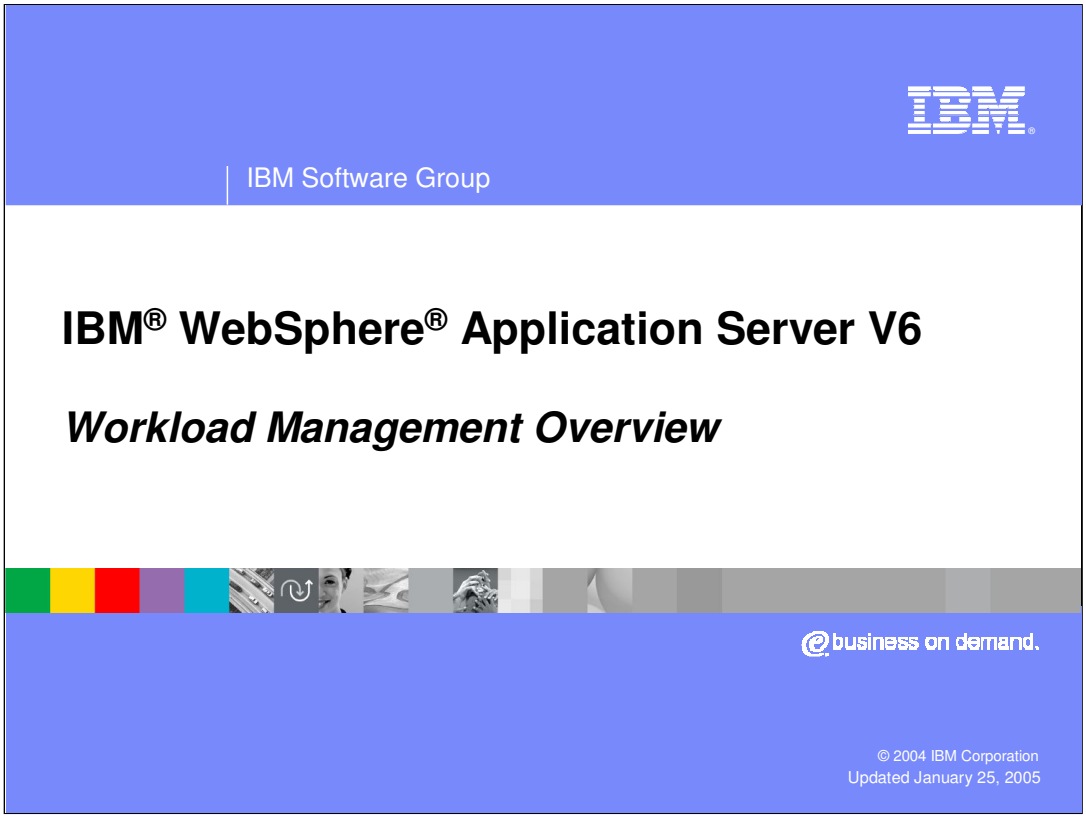


This presentation will focus on an overview of Workload Management

Goals

- Cover the following topics
  - Define Workload Management (WLM)
  - New V6 WLM functions
  - Describe clusters and cluster members
  - Examine the topology
  - Discuss weighted servers
  - Determine failover scenarios
- Covered by High Availability (HA) presentation
  - Failover of singleton services

The goal of this presentation is to define and explain Workload Management. Related High Availability information can be found in separate presentations.

Agenda

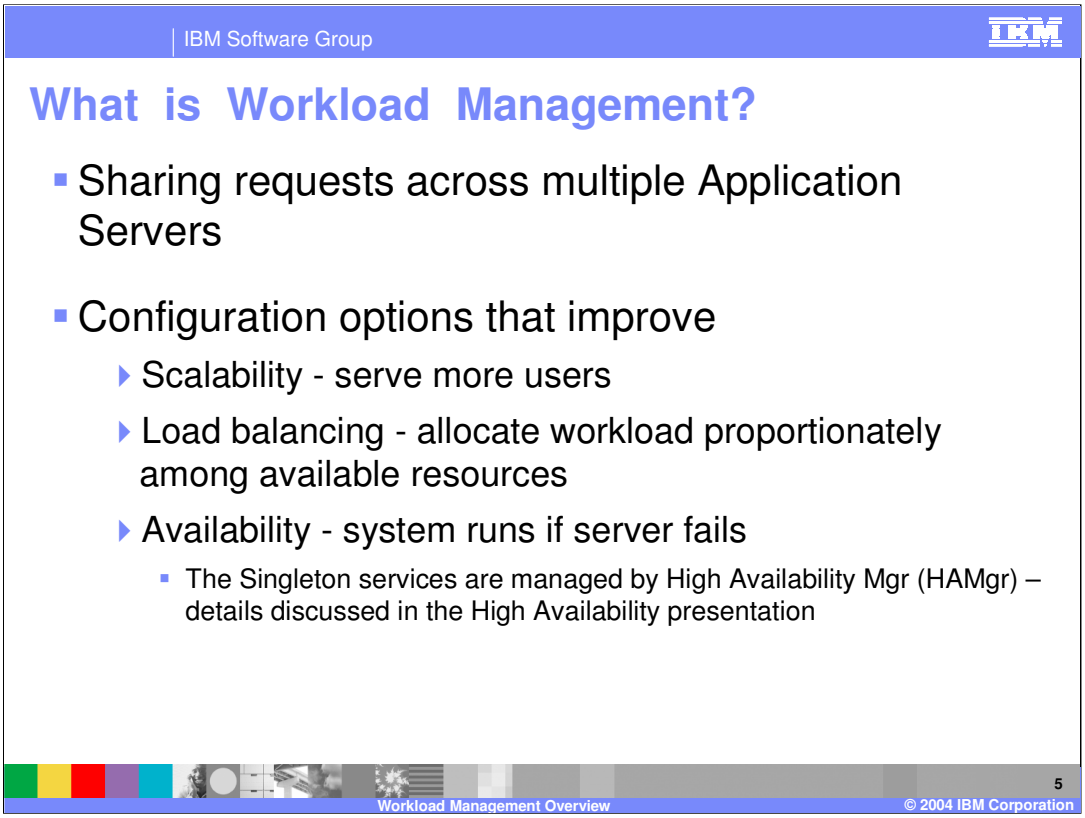
- Overview
- New V6 functions
- Cluster
- Request routing
- Failover

