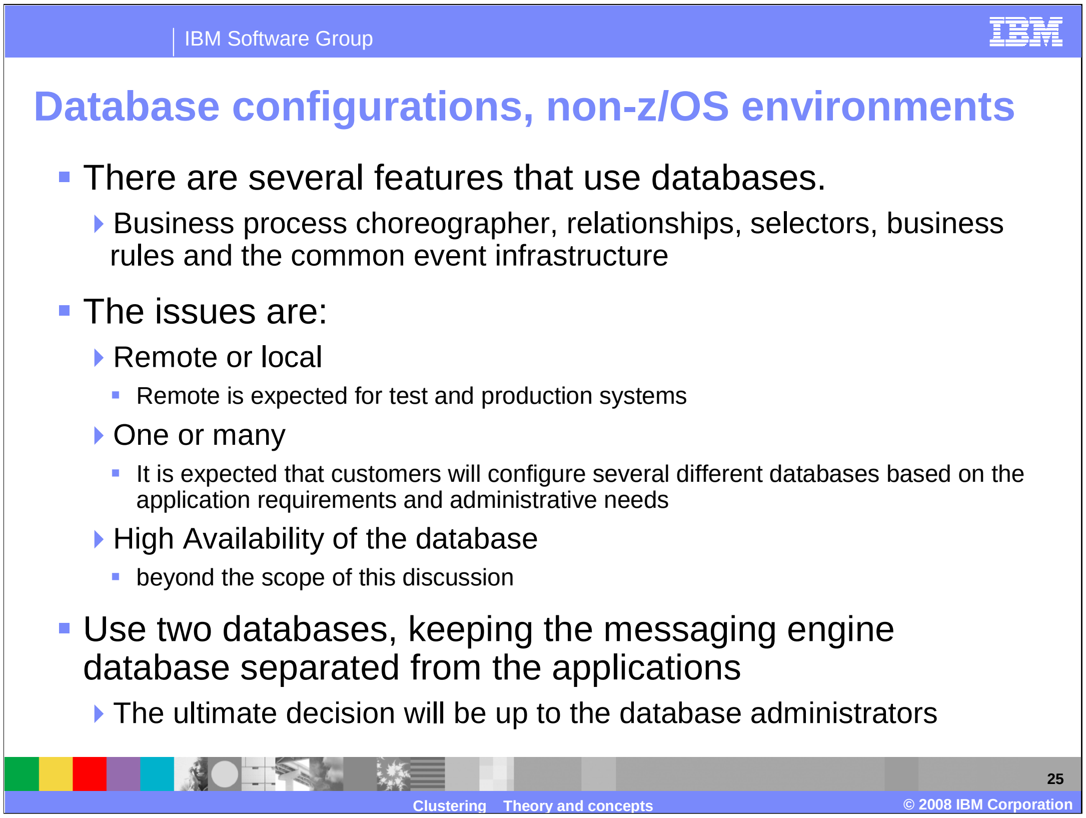


Whereas in the application clusters the servers are configured so that they can all be active at the same time.

The SCA System and Application Buses are required for the basic SCA functionality.
The BPC Bus is required for Business Process Choreography
The Common Event Infrastructure bus is required for CEI and monitoring.

The Nodes can be co-located on the same ‘box’ or distributed horizontally to different ‘boxes’.

