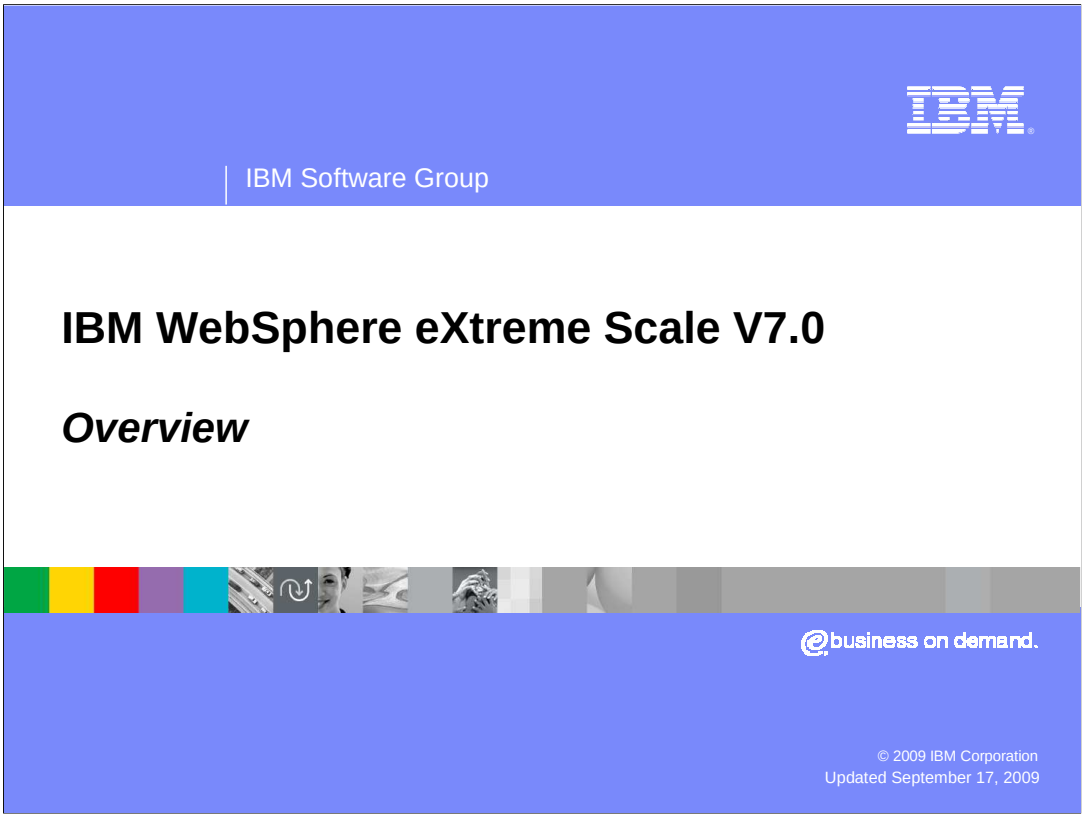
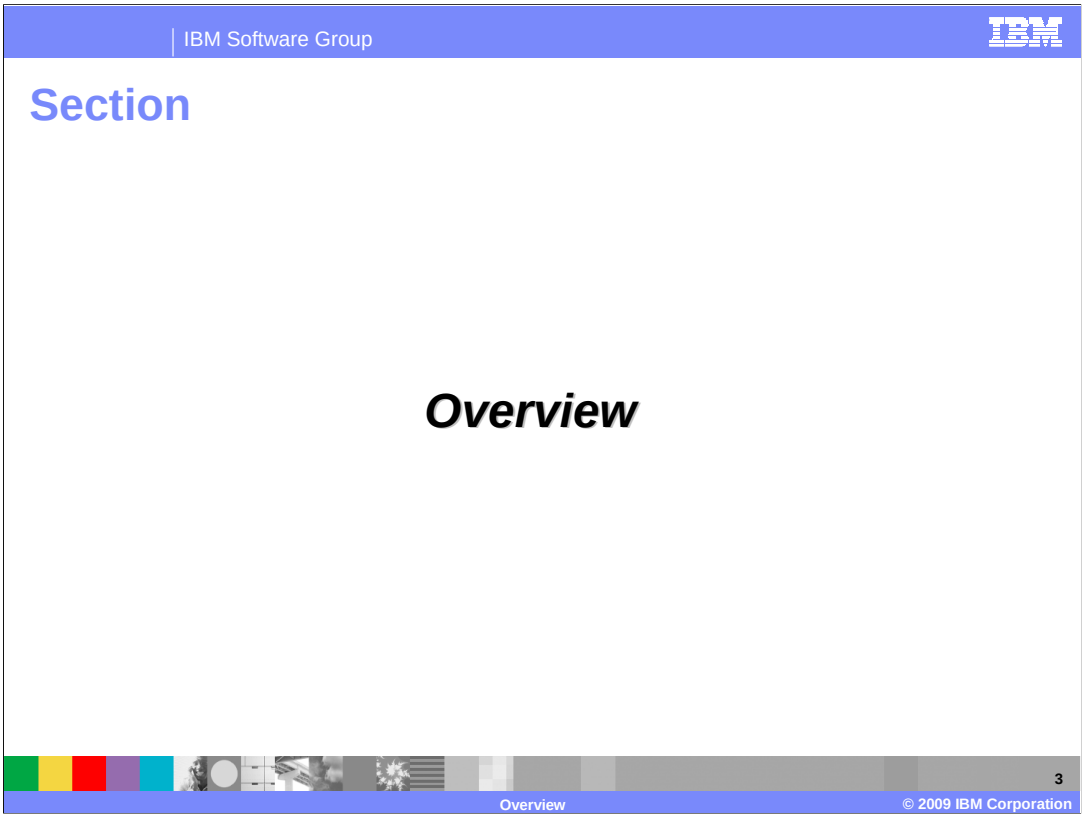