The agenda for this presentation is to start with an overview, then introduce the new features of V6.

Section

Overview

This section is the overview.

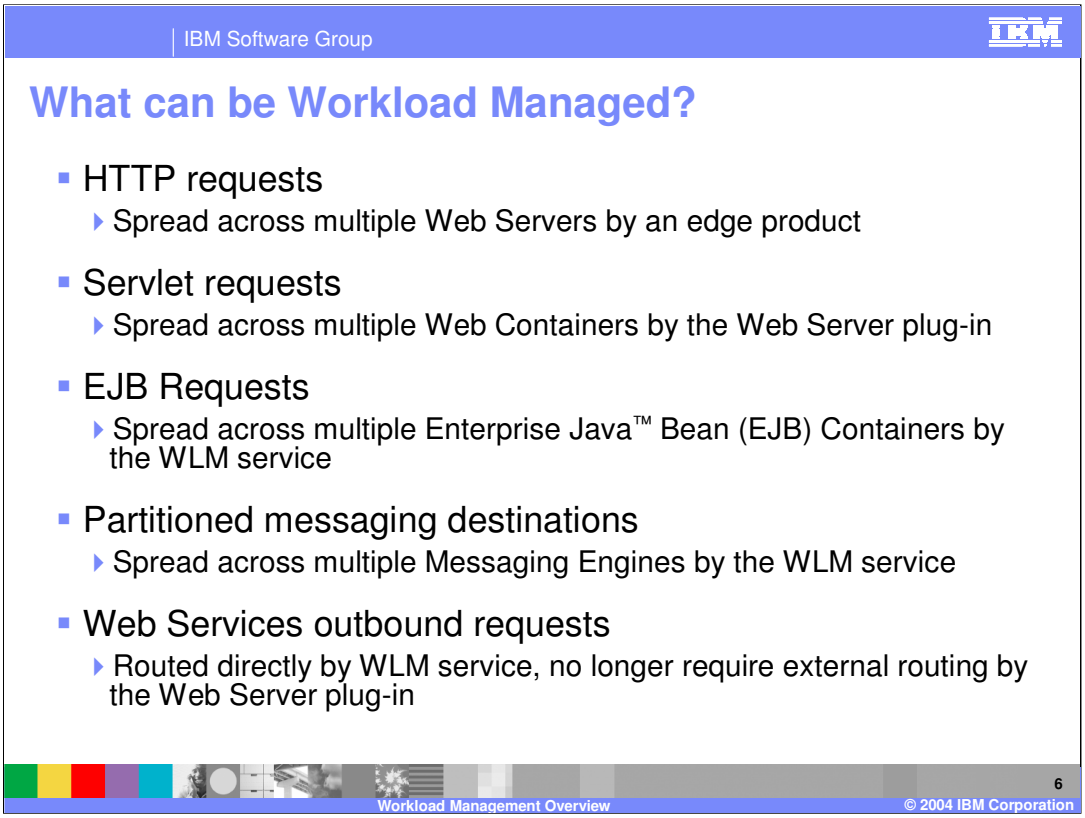


Workload management optimizes the distribution of client processing requests.

Incoming work requests are distributed to the application servers, enterprise beans, servlets, and other objects that can most effectively process the requests.

Workload management also provides failover when servers are not available, improving application availability.

In the WebSphere Application Server environment, workload management is implemented by using Clusters of application servers.



Several types of requests can be workload managed

HTTP Requests can be shared across multiple HTTP Servers.