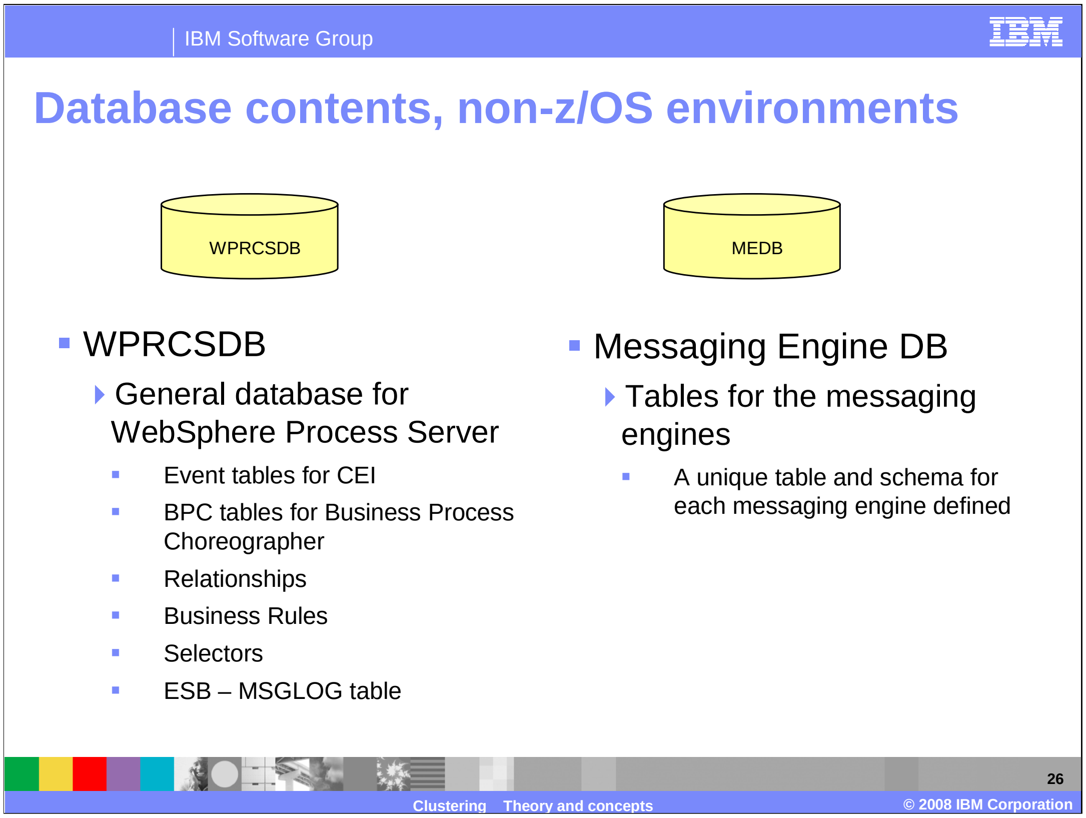
The databases are discussed after a few more slides.



With WebSphere Process Server there are quite a few databases introduced.

In a typical production environment there will be several databases, the ESBLogMedDB, the EVENTDB, the BPEDB, the WPRSCDB, the MEDB, along with the end-user application databases. It is expected that they will be remote, residing on database server, or managed by database administrators.

For the configuration being presented there will be two databases. The tables for the EVENTDB, ESBLogMedDB and BPEDB are created in the WPRCSDB, keeping the message engine data stores isolated in the MEDB. This decision is made for convenience only.



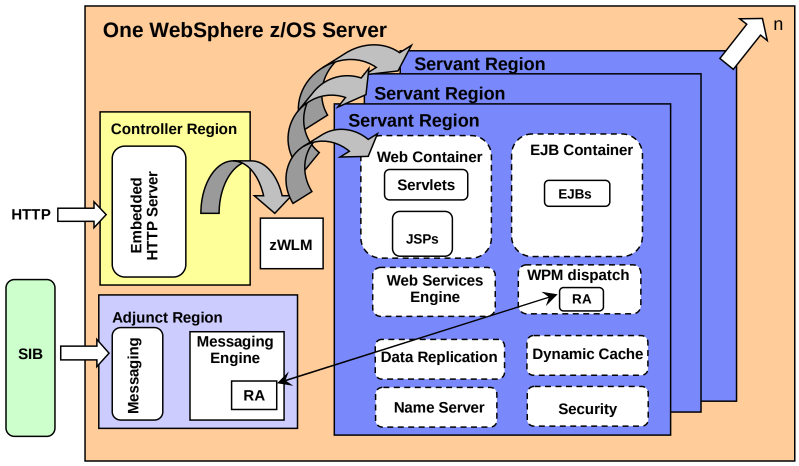
Note: create the WPRCSDB before creating the Deployment Manger node (DMGR)

The decision to use two databases rather than many simplifies things for demonstration purposes.

