IBM Software Group

IBM WebSphere Application Server v6

Workload Management and High Availability

e business on demand.

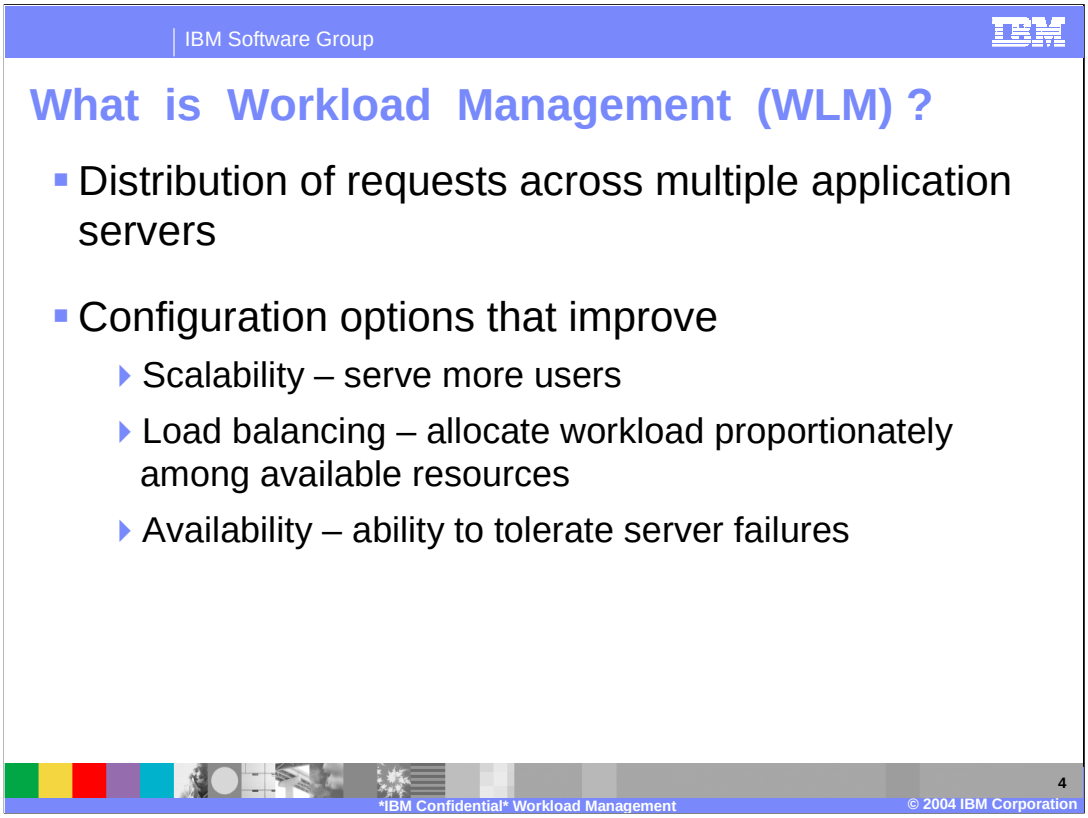
Updated October 12, 2004

Agenda

- WLM review
- New WLM functionality
- Changes to the Data Replication Service (DRS)
- Failover and High Availability

Section

Overview

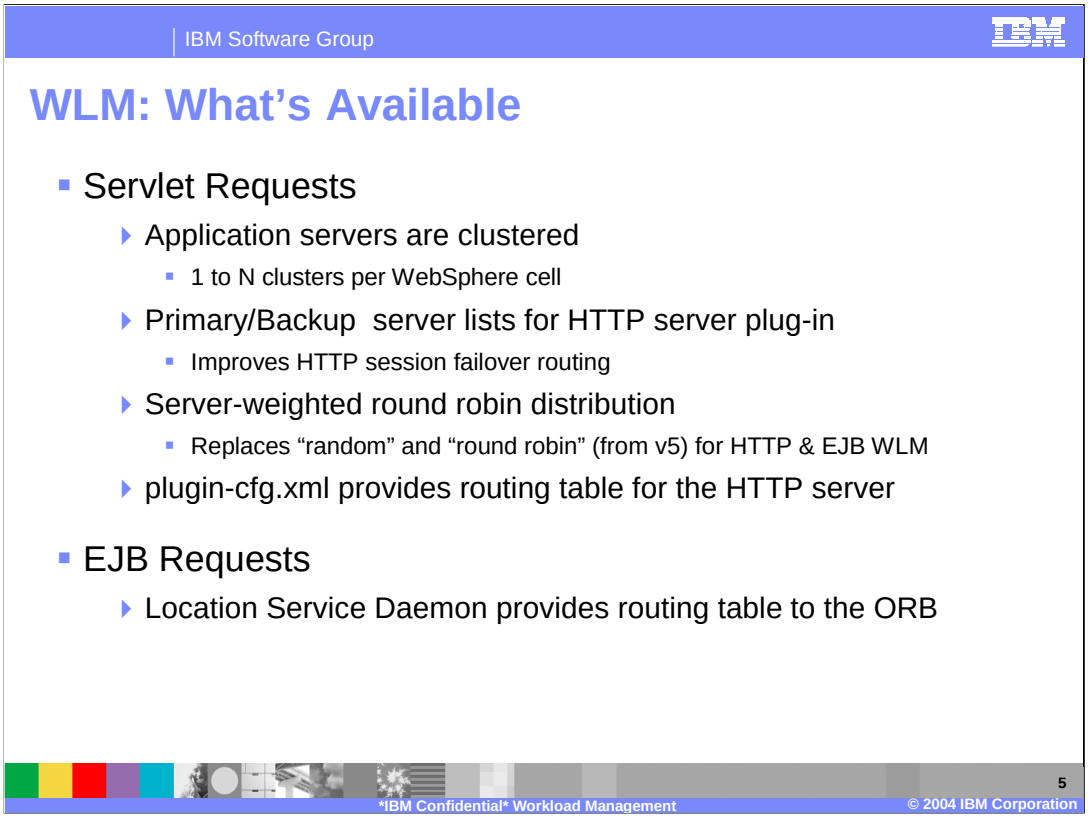


Workload management optimizes the distribution of client processing requests.

