
This presentation will cover the partitioning facility of WebSphere Extended Deployment for z/OS V6.0.1.

# Agenda

- Overview / example scenario
- Partitioning concepts
- Database partitioning
- Performance advantages




This presentation will first introduce the partitioning facility through an example scenario that compares a classic Java™ 2 Enterprise Edition (J2EE) application to a partitioned application. It will then cover concepts associated with partitioning, to familiarize you with the partitioning facility. Database partitioning will also be discussed, followed by summarizing the performance advantages of partitioning.

# Section

***Overview***



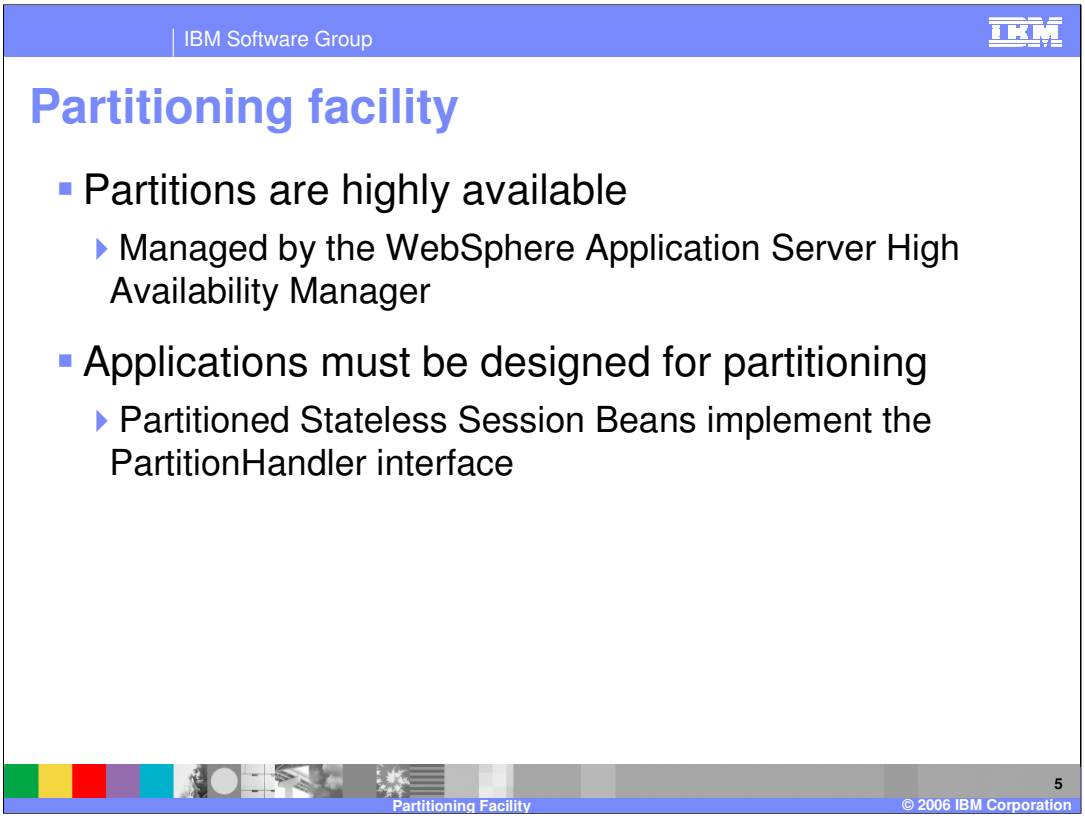

This section will introduce the partitioning facility.

# Partitioning programming model

- Caching of resource expensive data
- *Reduces lock contention*
- *Extensive caching*
- *Application portability*
- *Partitions are highly available*
- *Requires user design/coding*
- *Limited to JMS connections on z/OS*



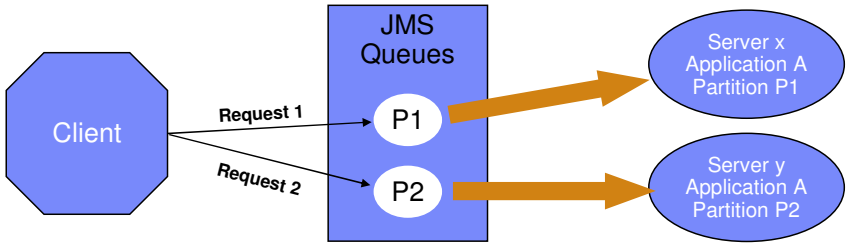

The partitioning facility is a programming framework and runtime environment that makes it possible for high transaction-volume applications to increase resource utilization. To accomplish this, an application is partitioned across multiple servers in a cluster. Each partition is a uniquely addressable endpoint within the cluster, to which requests for certain data sets are always routed. Partitioning solves some of the traditional challenges of very large clustering, because it can reduce data contention and reduce the overhead of replicating shared data, like caches or state information.