This presentation will cover WebSphere® eXtreme Scale V7.0, a high performance caching platform for conducting transaction processing, formerly known as Extended Deployment DataGrid or ObjectGrid.

Agenda

- WebSphere eXtreme Scale overview
- eXtreme Scale configuration
- eXtreme Scale topology options
- Summary

The agenda is to first introduce WebSphere eXtreme Scale, then briefly cover configuration and topology options.



This section will provide an overview of the WebSphere eXtreme Scale technology.

WebSphere eXtreme Scale overview

- WebSphere eXtreme Scale operates as an in-memory data grid that dynamically caches, partitions, replicates, and manages application data and business logic across multiple servers
- Performs massive volumes of transaction processing with high efficiency and linear scalability
- Provides qualities of service such as
  - Transactional integrity
  - High availability
  - Predictable response times

WebSphere eXtreme Scale operates as an in-memory data grid that dynamically caches, partitions, replicates, and manages application data and business logic across multiple servers.

The product enables data-intensive business applications to process massive volumes of transactions with high efficiency and linear scalability.

It provides transactional integrity and transparent failover to ensure high availability, high reliability, and predictable response times.

WebSphere eXtreme Scale overview

- Can be used as a powerful distributed cache to speed application access to data
- Can be backed by hardened storage
  - Cache loader interface enables automatic storage and retrieval of data according to user preferences to the technology of your choice (like a database)
- Can automatically replicate data either synchronously or asynchronously to ensure data availability and lower management burden
- Securable using Java™ authentication and authorization service (JAAS) API

WebSphere eXtreme Scale can be used as a powerful distributed cache with data stored in memory to speed application access to data.

The cache loader interface enables you to implement a class that uses the hardened storage technology of your choice, such as a database, as a backing store for the cache. The cache loader can also be used to retrieve data, so that all data does not have to be in-memory at one time.

WebSphere eXtreme Scale can ensure data availability and lower the management burden with automatic replication of data.

Additionally, an eXtreme Scale grid instance is securable using the standard Java Authentication and Authorization Service API.

Scalability

- eXtreme Scale can grow to hold hundreds of JVMs of data for very large data sets
  - Scales nearly linearly with additional hardware
  - Supports thousands of concurrent clients
  - Supports 32-bit or 64-bit JVMs
  - eXtreme Scale clusters can be replicated and partitioned for fault-tolerance and high performance
- Cluster members communicate with each other using an ultra-high speed pub sub system
  - High performance and low processor overhead

An eXtreme Scale grid is highly scalable, supporting a local cache within a single Java virtual machine, all the way up to a fully replicated cache distributed across numerous cache servers. As data volumes grow or as transaction volume increases, additional servers can be added to store the additional data and ensure consistent application access.

eXtreme Scale technology can be spread across clusters throughout an entire enterprise to guarantee high availability. If a primary server fails, a replica is promoted to primary automatically to handle fault tolerance and ensure high performance. To facilitate this, eXtreme Scale cluster members communicate with each other using a highly optimized, dedicated publish and subscribe messaging system.