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.



HTTP requests coming into the web server can be managed by a load-balancing product, such as the WebSphere Edge Components.

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 we will discuss 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

Unified Clustering Framework

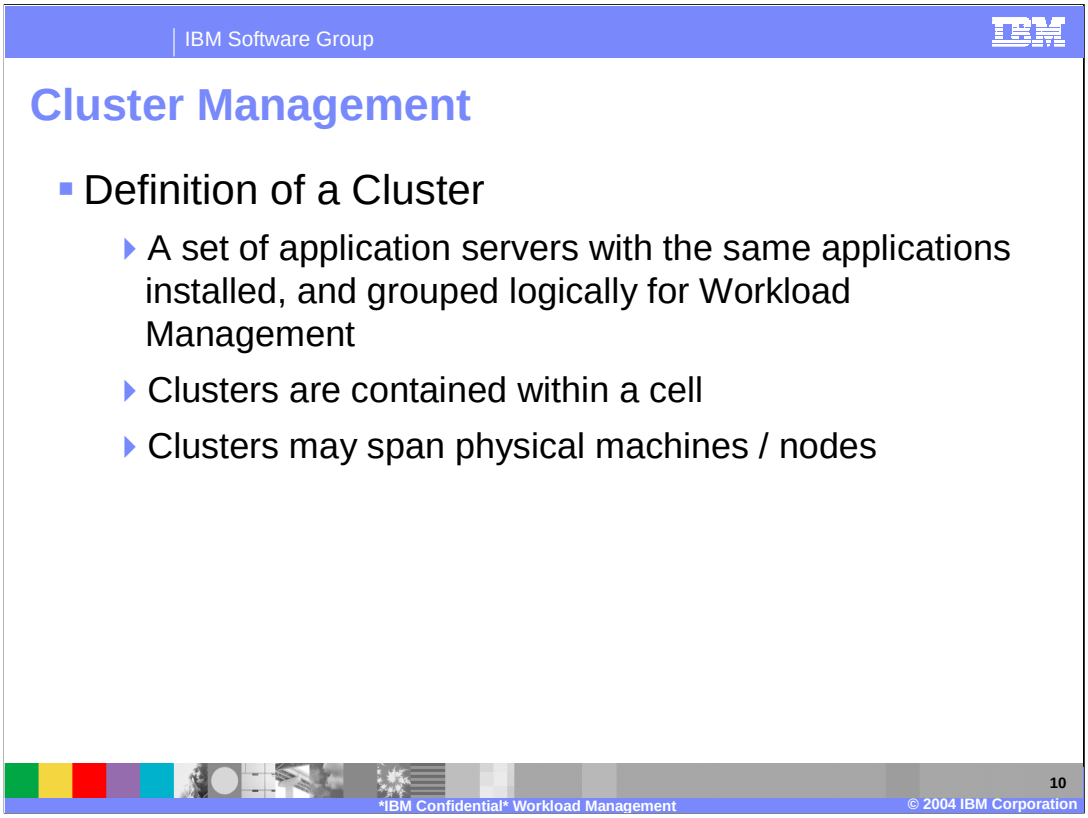
- Common clustering logic across different resources that require clustering
  - The view and use of clusters is administered in a unified and consistent manner for all protocols (HTTP, EJB, JMS, JCA, etc)
- New WLM functions can be implemented once for all protocols
- High Availability
  - Makes WLM routing a highly available service, which makes cluster and routing information always available

Failover of Stateful Session EJBs

- Uses the Data Replication Service, similar to HTTP session failover
- Always enabled
- 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

Section

Clustering



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.

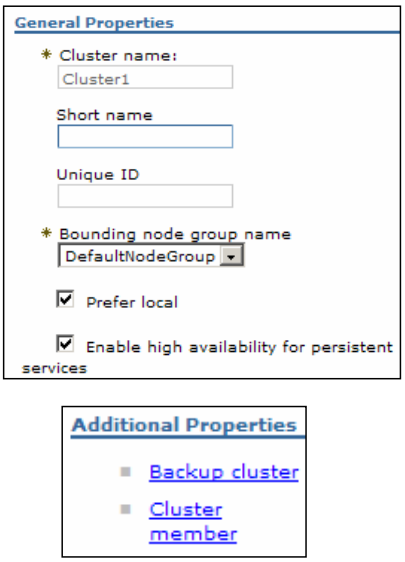
Several application servers can run on a single machine - but there is no requirement that they all be in the same cluster. Clustering is a logical grouping, not a physical one. All members of a cluster are nearly identical 'clones' of a common ancestor.