This requires a TCP/IP sprayer to take the incoming requests and distribute them. There are both hardware and software products available to spray TCP/IP requests. Network Dispatcher is a software solution that is part of the WebSphere Edge Server. Network Dispatcher applies intelligent load balancing to HTTP requests.

Servlet Requests can be shared across multiple Web Containers.

The WebSphere Plug-in to the HTTP Server distributes Servlet requests.

Web Containers can be configured on the same machine or multiple machines.

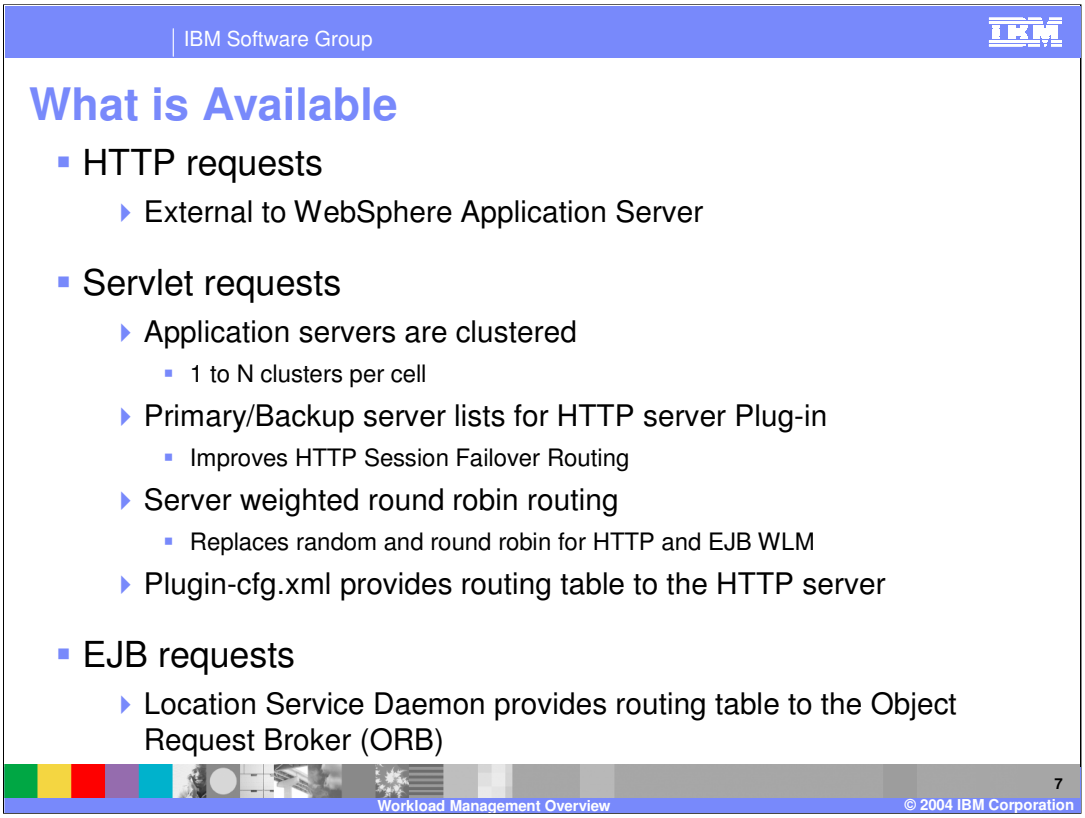
EJB Requests can be shared across multiple EJB Containers.

The Workload Management Plug-in to the Object Request Broker (ORB) distributes EJB requests.

EJB requests can come from Servlets, Java client applications, or another EJB.

Partitioned messaging destinations utilize workload management to distribute messaging workload across multiple Messaging Engines.

Web Services outbound requests are now routed directly, and no longer require an external process, such as the Web Server plug-in to act as an intermediary.



As mentioned previously, HTTP Requests are workload managed externally to WebSphere Application servers.

For Servlet requests, you can configure multiple Clusters in a cell, multiple cluster members within a cluster, as logic dictates for a given scenario. The HTTP server plug-in reads a list of servers it can route to from the plugin-cfg.xml file. In the unlikely event that ALL the servers fail to respond, the plug-in also has a back-up server list to route to.

Each of those servers also has an associated weight, which will be discussed momentarily. The routing option is a weighted round robin.

EJB Requests can also be routed among EJB containers. The Location Service Daemon process is responsible for the routing table, which can have entries for servers in other clusters.