Working with WebSphere eXtreme Scale

- Transactional support
  - Operations take place within the scope of a transaction for consistency
  - 1-phase commit supported
- Customizable cache life cycle features
  - Declaration, configuration, size management, cache loading, and eviction policies
- ObjectGrid query API
  - Flexible query engine for retrieving Entities and Java objects

As more and more data is stored in the grid, WebSphere eXtreme Scale provides a high performance, transactional cache facility that will allow you to work with ease with your data. It is highly customizable. For example, interfaces are provided for implementing custom cache loading, size management, and invalidation schemes. WebSphere eXteme Scale also provides powerful query mechanisms for retrieving objects from the cache using non-key attributes. The ObjectGrid query API allows select type queries over an Entity or Object-based schema using the ObjectGrid query language.

HTTP session persistence

- HTTP Sessions can be replicated across servers using eXtreme Scale
  - A servlet filter that enables session replication can be inserted into any Web application
  - Provides a session persistence approach that is independent of the WebSphere cell infrastructure
- Older versions of WebSphere products can use eXtreme Scale as an upgraded session persistence mechanism
- Non-WebSphere servers (such as Geronimo or JBoss) can also use this servlet filter

WebSphere eXtreme Scale can be used to persist and set up your HTTP session data. Basically, you can override the default session manager in the base application server to provide HTTP session management capabilities for an associated application. The eXtreme Scale session manager can create HTTP sessions and manage the life cycle of HTTP sessions that belong to the application. This life cycle management includes the invalidation of sessions based on timeout or an explicit servlet or JavaServer Pages (JSP) call and the invocation of session listeners that are associated with the session or the Web application.

Runtime environment support - distributed platforms

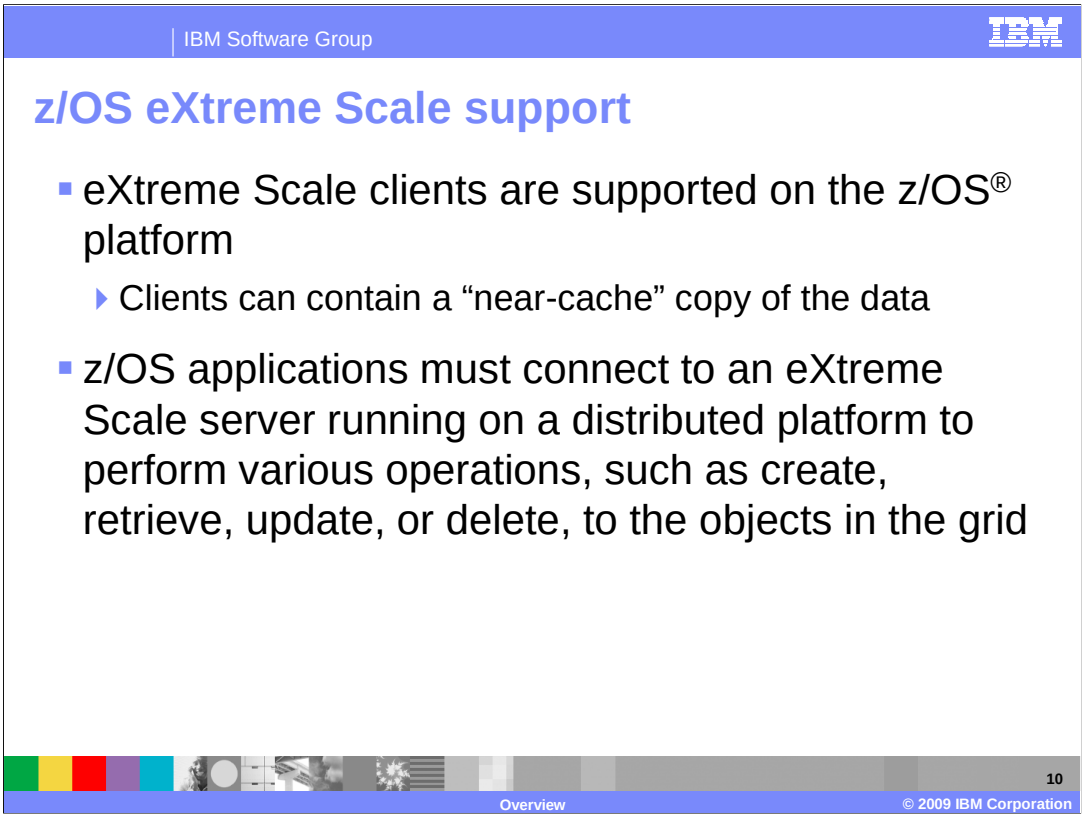
- eXtreme Scale caches are supported in
  - WebSphere Application Server Network Deployment V6.1 (or greater) runtime environments
  - WebSphere Virtual Enterprise V6.1 (or greater)
  - WebSphere Extended Deployment Compute Grid V6.1 (or greater)
- eXtreme Scale can also be installed ‘stand-alone’
  - Runtime can start within any Java EE or J2SE 1.4.2 or higher Java Virtual Machine (JVM)