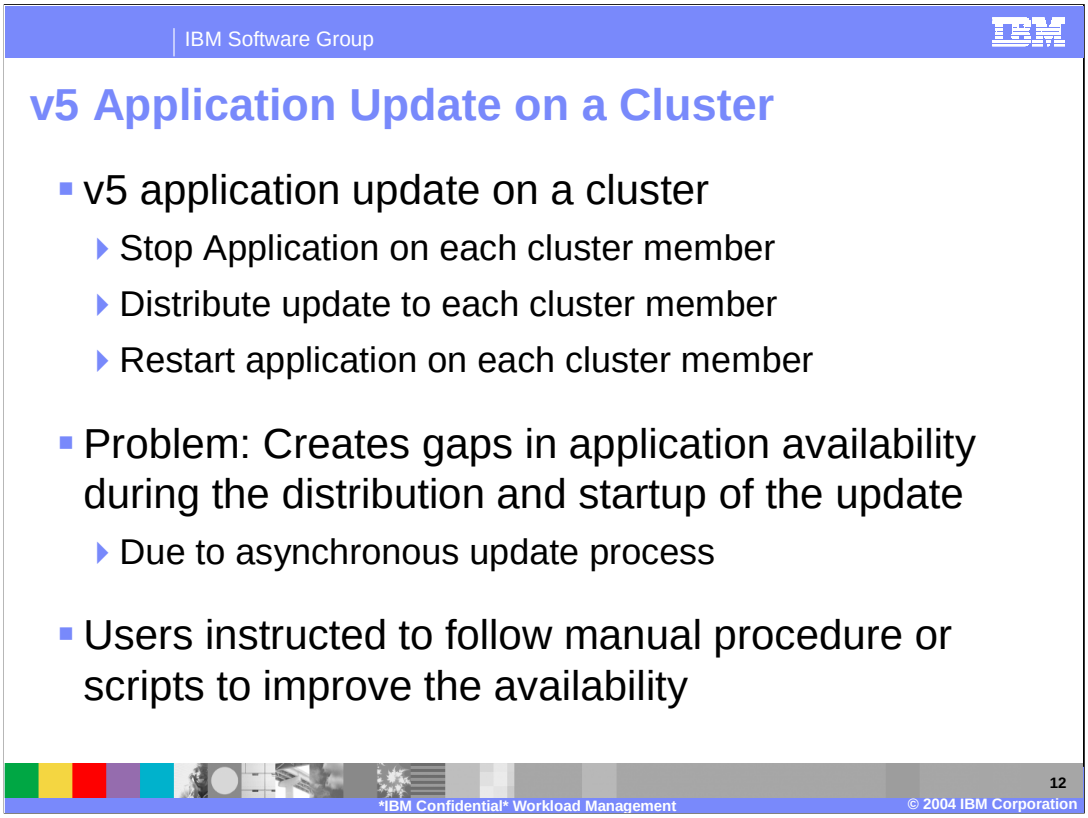
Cluster Configuration

- Prefer Local – routes EJB client request to local EJBs, if possible
- Enable HA for persistent services (for transaction log recovery)

New v6

- Backup Cluster
  - If the entire cluster fails, the backup cluster supplies failover routing information
- Cluster member weights





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:

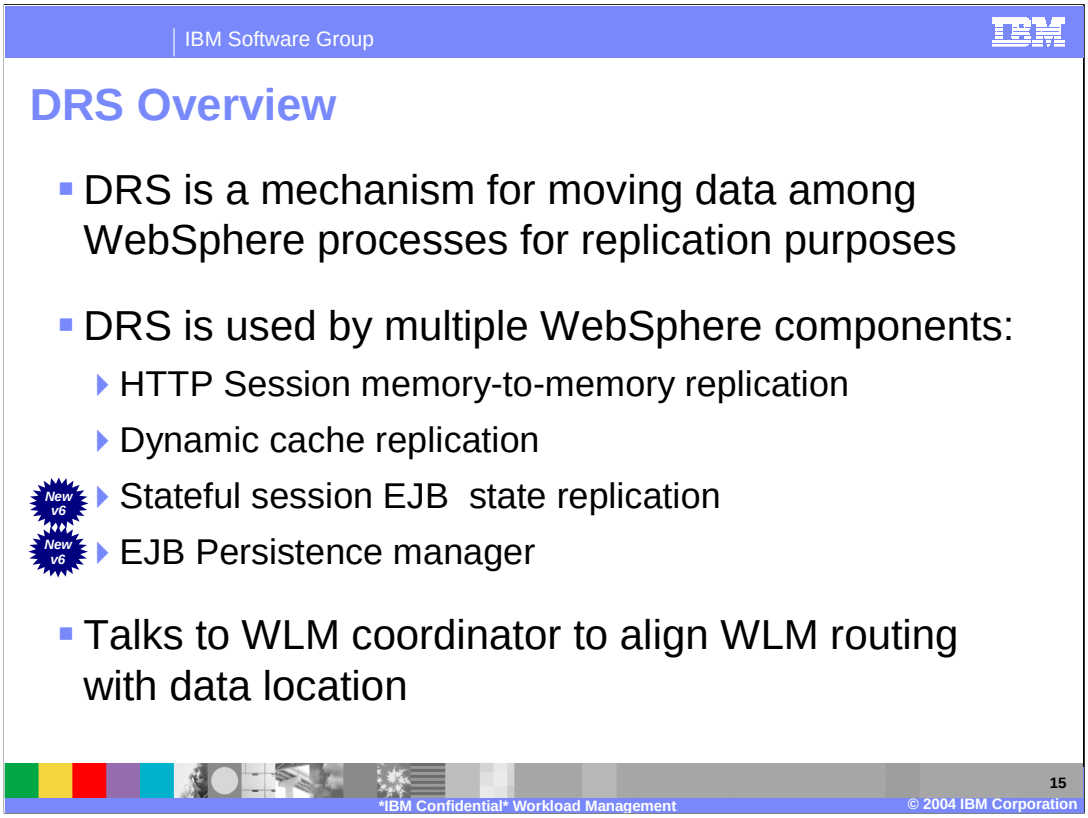
1. Route work away from cluster member
2. Stop application
3. Distribute update to node
4. Re-start application
5. Resume routing work to cluster member

Improved Application Update on a Cluster in v6

- 'Ripple start' option for application updates on clusters
  - For each cluster member, ripple start will:
    - stop server
    - distribute update to node
    - start server
- Plug-in detects server outages during update and can then select another cluster member