Section

New V6 WLM Functions

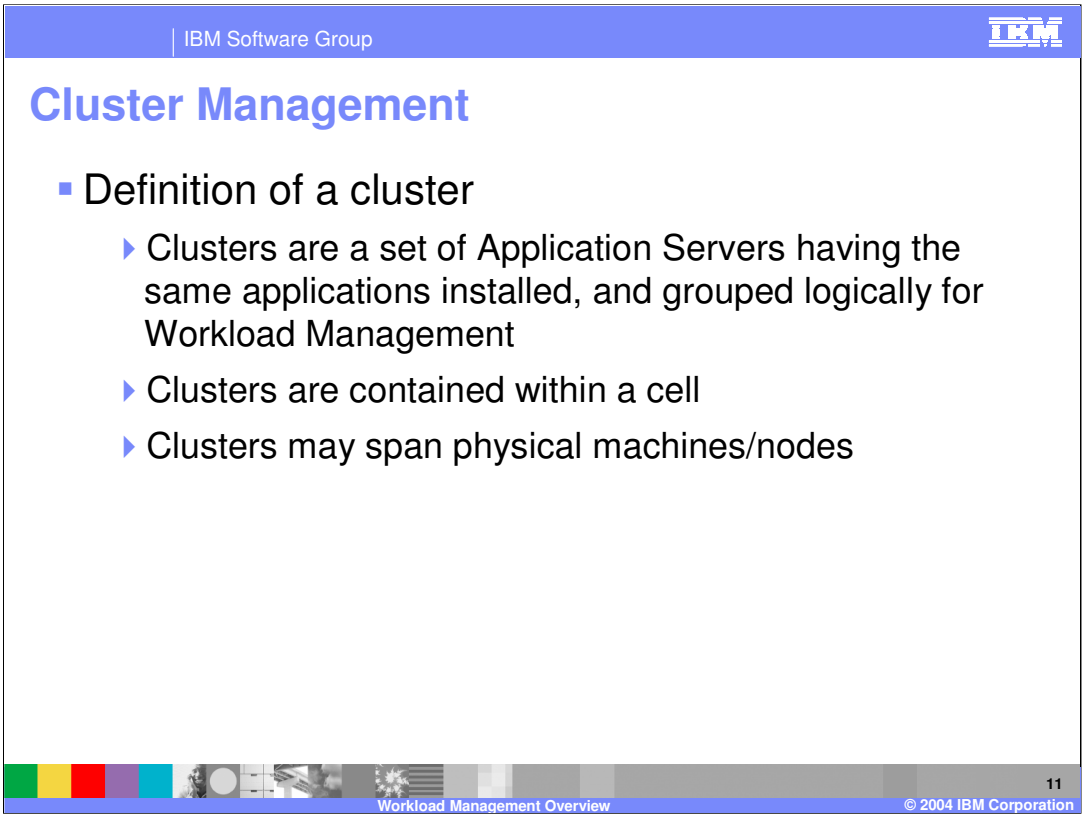
This next section includes the new workload management functions in V6.

Unified Clustering Framework

- Common clustering logic across different resources that require clustering
  - The view and use of clusters is administered in a unified and consistent manner for all protocols (HTTP, EJB, JMS, and others)
- New WLM functions can be implemented once for all protocols
- High Availability
  - Makes WLM routing a highly available service, which makes cluster and routing information always available
  - Does not depend on the Deployment Manager

Failover of Stateful Session EJBs

- Uses the Data Replication Service, similar to HTTP session failover
- Can be enabled on a per-Application basis
- WLM will fail beans over to a server that already has a copy of the session data in memory if possible
- Ability to collocate stateful session bean replicas with http session replicas with hot failover
  - J2EE 1.4 spec requires HTTP session state objects to be able to contain local references to EJBs



By default, you can only install one copy of the application server binaries on a machine. Once those binaries are installed, you can have multiple application servers configured - the data needed for each additional server is stored in several XML files, and uses up about 50 K of disk space.

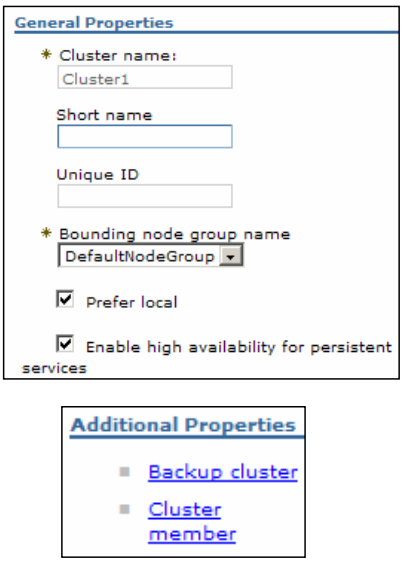
Several application servers can run on a single machine - but there is no requirement that they all be in the same cluster. Clustering is a logical grouping, not a physical one. All members of a cluster are nearly identical 'clones' of a common ancestor.