While partitioning reduces the number of locations within a cluster where a particular piece of logic may be running, your entire application remains highly available, because partitions are managed by WebSphere Application Server's High Availability Manager. The High Availability Manager ensures that every partition is running at all times, even in the case of a server failure.

While the standard J2EE programming model is available to partitioned applications, you must also create a special kind of bean, known as a Partitioned Stateless Session Bean. This bean instructs the partitioning facility about how your application should be partitioned, and how to associate requests with the correct partition, so they can be routed properly.

# Partitioning on z/OS




- Server implements Partitioned Stateless Session Beans (PSSB)
  - Each server is directed to connect to a specific queue by WebSphere
  - Satisfies requests from that queue until directed to disconnect
- Client determines partition
  - Sends request to corresponding JMS queue (unique end point)




1) When the application “A” starts on server “x”, WebSphere Extended Deployment detects it is a unique endpoint for a partitioned space. WebSphere then directs it to be partition “P1” and the application connects to JMS queue “P1”.

2) In a similar fashion application “A” on server “y” is to service partition “P2” and connects to JMS queue “P2”.

3) In the diagram, before sending “Request 1” to application “A”, the client code determines the request is intended for partition “P1” and queues the request in JMS queue “P1”. The application running on “server x” services the request.

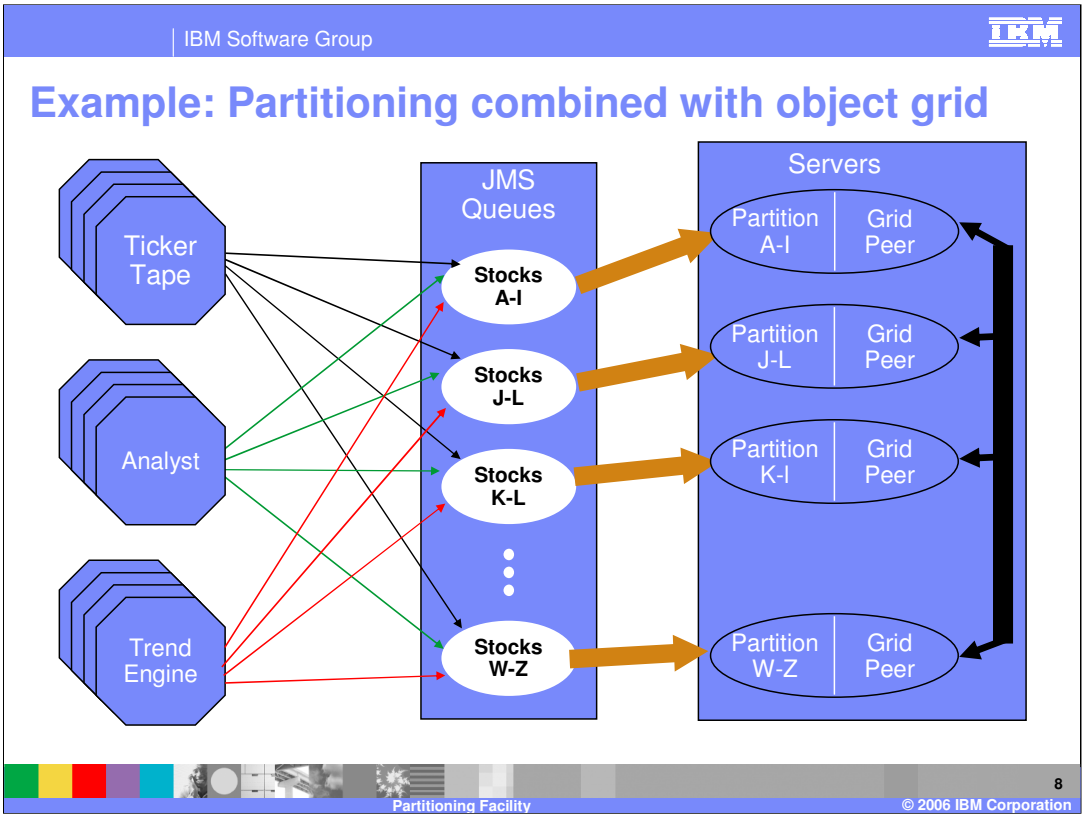
4) Again in the picture, the client code determines the next request is intended for partition “P2” and queues the request in JMS queue “P2”. The application running on “server y” services the request.