Section

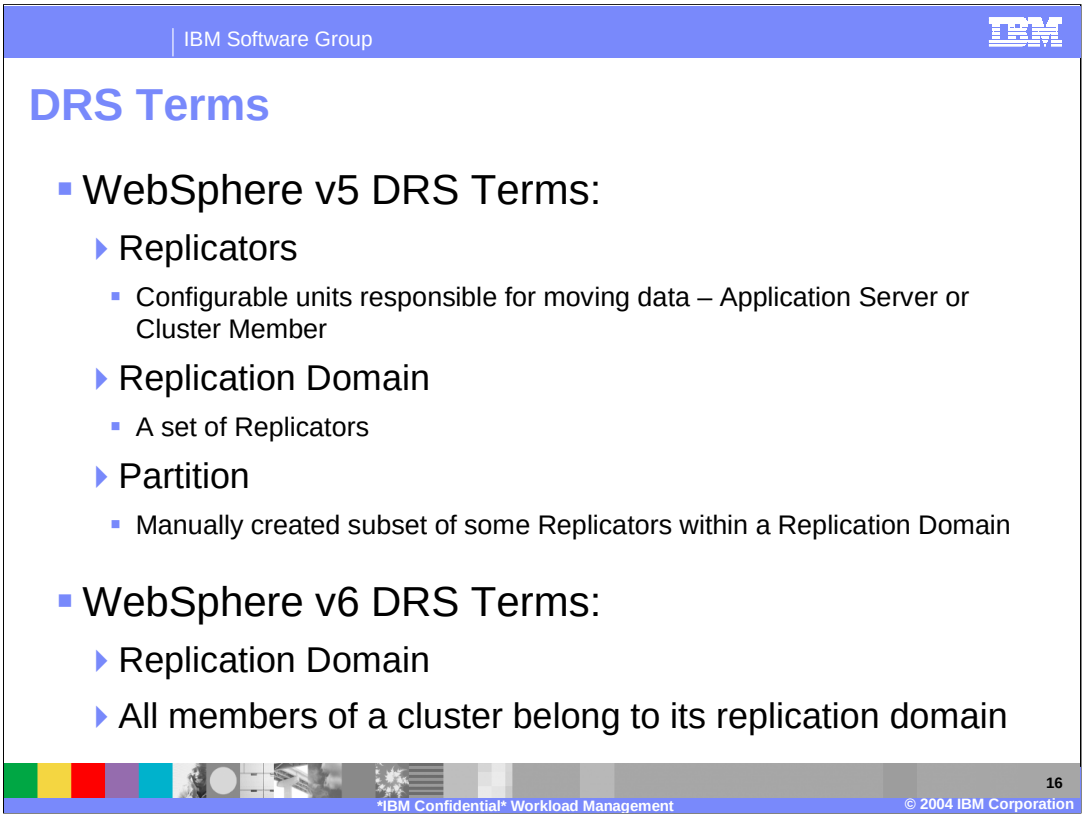
Data Replication Service (DRS)



Data Replication Service is an internal component of the WebSphere Application Server. It is used by other internal components to move data from place to place. The most visible use of DRS is for replicating persistent HTTP Session data so that if an application server fails, the request can be routed to another application server, and the session data will be available there.

In order to minimize the impact of a failover, DRS coordinates with the Workload Management routing algorithm to assure that requests and data end up in the same place.

A new feature in Version 6 is the capability to capture the state of a stateful session bean and enable failover to another instance of that bean in another application server.



Version 5 of WebSphere leaves much of the configuration to the Administrator. Replicators are the JMS producer and consumer that are responsible for moving data as JMS messages. Administrators create Replicators within a Replication Domain; the default configuration is to have all the application servers in a domain talk to all the other application servers. It is possible to reduce the overhead by limiting which application servers talk to which; those that are configured to talk to each other are a partition, as well as being part of a MultiBroker replication Domain.

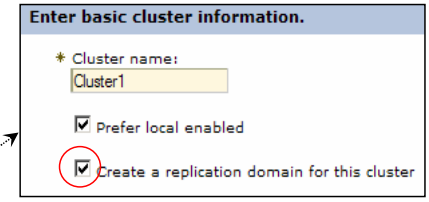
Version 6 simplifies this. Because the underlying mechanism has changed, it is no longer necessary to manually create and configure Replicators. Also, the concept of Partitioning is masked. Even though it is still possible to limit the number of copies of the data, it is not necessary to expose the details of that configuration.

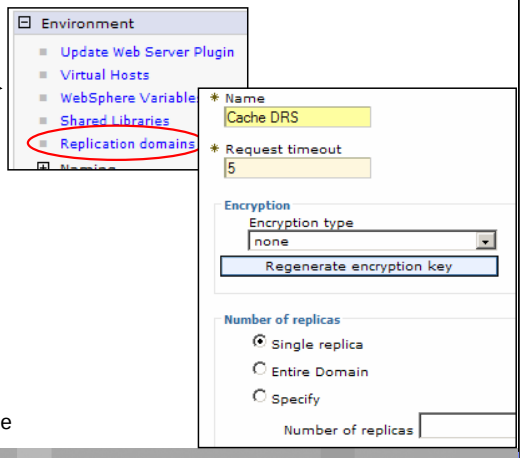
DRS: New v6 Functional Changes

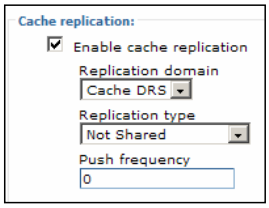
- Integration with WLM to provide "hot failover" in peer to peer mode.
- Ability to collocate stateful session EJB replicas with HTTP session replicas with hot failover
- Faster underlying transport
  - Allows for the use of both multicast and unicast IP