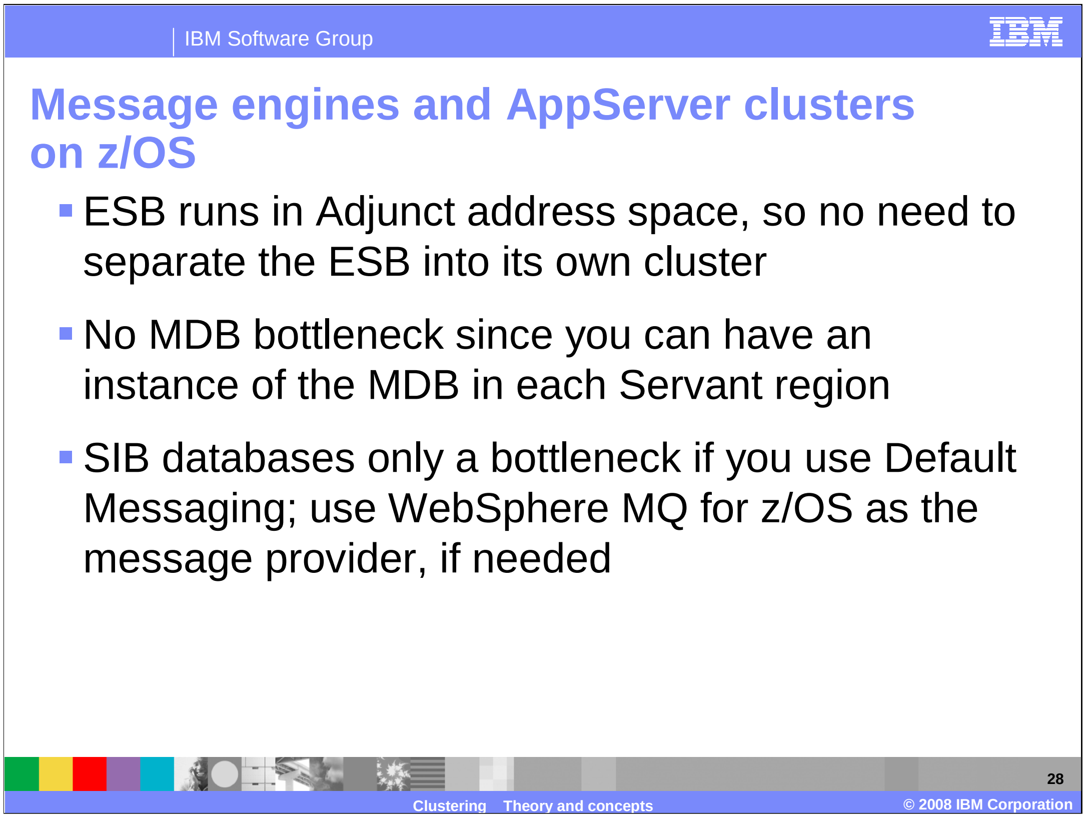
Creation of the databases and tables for non-z/OS environments will be covered in detail as part of the tutorial.

WebSphere Application Server structure on z/OS

- WebSphere on z/OS uses a different structure for a 'server' than other platforms



The structure of a 'server' in WebSphere z/OS shown here is different than that of a 'server' in WebSphere Application Server on other platforms. Some of the recommended topologies you can read about in connection with non-z/OS platforms may not be necessary on z/OS. As you can see here, in WebSphere Application Server for z/OS, *one Application Server* is comprised of a *Controller* region, an *Adjunct* region and one or more *Servant* regions. Each z/OS Application server is Clustered by default without additional configuration.



You might read that one reason for defining the enterprise service Bus (ESB) in its own cluster is to remove it from the JVM in which process server and your applications are running. Since the least reliable part of the infrastructure is the JVM where applications are running, it makes sense to put an infrastructure like the ESB in its own cluster. This also frees up heap in the JVM for the applications.

In WebSphere Application Server for z/OS, however, *one application server* consists of a *controller* region, an *adjunct* region and one or more *servant* regions as you just saw on the previous slide. The service integration buses (SIBs) that make up the ESB run within the adjunct address space so the messaging engines are already in a JVM that is separate from the JVMs used by the applications that run in the servant regions. Therefore, availability or JVM heap limitations are probably not reasons for splitting the ESB into its own cluster.