eXtreme Scale technology works well in conjunction with WebSphere Virtual Enterprise V6.1 and WebSphere Extended Deployment Compute Grid V6.1.

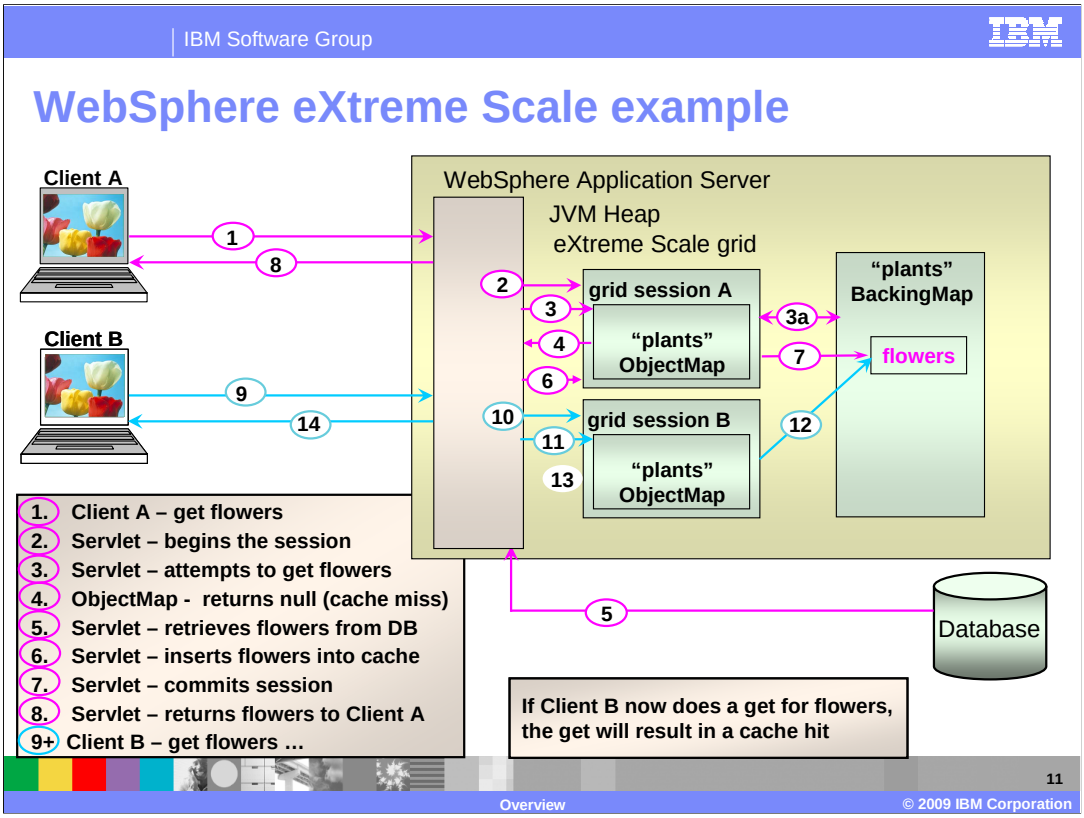
eXtreme Scale caches can also be hosted on WebSphere Application Server Network Deployment version 6.1 (or higher) servers.

The main benefit of this configuration is that you can more easily manage your environment using the administrative capabilities available in the WebSphere Network Deployment product.