DRS Administration

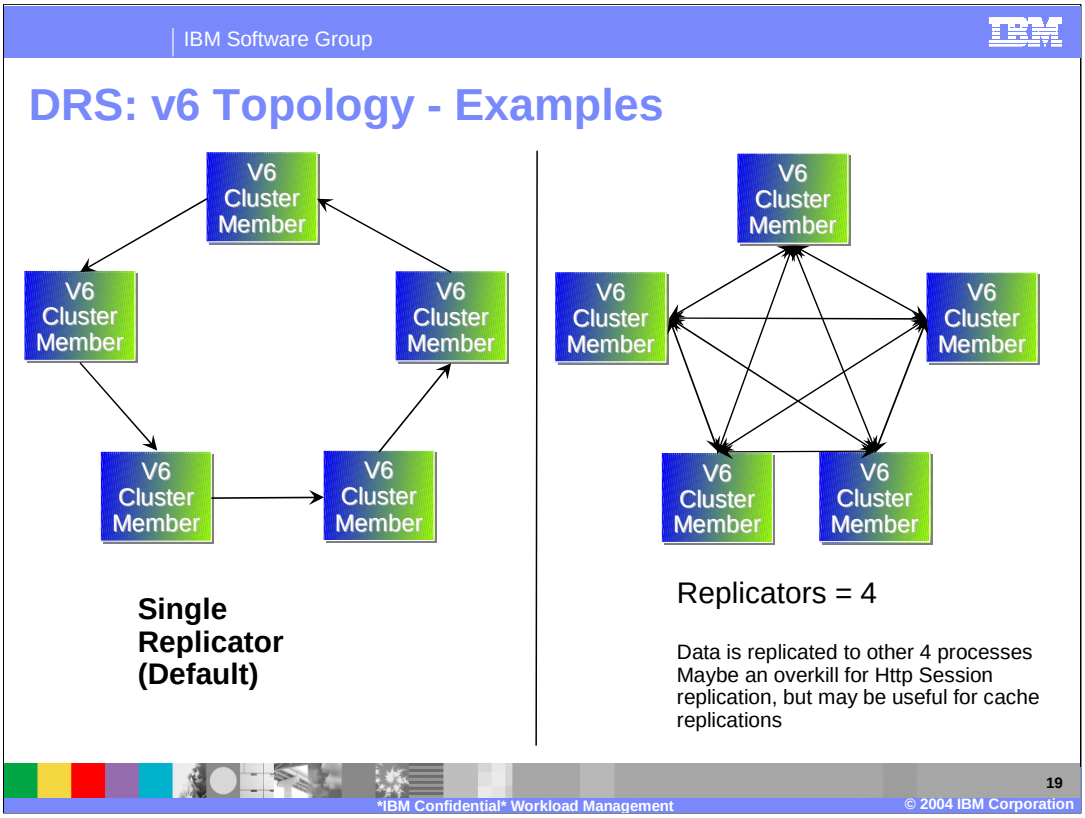
- DRS domain can be created as follows:
  - Optionally, when creating a cluster - DRS domain name same as cluster name
  - Manually create DRS domain
- Enabling Dynamic Cache to use DRS







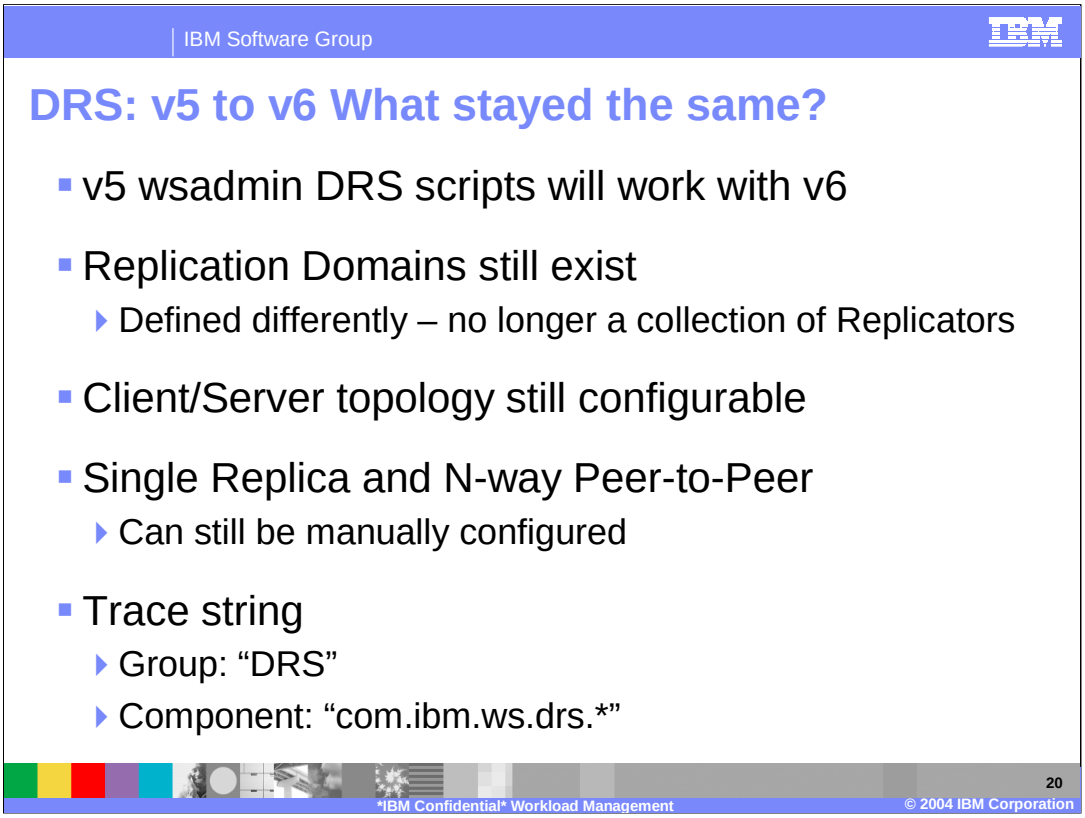
Server -> Container Service -> Dynamic Cache Service



Note that the backup cluster member may not be in the same sequence as the WLM routing.