## Example: Partitioning combined with object grid

- Technical analysis of stock prices
  - Compute Intensive
  - Large volume of running data
    - Large heaps
    - Frequent cache writes
  - Real time tracking




Prediction of stock price movement requires constant resource expensive computations. When a query is made against a stock price, hundreds of coefficients need to be updated based on all the stock transactions since the last computation. The analysts will query these results for dozens of stocks at frequent intervals as they search for trends. At the same time trend analysis engines will query all the stocks at a lower periodic rate. Trying to keep all the coefficients for all the stocks in a few servant regions would require a huge heap and sharing all the cached coefficients among several servant regions would be a monumental task. Combining partitioning with object grid provides a manageable solution. Here a stock will be routed by stock name to a fixed set of servers and peer to peer object grid could share information for the case of multi stock comparisons and failover.




This is a stylized view of the stock analyst problem from the previous chart. Here the "Ticker Tape" boxes are input feeds of stock prices, volumes and other pertinent data. The "Analyst" boxes are individual people searching for some quality shown in the technical analysis performed on the backend server, labeled ere as Partition A-I and so forth. The stocks are filtered to the correct JMS queue by stock name. If a comparison of one stock's behavior to other stocks is required, the peer-to-peer object grid, labeled as "Grid Peer" here is used to obtain the proper coefficients. In a like manner, the trend engines search trough the stocks for predefined behaviors.

## Section

***Partitioning concepts***




This section will discuss concepts associated with partitioning.

# What is a partition?

- A partition is a uniquely addressable endpoint within a cluster
  - Any cluster member can host a partition
  - A cluster member may host multiple partitions
- Partitions are created at startup when a partitioned application is detected
- Partition lifecycles are managed by the High Availability Manager
  - Ensures availability and uniqueness of partitions




A partition is a uniquely addressable endpoint within a cluster. Each cluster member might host multiple partitions, but each partition exists on only one cluster member. Each partition is started as a highly available singleton, managed by the High Availability Manager infrastructure. This ensures that the partitions remain constantly available while running in only one cluster member at any given time.

## Request routing

- When partitioned applications are detected, a special workload manager singleton for partitions is started
- JMS requests are dispatched to an partition based on a 'key' (such as a passed parameter)
  - User-programmed logic determines how to map parameters to partitions
  - If no partition context is detected, request is routed using normal workload management




The WebSphere Extended Deployment runtime provides a separate workload management router for partitioned applications, which is started automatically when a partitioned application is detected. This router locates and routes the request to the appropriate partition based on a partition key. The application defines partition keys and how they are mapped to partitions. The partition key could be a parameter passed to the method being called, or a hash based on such a parameter. If a partition key is not associated with the request, it will be handled by the normal workload manager, just like any other JMS request.

# What is a partitioned application?

- A partitioned application is a regular J2EE application that contains a Partitioned Stateless Session Bean (PSSB)
- A PSSB is a bean that uses the partitioning facility framework to create partitions according to your policy
  - Implements the PartitionHandlerLocal interface
  - Calls the High Availability Manager to create and activate partitions




A partitioned application contains one or more Partitioned Stateless Session Beans. These beans implement the PartitionHandlerLocal interface, and instruct the partitioning facility as to how partitions should be created and how requests should be mapped to partitions. The WebSphere Extended Deployment runtime identifies partitioned applications by looking for these beans.