Another possible bottleneck in a non-z/OS WebSphere configuration is the message driven Bean (or MDB) that service the ESB, since only one of these can run per server in WebSphere on non-z/OS platforms. In WebSphere Application Server for z/OS, however, you can have an instance of that MDB in each servant region. Therefore, the possible MDB bottleneck should not exist in WebSphere application server for z/OS.

Another potential bottleneck is the DB2 database that supports the ESB. When using the Default Messaging implementation of WebSphere Application Server for z/OS, the queues are implemented as tables in DB2. A SIB runs in one Adjunct, and when you have a cluster, the adjuncts of other cluster members can takeover if the active SIB fails for any reason.

However, the SIB databases are only a bottleneck if you use the default messaging that comes with WebSphere Application Server. In order to create a highly scalable configuration for messaging on z/OS, you could use WebSphere MQ for z/OS as the message provider. In a parallel sysplex with shared queues there should be no bottleneck in putting or getting messages to queues.

Note: This is only true for MDB implementations. When using SCA , many queue destinations are automatically created and managed on your BUSes. It would be difficult to provide the equivalent functionality through manual processing. Having separate bus's can alleviate the bottleneck, as each bus has it's own DB instance.

Workload balancing using WLM classification on z/OS

- Use workload classification document
  - XML file called *WebSphere for z/OS Workload Classification Document*
  - Specified to server with WebSphere variable **wlm_classification_file**
  - Can be set at the Cell, Node or Server Instance level
- Classify inbound HTTP, IIOP and MDB work requests and assign to a transaction class (**TCLASS**)
- Workload Manager (WLM) uses TCLASS value to assign service class to each incoming request
- For Mediation work, use **SiBClassification** elements and z/OS Workload Manager **TCLASS** value of "**SIBUS**"
- For JMS activation specification to be used by MDB applications with the version 6 default messaging provider, replace any related **InboundClassification type="mdb"** classifications with **SibClassifications type="jmsra"** classifications

You can classify inbound HTTP, IIOP, MDB, and JMSRA work in a common workload classification document. You might have configured classification documents with a previous release using a different format, and specified these files using an advanced setting in the Web container or through the endpoint_config_file custom property. If you configure the new document that classifies HTTP, IIOP, MDB, and JMSRA work in one file, the new file overrides the old files. For example, if you have HTTP classification rules in the new workload classification document, these rules are used instead of other rules that you might have in the old style HTTP classification document. However, if your new document specifies only classification rules for EJB 2.0 message-driven beans deployed with listener ports, the old style HTTP classification document is still used.

A workload classification document file is an XML file in which you classify incoming HTTP, IIOP, and message-driven bean (MDB) work requests and assign them to a transaction class (TCLASS). The TCLASS value, if it is assigned, is passed to the MVS Workload Manager. WLM uses the TCLASS value to classify the inbound work requests and to assign a service class or a report service class to each request.

If you want to classify work for message-driven beans deployed against JCA 1.5 resources with the default messaging provider, or you want to classify mediation work for use with service integration buses, you need to define a Classification element that uses SibClassification elements. You must also perform z/OS Workload Manager actions that are required to use the default TCLASS value "SIBUS". If you replace any listener port with a JMS activation specification for use by MDB applications with the version 6 default messaging provider, you should replace any related InboundClassification type="mdb" classifications with SibClassifications type="jmsra" classifications.