For N=4

Note that the arrows each have two heads – data flows from each process to every other process, so that for each replicated object, there are four remote copies and the local original. This topology would probably be overkill for HTTP Session replication, but it is the most useful configuration for a cache replication domain. When caching dynamic content, the cache is most useful if it is available on all the machines where a request could arrive.



wsadmin scripts that create replication domains for version 5 will still work with version 6, and the common topologies used in version 5 will still be possible. The difference is that the topology with the highest performance cost will be the most tedious to recreate – N-way peer-to-peer topology will require changing the value of N whenever a cluster member is added to the domain.

Best Practices

- Create distinct replication domains for distinct data
  - e.g., one domain for Dynamic Cache, another for HTTP Sessions
- Put Stateful Session Bean and HTTP Session data in the same domain
  - Typically developers stash a stateful bean ref in session
- Set number of replicas to small values
  - Balance resource use with failover ‘comfort level’

Where Does The Backup Data Go ?

- Dynamic Cache data goes to all nodes
- HTTP Session and Stateful Session Bean data is sent to 1-n other servers
  - Depends on # of replicas setting
  - Decided at startup
  - Based on which servers are backing up fewest servers
    - Not based on WLM routing information
  - Core Group settings define which servers participate

Section

Failover and High Availability

Failover: Sequence of Events

- Backup servers determined at server startup
- HTTP Request
  - Session information persisted to backup server or servers
  - Session manager places data in cookie indicating backup
- Server fails
  - First request routed to next server
  - Session data fetched from backup
  - Cookie updated and WLM coordinator notified of new backup server
  - WLM coordinator re-routes subsequent requests

What is updated in the WLM coordinator is the list of servers and the partitions that they are serving. The plug-in can then decide where to route a request based on session affinity – the new affinity, based on the new information placed in the cookie by the session manager.

WebSphere v5: HA Status

- WLM services run in the Deployment Manager
  - This makes the DMgr a SPOF
  - Must be made highly available to make sure WLM works correctly
- External clustering solution (e.g. HACMP) is required on WebSphere boxes if:
  - Using 2PC transactions
    - Clustering solution is required to recover in-flight transactions
  - Using Embedded messaging
- Cold failover only
  - Potentially 5-6 minutes downtime, if there is a failure

WebSphere v6: HA Overview

- Significant improvements in high availability
  - Can be used as part of an overall 99.999% availability solution.
- High Availability Manager is responsible for running key services on available servers rather than on a dedicated one (such as the DMgr)
- Can take advantage of fault-tolerant storage technologies such as NAS
  - Significantly lowers the cost and complexity of HA configurations
- Hot standby and peer failover for critical singleton services
  - WLM routing, PMI aggregation, JMS messaging, Transaction Manager, etc.
  - Failed singleton starts up on an already-running JVM
  - Planned failover takes < 1 second
- The configuration of highly available systems is simplified
  - Works out of the box in most cases

Failover for Different Service types

- Singleton Services that require failover fall into two categories:
- User Resources: Artifacts coming from the application
  - Example: Transaction log for 2PC transaction, JMS Messaging Engine
  - Failover occurs only within the cluster boundary to another cluster member, not to any other servers or clusters within the Core Group
- System Resources: Used internally by WebSphere
  - Example: WLM routing, PMI aggregation
  - Fail over can occur to any process within the Core Group

WebSphere v6: HA of DMgr and Node Agent

- Deployment Manager
  - No longer a SPOF for WLM routing
  - DM only needed for configuration changes and JMX routing
  - Still requires a shared file system or shared drives with external cluster software to be highly available
- Node Agent
  - If a node agent needs to be highly available then the same applies
  - The need to failover a node agent is significantly less with v6 when using NAS for transaction logs or a remote DB for messaging
  - The node agent should be kept running using a process nanny, since the Location Service Daemon still only runs inside the node agent

Section

High Availability (HA)

Core Groups

HA Domain: Core Groups

- Defines the set of WebSphere processes that participate in providing High Availability function to each other
  - The set of processes is called a **Core Group**
- Processes in Core Group can be DMgr, Node Agents and Application Servers (Cluster Members)
  - A process is a member of exactly one core group
  - All members of a cluster must be within the same Core Group
- WLM information is shared automatically between Core Group processes in a peer-to-peer fashion
- Singleton services running in a Core Group can failover only to another member of the same Core Group

Default Core Group

- DMgr installation creates a default Core Group
  - Called “DefaultCoreGroup”
  - Has default HA policies for Transaction Manager and JMS messaging
- As WebSphere processes are added to the cell, they are automatically added to the Default Core Group
- In most of cases, the default setting is good enough
  - You don’t usually need to change the defaults or add more groups