Another option is to use stand-alone servers running the JVM of your choice to host the eXtreme Scale grid. The runtime requires a Java 1.4.2 or higher Java Virtual Machine to execute.



WebSphere eXtreme Scale V7.0 provides client support access on the z/OS platform. Clients can contain a near-cache copy of the BackingMap. However, an application deployed to z/OS will require access to an eXtreme Scale server. The z/OS eXtreme Scale client must connect to an off-platform eXtreme Scale server, such as Linux®, in order to perform various operations to objects in the grid.



This example shows an application that requests an object that is not in the grid. The object is retrieved and then inserted in the grid, so future requests for that object can be fulfilled directly from the eXtreme Scale grid. **(Atl text)**

The basic sequence is; client A sends a request for the flowers object to the servlet. The eXtreme Scale grid and BackingMap have already been created, so when a user initially interacts with the grid a session is established. In this case, the servlet begins a session for Client A. The servlet then attempts to "get" the flowers object from the eXtreme Scale grid. As this is the first time a request for the flowers object has been made, it will result in a miss because it is not yet present in the grid. So, the servlet will retrieve the flowers object from a back end database and use an "insert" to place the object into the BackingMap in the grid. At this point, the session's ObjectMap is committed to the BackingMap and the flowers object is returned to Client A. A similar sequence will occur for Client B when it requests the flowers object, only this time the object already resides in the BackingMap, so this request is fulfilled directly from the eXtreme Scale cache.

Working with eXtreme Scale data

- An eXtreme Scale grid contains one or more map-like objects (ObjectMaps)
  - Interface similar to java.util.Map
  - ObjectMaps support all of the expected map methods
    - put(), get(), insert(), update(), and so on.
- Objects are stored as map entries (key value pairs)
  - Can be entered into the map by the application
  - Can be loaded from an external source using a custom loader class
- Can be indexed on key or attribute values

Java objects are stored in an eXtreme Scale grid using key value pairs within map objects called ObjectMaps. Data can be put into and retrieved from an ObjectMap within the scope of a transaction using all of the typical map-like methods through an interface similar to java.util.Map. The map can be populated directly by the application, or it can be loaded from a back-end store by implementing a custom cache loader class.

Section

eXtreme Scale configuration

This section will cover basic eXtreme Scale configuration.

Configuration options

- eXtreme Scale instances can be configured programmatically or using Extensible Markup Language (XML) files
  - Sample configuration files are provided with the WebSphere eXtreme Scale installation
- XML files define the Java implementations that should be used and how they are associated
- To create an eXtreme Scale grid using an XML file:

```
ObjectGridManager myObjectGridManager =
  ObjectGridManagerFactory.getObjectGridManager();

ObjectGrid myObjectGrid =
  objectGridManager.createObjectGrid("newGrid", "newgrid.xml");
```


To cache objects using an eXtreme Scale grid, you must first create an ObjectGrid instance within your application. The instance can be configured programmatically, or created based on configuration data stored in a descriptor XML file. The code snippet shown here illustrates how to instantiate the “newGrid” ObjectGrid based on a configuration file, “newgrid.xml”, using the ObjectGridManager class. Note that the term ObjectGrid is still used when referring to eXtreme Scale grid object.