# Deploying a partitioned application

- Run *ejbdeploy* against the EAR file to deploy the beans as usual
- Then run *wpfstubutil* to update the EJB stubs to reflect partition routing
- Order is important
  - Do not redeploy EJBs after running *wpfstubutil*




An extra step is required to deploy a partitioned application. Use the ‘ejbdeploy’ tool to deploy the enterprise beans, as you would for any other application. Then run the ‘wpfstubutil’ command, which modifies the EJB stubs to reflect the partitioned routing scheme. It is important not to redeploy the EJBs after calling ‘wpfstubutil’, because the partitioning-specific changes will be undone, requiring you to run ‘wpfstubutil’ again.

# Controlling partitions with wpfadmin

- Wpfadmin is a command-line utility for managing a partitioned environment using JMX
  - Wraps the wpfadmin.pty Jython script
  - Users are encouraged to use wpfadmin as an example and implement their own scripts




WebSphere Extended Deployment provides a command line tool, ‘wpfadmin’, for managing a partitioned application environment. Wpfadmin allows you to view the location of partitions within your cluster, to move partitions from one server to another, and to configure the High Availability Manager, among other things. Wpfadmin is an interface to a Jython script, wpfadmin.pty, that is well documented and makes a useful example for implementing your own partitioning facility management scripts.

# Partition aliases

- Partition aliases are a new feature in WebSphere Extended Deployment V6.0
- Using aliases gives you more flexibility in configuring and managing partitions
- Aliases can refer to one or more actual partitions
- Created programmatically using *PartitionDefinition.setPartitionAlias()*



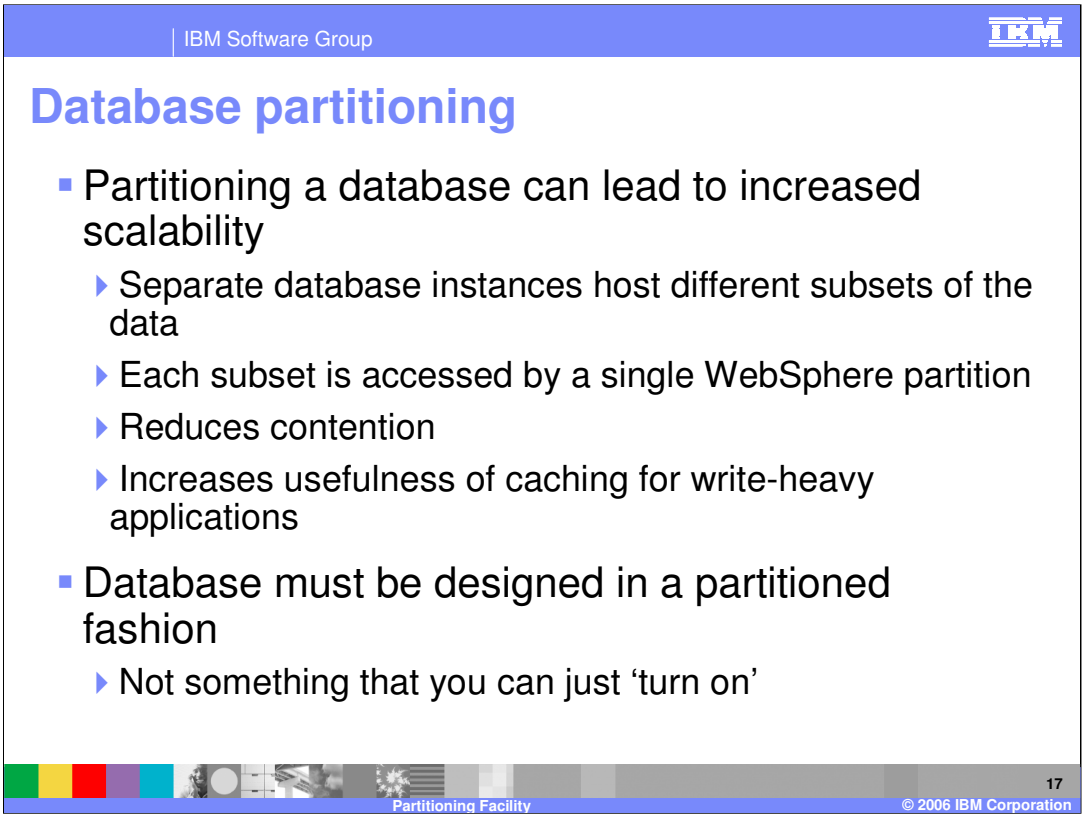

Partition aliases give you increased flexibility in how client requests are mapped to partitions. It allows you more flexibility in programming because partition names can be changed later if client code is using aliases. Partitions also give you the ability to group partitions within the same context, or to route requests to a partition based on a value other than the partition name.