Creating a Cluster

- Start with an existing server configuration
  - This server may become the first cluster member
- Start with a server template
- Multiple cluster members can be created while creating the cluster
- Servers-> Clusters -> New

Cluster Configuration

- Prefer local
  - Routes EJB client request to local EJB, if possible
- Enable HA for persistent services (for transaction log recovery, discussed in HA presentation)
- Backup cluster
  - If the entire cluster fails, the backup cluster supplies failover routing information
- Cluster member weights



Installing Applications to a Cluster

- Same steps as installing to a Stand-alone server except:
  - Select a cluster as the target, rather than selecting a server
  - Required resources, such as databases, must be available to all cluster members
- Application files (binaries and configuration files) are copied at next synchronization
  - Can be changed using the Administrative Console

Updating applications to a cluster is done in the same manner as updating applications to a stand-alone server.

v5 Application Update on a Cluster

- Steps:
  - Stop application on each cluster member
  - Distribute update to each cluster member
  - Restart application on each cluster member
- Problem: Creates gaps in application availability during the distribution and startup of the update
  - Due to asynchronous update process
- Users instructed to follow manual procedure or scripts to improve the availability

In version 5, users were instructed to follow manual or scripted procedures to achieve availability during application update. The procedure varied slightly between Distributed and z/OS® platforms, but followed a similar pattern:

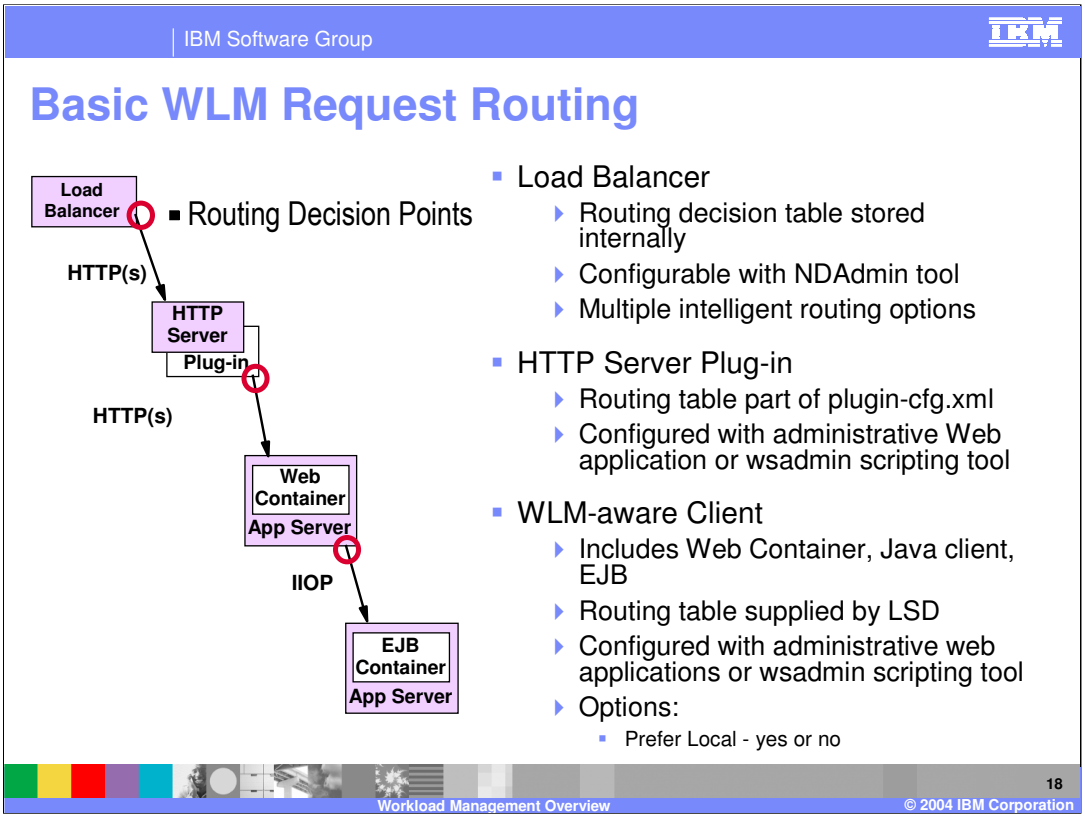
1. Route work away from cluster member
2. Stop application
3. Distribute update to node
4. Re-start application
5. Resume routing work to cluster member

Improved Application Update on a Cluster in V6

- Rollout Update option available for application updates on clusters
- Following steps will be applied to each cluster member in turn:
  - Stop cluster member
  - Distribute update to node
  - Re-start cluster member
- Plug-in detects server outage and can then select another cluster member
- Superior to manual and scripted approaches