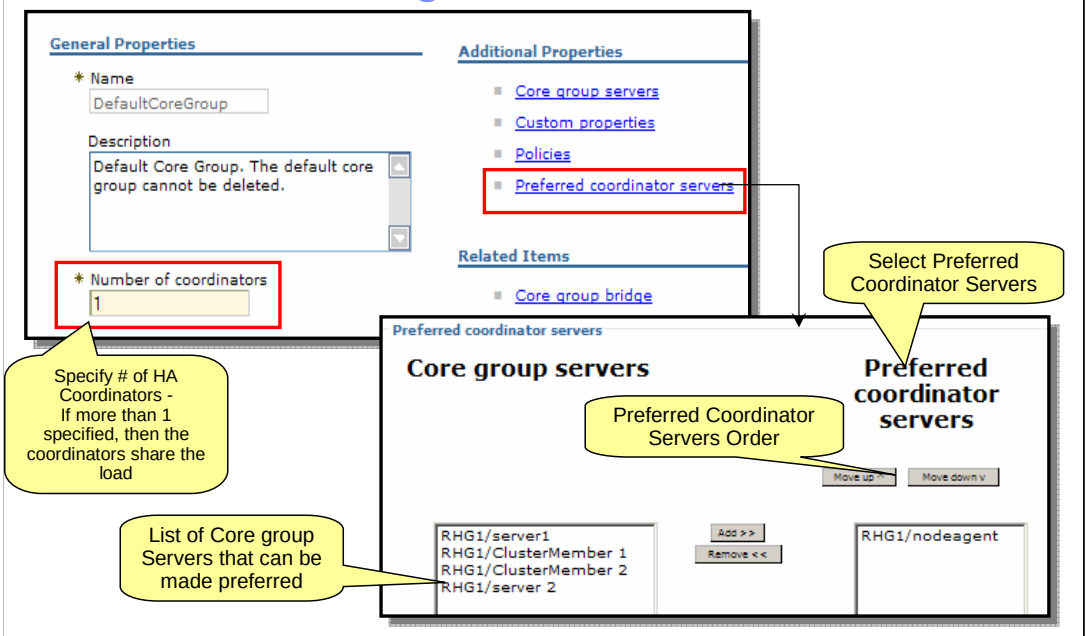
Multiple Core Groups

- When to use more than one core group:
  - One cell spans multiple geographies
    - e.g. London, New York, San Francisco core groups may be in one cell
  - Some servers running within the DMZ
    - e.g. to manage HTTP Servers
  - For performance when large number of nodes in use
- Core Group Bridge
  - Connects 2 core groups that are intra or inter cell
  - Allows WLM information between the core group processes

HA Coordinators

- Each Core Group needs an HA Coordinator that coordinates all HA activities among the Core Group processes
  - Collects all the HA information – what services are running in which processes
  - If a service in one process fails, then the services are restarted in another process (based on HA policies)
- Any process in the Core Group can be the HA Coordinator
  - selected using internal algorithm as to who is the HA Coordinator
- Can have multiple live HA Coordinators
  - The coordination load is shared so one process does not get overwhelmed
  - Useful when you have many clusters within the cell
- Can assign preferred servers to be the HA Coordinators
  - WebSphere will try its best to use the preferred server
  - Will use other servers only if the preferred server is down
- What happens if the Coordinator fails
  - The servers in the Core Group “elect” another server to be the coordinator (from the preferred list, if possible)

Coordinator Configuration



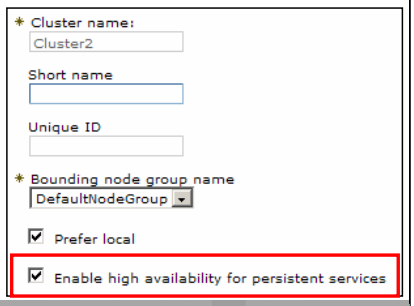
Section

High Availability of

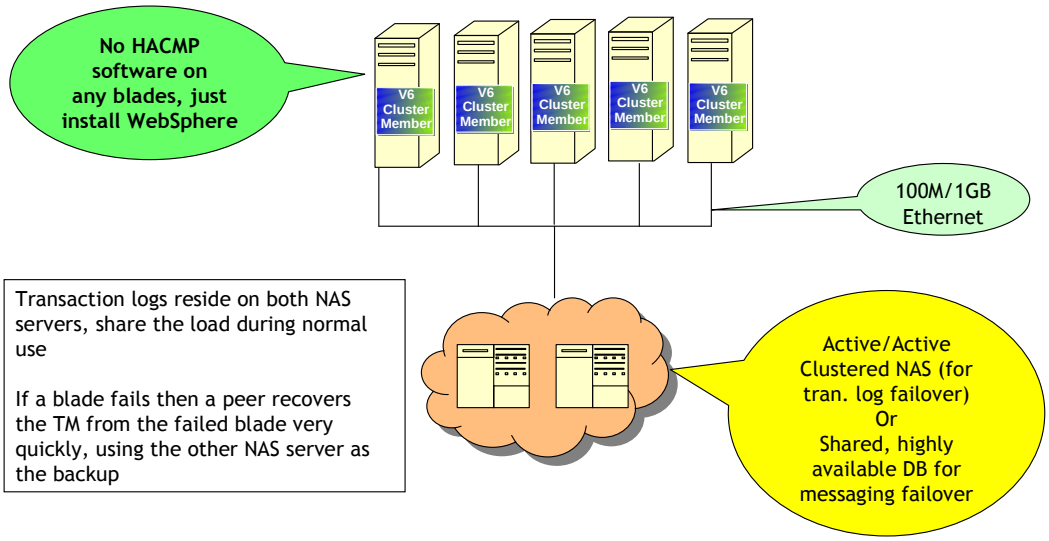
Transaction Logs and JMS Messaging

Transaction Log Hot Standby

- Allows failover of in-transit two-phase commit (2PC) transactions
- WebSphere v6 can be configured to store transaction logs for each server on a NAS (Network Attached Storage) shared file system
- When a v6 cluster member fails, then a peer is elected and directed to recover the transaction log from the failed server – all peers can see everyone else's transaction logs
- This allows the in doubt transactions from a failed server to be recovered very quickly
  - Huge improvement over v5, where recovery was in minutes and required OS clustering and shared disks
- This option must be enabled explicitly