## Section

***Database partitioning***




This section will discuss database partitioning.




In addition to partitioning your application, partitioning the database that your application accesses can drastically increase scalability. Database partitioning involves creating a separate database instance for the subset of data that will be accessed by each partition. This gives each partition exclusive access to the data that it will be using, reducing database contention and giving the application the freedom to more aggressively cache values in memory. It also gives you the ability to easily scale the database across multiple servers, rather than implementing a database clustering solution. Database partitioning requires careful planning and consideration at the time of application design; it is not a feature for which there is a simple on-off switch. Also when DB2® data sharing is implemented, the potential gain is greatly reduced due to the efficiency already found with DB2 data sharing.

# Data source configuration

- Accessing a partitioned database requires a proxy data source
  - Special kind of data source that lets the application indicate which data source to use per-transaction
  - JNDI name of the target data source must be specified at the beginning of each transaction
- Configure a data source for each database partition
- Create a proxy data source to encapsulate the regular data sources
  - Use the “proxy JDBC provider” to create the proxy data source
  - Different implementation than the proxy data source V5.1




WebSphere Extended Deployment enables applications to access partitioned databases by utilizing a special Data Source, called a Proxy Data Source. A Proxy Data Source allows an application to specify the name of another Data Source to use for each database transaction. The application therefore has the ability to dynamically access different database instances based on the partition context of each request.

# Database partitioning restrictions

- Currently only supports DB2 JDBC drivers and the Oracle type 4 (thin) driver
- CMP beans must be invoked using the local interface
- “Test connection” in Administrative Console will fail for Proxy Data Sources




WebSphere Extended Deployment currently only supports database partitioning when using the DB2 JDBC drivers or the Oracle Type 4 JDBC driver. It is also important to keep in mind that beans using container-managed persistence must be invoked using their local interfaces. Also, the “test connection” feature of the Administrative Console does not support Proxy Data Sources.

## Section

***Summary***




This section will summarize the presentation.

# Summary of performance advantages

- Data is local to a single server, so caches remain effective even with high write rates
- Optimistic locking is not necessary
  - Improves database performance by reducing overqualified updates
- Can reduce lock contention within an application
- Applications can take advantage of data affinity to batch together database writes for increased performance




Many potential performance advantages are inherent in the partitioned application model. Partitioned applications can cache data very aggressively, and these caches are effective even with high write rates. The need for overqualified updates introduced by optimistic locking is reduced when using partitioning, since each partition has exclusive access to a database instance. Partitioning can also help increase parallelism within an application, because lock contention within the application server itself can also be reduced. Lastly, applications can be designed to take advantage of the fact that they exclusively handle requests for certain data sets by performing batched database writes to decrease load on the database.

# Getting started with partitioning

- Partitioning facility user's guide
  - Included in the WebSphere Extended Deployment information center
  - Covers managing, programming, performance tuning, and more
- Several sample applications provided with WebSphere Extended Deployment
  - Samples cover all types of partitioning
  - Found in <INSTALL_ROOT>/installableApps
- Partitioning API JavaDoc
  - Found in <INSTALL_ROOT>/web/xd/apidocs/index.html




Several helpful resources are available to help you get started with the partitioning facility. The partitioning facility user's guide is included in the WebSphere Extended Deployment Information Center, and thoroughly covers developing, managing, and tuning partitioned applications. You can also find several samples of partitioned applications in the 'installableApps' directory after installing WebSphere Extended Deployment. Lastly, the Javadoc installed with the product is the official documentation for the partitioning API.

# Summary

- WebSphere Extended Deployment includes support for 'partitioned' applications
- Partitioning eliminates some of the overhead traditionally associated with clustering
  - Data contention, caching, data replication
- Provides near-linear scalability for high transaction volume applications




In summary, the partitioning facility in WebSphere Extended Deployment is a programming framework and runtime infrastructure that enables you to build high-performance J2EE applications that can scale efficiently on distributed hardware. Partitioning addresses the drawbacks typically associated with large-scale clustering, such as data contention, distributed caching, and data replication.

# 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.