Section

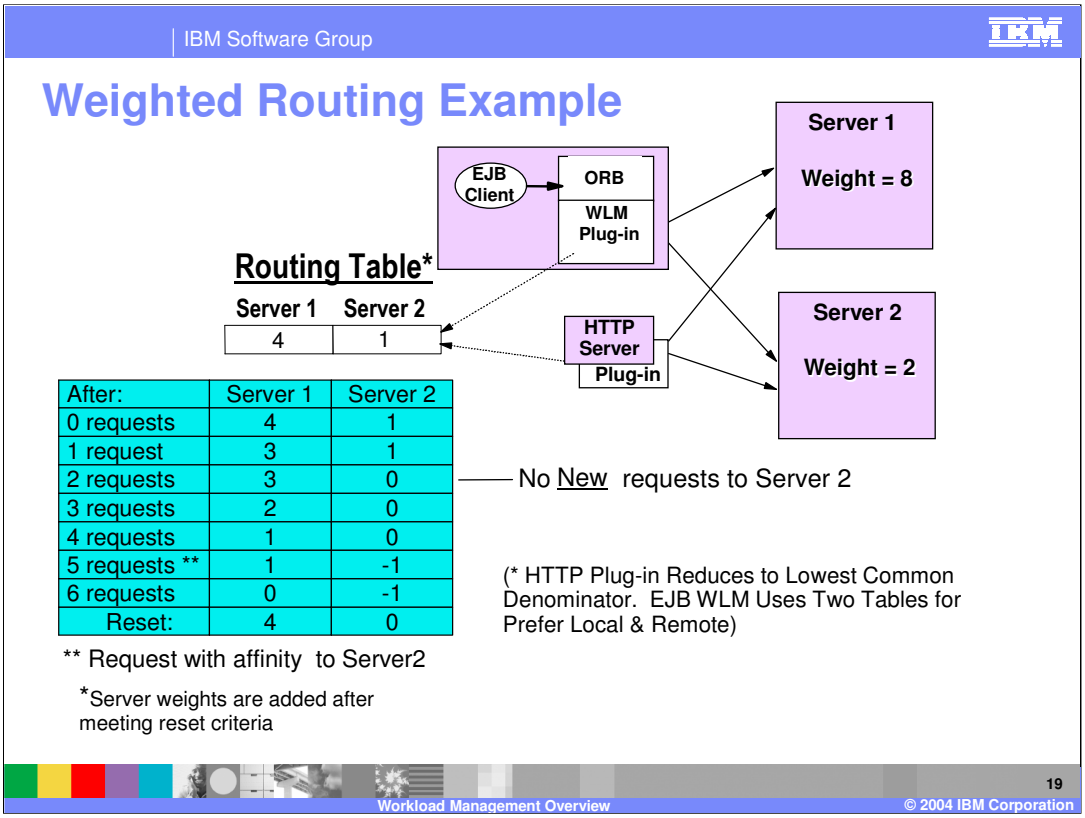
Request Routing



This section addresses the scenario where everything works as planned. Failover situations will be addressed later.

The Load Balancer is an IP sprayer that makes intelligent load balancing decisions. Using the NDAdmin tool, you can set it up to route to your HTTP servers based on Round Robin, Statistical Round Robin, Best, Custom Advisor, or Content Based Routing.

Once the request arrives at an HTTP server, the routing is Weighted Round Robin - the only configuration option is how much 'weight' to give each server. The routing information, the list of available servers and their weights, is ultimately stored by the Deployment Manager. This configuration is used to create the plugin-cfg.xml file for the HTTP server plug-in to read.



Weighted routing is fundamentally the same concept for HTTP requests, or for EJB client requests, so the two are combined on this slide.

When the HTTP Server plug-in is generated, servlet request routing weights are written into the plugin-cfg.xml file, which the HTTP server will reload at configurable intervals.

When a client requests an IOR for an EJB, the Location Service Daemon returns the IOR and a copy of the routing table. The client uses two in-memory copies of the table - one static, one dynamic. The static copy is used only to cache a local copy of the weights.

When the HTTP Server plug-in is generated, servlet request routing weights are written into the plugin-cfg.xml file, which the HTTP server will reload at configurable intervals.

There is a distinct Routing Table for each cluster.

The table illustrated here has two entries, Server 1 and Server 2. The initial values are 4 and 1 (the Least Common Denominator is calculated) The first requests is sent to Server 1, and the counter for Server 1 will be decremented by one. The next request will be routed to Server 2, and the count for server 2 will be decremented.

Basically, the routing is Round Robin for all servers with non-zero table values

Requests with affinity are used to decrement the table. As you can see this can cause a server to go "negative",

When all servers have a zero (or lower) value, then the weights are added to the values in the table.

This assures a more even distribution since requests with affinity are part of the calculation.

Weighted Routing: Mechanics

- HTTP Plug-in and the ORB in the EJB client has a routing table for each cluster
- Routing table decremented on each new request
- No new requests to the Application Server, once the routing entry reaches zero or less, except when overridden by:
  - Affinity (Transaction, HTTPSession)
  - In Process (Handled by ORB)
  - Prefer Local
- Suggested best practice
  - Utilize low values to avoid load variations
  - Plug-in will use least common denominator to minimize variation

One thing to emphasize is that only NEW requests are subject to the weighted routing - requests that have sessions already in progress will be sent to the server that started the session. Either session Affinity or transaction affinity will override the routing.

There are actually two tables passed when Prefer Local is set - only the 'Local' table is used, unless all the servers in the Local table have failed. At that point, the other table comes into play.

Since the difference between the values in the tables is handled sequentially, it is a good idea to use small numbers for the weights.