Business rules, selectors and relationships

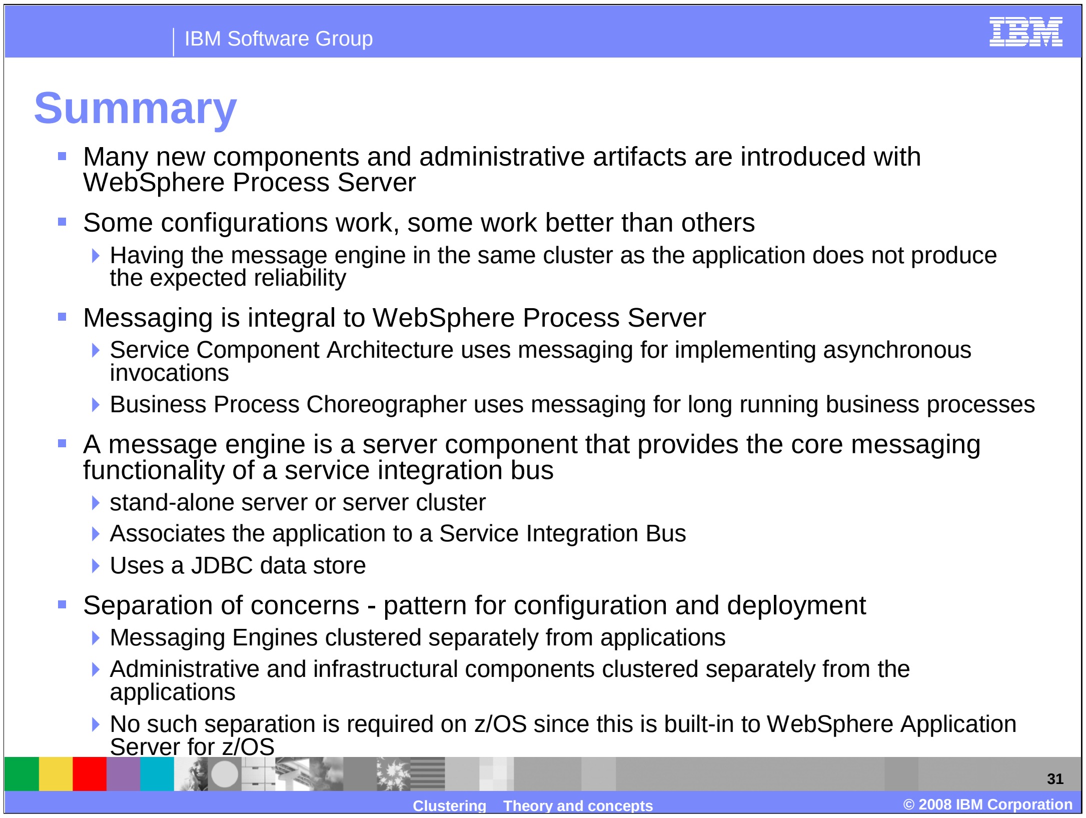
- Business rules, selectors and relationships make use of the global WPSRCDB
- SCA component modules are deployed as J2EE EARS
- SCA module can only be deployed once in a cell
- As global entities there are additional considerations
  - Uninstalling does not remove potentially shared objects from the database
  - Unique names and namespaces are required across the cell
- WebSphere Adapters
  - Adapters that use inbound traffic can only be deployed once in a cell

Here are a few considerations regarding the use of other service component architecture components, which can impact the design of the cell, the databases and application server clusters.

The global nature of the WebSphere Process Server components presents challenges when deploying and redeploying applications.

The potential for name clashes prohibits duplicate names in many situations and the use of common components means that an entity might not be removed during uninstallation as expected, causing a deployment failure on the next deployment cycle.

WebSphere adapters that are used for inbound traffic can only be deployed once in a cell.



Moving to WebSphere Process Server ND V6.0.2 introduces a lot of new parts and pieces to consider and work with.

There are many combinations for deployment and configuration, some work, some work better than others and some just don't work in a realistic manner. The recommendations for deployment and configuration are different for z/OS and non-z/OS environments, based on the differences in the way WebSphere Application Server is structured in the different environments.

The key element in business process applications and all WebSphere Process Server applications is messaging. Messaging lies at the heart of everything and when it comes to creating applications that leverage the WebSphere clustering capabilities the message engine is the component that must be considered first.

The deployment pattern presented here for non-z/OS environments has 2 features. First separating the message engine cluster from the application cluster. This is imperative for reliable messaging. The second feature presented as part of this pattern is the separation of the administrative or infrastructural components of the system from the user application components. This second feature is optional and may not be necessary in all cases.

For z/OS environments, separating the messaging engine into its own cluster is unnecessary. Using WebSphere MQ for z/OS as the message provider is recommended. Also, there is no need to separate the administrative components from the application components, because WebSphere Application Server on z/OS has this separation built-in. The Service Integration Buses (SIBs) that make up the ESB run in a separate address space from the Messaging Engines.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WPIv602_ClusteringConcepts.ppt

This module is also available in PDF format at: ../WPIv602_ClusteringConcepts.pdf

You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

Cloudscape DB2 WebSphere z/OS

A current list of other IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

J2EE, Java, JDBC, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.