Hot Standby Example



Transaction logs reside on both NAS servers, share the load during normal use

If a blade fails then a peer recovers the TM from the failed blade very quickly, using the other NAS server as the backup

Section

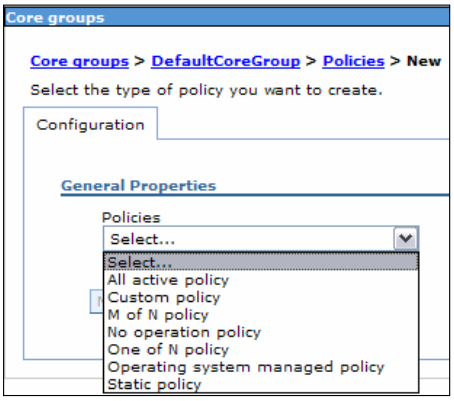
High Availability (HA) Policies

Core Group HA Policies

- HA Policy defines how the failover occurs and which server to use for failover
- Core Group can have different HA policies for different services
  - Example: WLM clustering for transaction can use one HA policy, while JMS Messaging can use another HA policy
- By default, following HA policies are defined for the DefaultCoreGroup:
  - Clustered TM policy for clustered applications
  - Messaging policy for Service Integration Bus services

Custom HA Policies

- Singleton services can be configured to run in three basic modes
  - Can be specified per singleton
- These modes are:
  - Static
  - 1 of N
  - Operating System (OS) Managed



There are some other modes available for some more advanced purposes.

Custom HA Policy: Static

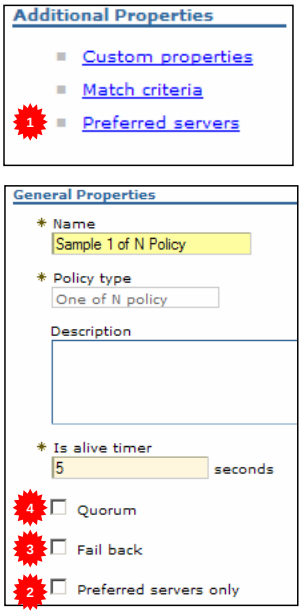
- Makes WebSphere v6 behave like WebSphere v5
- A singleton is only placed by on a (fixed) single server
- This means that if that server is not running then the singleton service is not running
- However, the fixed server can be changed at any time without restarting WebSphere, which is different than WebSphere v5
- If failover is required then the whole node needs to be failed over (using HACMP, etc.)

Custom HA Policy: 1 of N

- WebSphere decides on which candidate server the singleton services (JMS, TM, DRS replicas, etc.) can be started
- All external resources must be available to all possible candidates
  - For messaging this implies a remote database or a cloudscape database on a NAS mounted on all candidate machines.
  - For transactions this implies the transaction logs are on a NAS mounted on all candidate machines.
- WebSphere will keep the singleton service running on exactly one of the candidate servers using this mode

1 of N: Additional Options

- **Preferred Server list** 1
  - Specify a list of servers that WebSphere will prefer when choosing where to run a singleton
- **Preferred servers only** 2
  - This list can be exclusive (i.e. only run on those in the list), or WebSphere will use the list, but if all preferred servers are down WebSphere can choose a server from the remaining servers
- **Fail back** 3
  - If a more preferable server becomes available, more the service back to that server
- **Quorum** 4
  - Need a quorum (more than half the servers) to start the singleton services
- All the above configuration can be changed without restarting the Application Server
  - *cron* can be used to change configuration using a schedule



Example: 1 of N Policy

- Servers elect one server to be “The Server”
  - Special election held when “The Server” is no longer visible AND more than 50% of the servers ARE visible
- Configurable options:
  - Preferred Server List – {A,C}
  - Preferred servers only
    - True - Try A, try C, then fail
    - False - Try A, try C, try anything else visible
  - Fail back
    - True - A dies, C takes over, when A recovers, A takes over
    - False - A dies, C takes over, when A recovers, C stays

Custom HA Policy: OS Managed

- WebSphere never activates the singleton on its own
- A third-party must use JMX to tell WebSphere where to place a group of singletons, which are grouped together using a keyword
- Typically, this mode is used when the singleton has dependencies on resources managed outside of WebSphere by external clustering software
  - e.g., Transaction Manager logs may be on a shared disk that is only mounted on a single server at a time
  - Messaging may need to use a co-located database which is managed by external clustering software
- This allows external clustering software to leverage the hot standby capabilities of WebSphere v6, reducing recovery times from minutes to seconds

Section

Failure Detection

Failure Detection: Active Heart Beat

- Active heart beating
  - All JVMs send a heart beat to each other every second
  - If a JVM doesn't receive a heart beat from a peer for 20 of these intervals then it tells the others to suspect that peer
  - These values are fixed
  - This approach is problematic on machines that swap or are prone to become unresponsive. It is not recommended in these cases.
- This is mandatory in multicast mode
- This is optional in TCP/IP mode

Failure Detection: TCP Keep-Alive

- Open a keep-alive socket between peers
  - If the socket to another peer is closed then that peer is suspected
- This works well on machines that will swap or become unresponsive
- This applies only to TCP or channel framework
- Tune the operating system's KEEP_ALIVE setting to detect failure in an acceptable time
- If KEEP_ALIVE is turned off then active heart beating must be enabled

Section

Summary and Reference

Summary

- New v6 HA Services provide impressive levels of availability that traditionally could only be handled by expensive clustering products
- The HA functions can be customized by creating HA policies

Trademarks and Disclaimers

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

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

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.

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.

This Page Intentionally Left Blank