Weighted Routing: V6

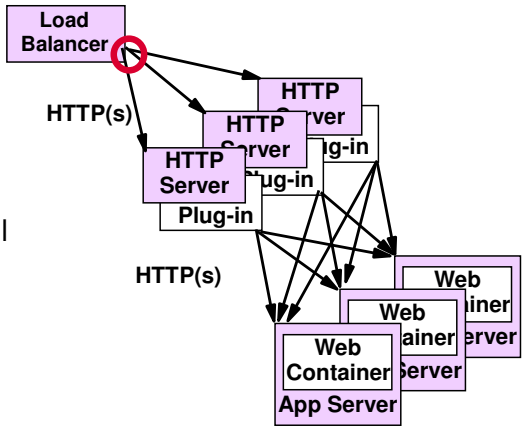
- Round robin routing based on provided weights
- Server affinity maintained
  - Same as in V5
- WLM routing coordinator is an internal service
  - Highly available – if the process it is running on fails, the WLM coordinator will be moved to another server
- EJB WLM includes ‘fairness’ balancing
  - For example, weights of 2 and 7 will result in a-bbbb-a-bbb rather than a-b-a-bbbbbb

Section

Failover of the Servers

HTTP Server Failover

- Multiple HTTP servers provide coverage
- Edge Server can route around failed HTTP server
- HTTP Plug-in
  - Every plug-in contains configuration information for all Web containers
  - Session key contains address of server
  - Sessions get properly routed
- Topology is 'Active/Active', with all HTTP servers handling load before failover



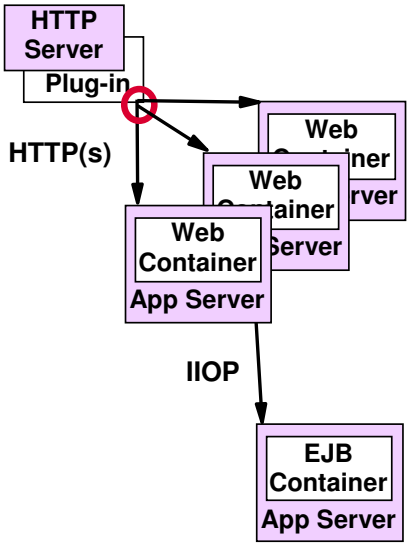
Typically, a production environment will have multiple HTTP servers; each of those HTTP servers will route to multiple WebSphere Application Server instances.

Each plug-in contains configuration information for all of the servers; it has a server list and a back-up server list. If all the servers in the server list are unavailable, it will route to the backup list. (This must be manually edited into the plugin-cfg.xml file.)

If any HTTP server fails, the Load Balancer will simply route around it. The plug-in reads the cloneID from the session key, and can route the request to its originating server.

Web Container Failover

- HTTP Server Plug-in
  - Detects failure
  - Marks container as unavailable
  - Tries next cluster member in the cluster
- What about In-flight sessions?
  - Sessions may be persisted to database or replicated in memory (using DRS)



If a server process fails, the HTTP server notes the failure and marks that application server as unavailable, then routes the request to the next cluster member.

Sessions already in progress will have a server ID for that failed server; the HTTP server routes them to the next server... Session data can be handled that two ways. Session Persistence to a Database, or internal messaging of session information.

Database session persistence functions largely as it did in version 4.0.

WebSphere Internal Messaging was new in 5.0, and the details are in the Data Replication Service module. One point to note here is that since the routing algorithm is a round robin algorithm, if a server fails, requests for that server will always go to the SAME 'next' server. This allows you to use the configuration where servers get memory information from only one other server... The 'Buddy' System. This reduces the amount of memory replication necessary in a cluster.

One path through the GUI to session configuration settings is:

Servers->Application Servers -> <Server Name>->Web Container->Session Management->Distributed Environment Settings

Then either:

->Database

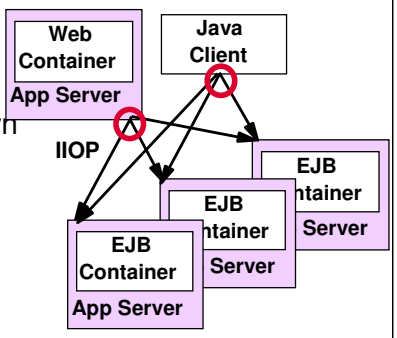
or

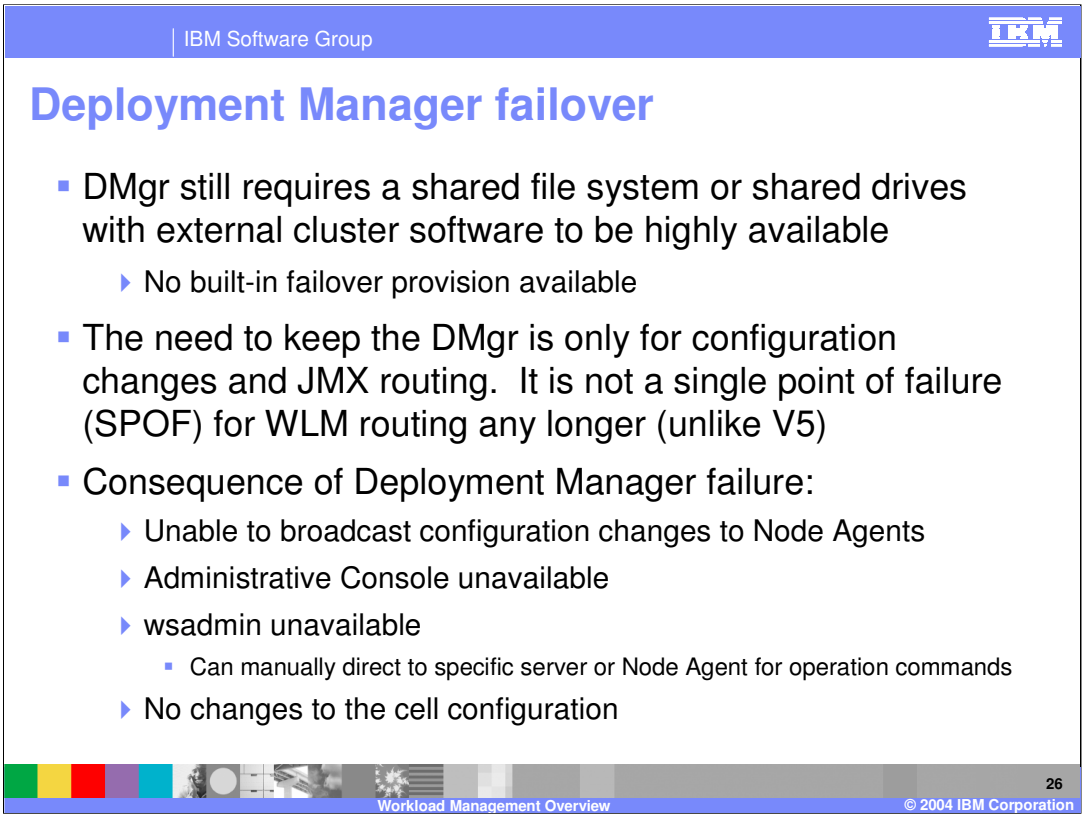
->Memory to Memory Replication

In version 6, the scope of the replication domain is the core group; DRS will coordinate with the WLM component to determine which cluster members should hold backup session information for a specific cluster member.

EJB Container Failover

- Client code and ORB Plug-in can route to next available cluster member
- If failure occurs before any work was initiated on the cluster member:
  - ORB automatically re-routes EJB request
  - If no other cluster member available, throws "NO_IMPLEMENT" exception
- If failure occurs after EJB method call:
  - Transient “signal retry” exception is thrown
  - Client needs to determine appropriate recovery
    - Re-issue request, rollback transaction, etc.
  - If "NO_IMPLEMENT" exception thrown, no recovery is possible





What about a Deployment Manager failure?

The Deployment Manager does not handle client requests – it only handles the configuration repository. Since the configuration information is replicated to the Node Agents, and the Node Agents are distributed, Network Deployment does not address Deployment Manager failover.

Other than the fact that Node Agents will not be notified of failed servers on other nodes, the impact of a downed server is minimal. You will want to restart it, but while it is down, no requests will fail in-flight.

Singleton services, like the WLM routing component, are 'portable' in V6, they do not run in the Deployment Manager.

Node Agent Failover

- Node Agent still requires a shared file system or shared drives with external cluster software to be highly available
  - No built-in fail over provision available
- The need to failover a Node Agent is significantly less with V6
- The Node Agent should however be kept running using a process nanny as the Location Service Daemon (LSD) still only runs inside the Node Agent
- WebSphere V5 and V6 clients need at least one Node Agent available for first call to LSD
  - These clients can use a list of Node Agents, if available
- Interoperability with v3.x and v4.x
  - Clients will work only while the specific Node Agent they connect to remains alive
- WebSphere-aware clients get "special treatment“ - private information in the WLM TaggedComponent
  - Includes a complete description of the cluster topology
  - Secure and non-secure transport ports for all cluster members are included and can be used directly by the client ORB

Section

Summary and Reference

Summary

- Defined Workload Management
- Described Clusters and Cluster Members
- Examined the Topology
- Weighted Server routing topology
- Reviewed Failover scenarios
- Visited the HTTP Plug-in briefly
- Listed Problem determination steps

In summary, this presentation has reviewed a basic definition of Workload Management. Then it looked at the topology, and explained where request routing decisions are made. Also explained were failover scenarios, and how various components are configured to avoid a Single Point of Failure (SPOF).

Finally, some problem determination suggestions were offered.

Section

Appendix

Running without Node Agent

- In specific configurations, running specific applications, application servers can run without the Node Agent running

Not Recommended

- Consequences:
  - V5.1 - No Location Service Daemon – no indirect IOR lookups
  - ORB_LISTENER_ADDRESS set manually
  - EJB Workload Management may not work
- Specifics: Information Center topic “Configuring Inbound Transports” (search string: “without node agent”)

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:

IBM	CICS	IMS	MQSeries	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

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

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

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

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 and/or changes in the product(s) and/or program(s) 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 (e.g., 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.