Configuration files

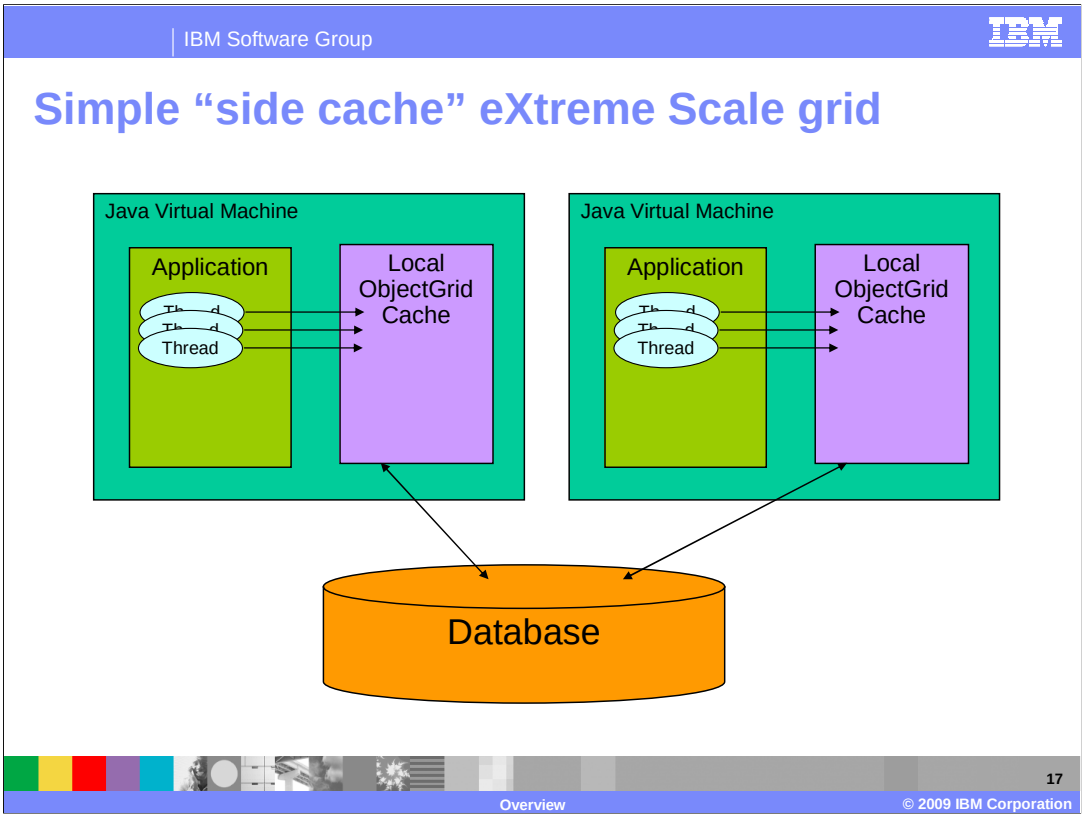
- ObjectGrid configuration file
  - Defines ObjectGrid instances, associated maps, and plug-ins
- eXtreme Scale servers can also bootstrap to a running server and read the configuration over TCP/IP

ObjectGrid configuration files are used for configuring the eXtreme Scale grid itself, but they are also used to define the associated maps, or what kind of data is being stored in the grid, and any plug-ins you might be using. The configuration information can be read either from the local file system or you can bootstrap from a running server and read its configuration. This is a good way to ensure that all of your grid servers use the same configuration.

Section

eXtreme Scale topology options

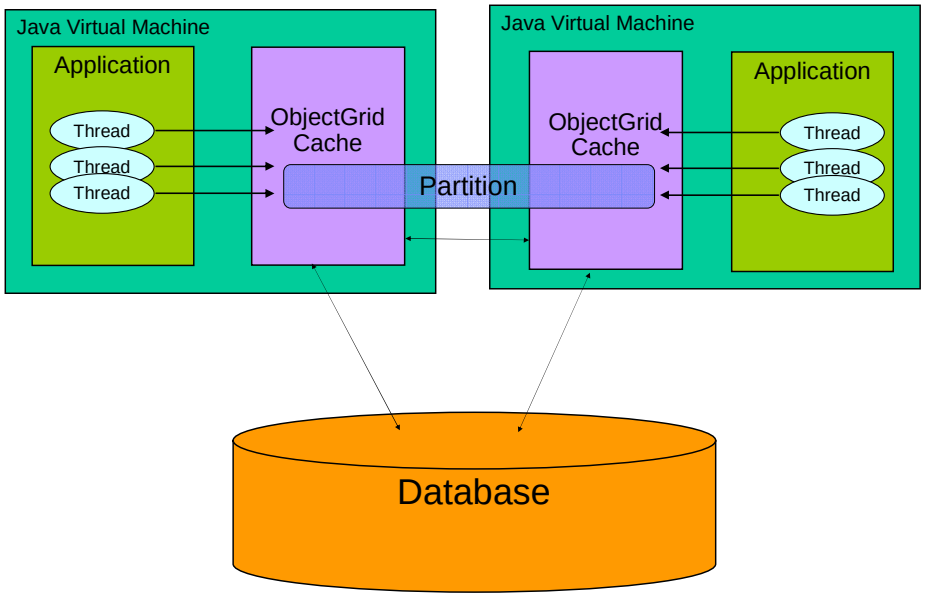
There are many different ways to integrate eXtreme Scale into your environment. This section will discuss several of the eXtreme Scale topology options.



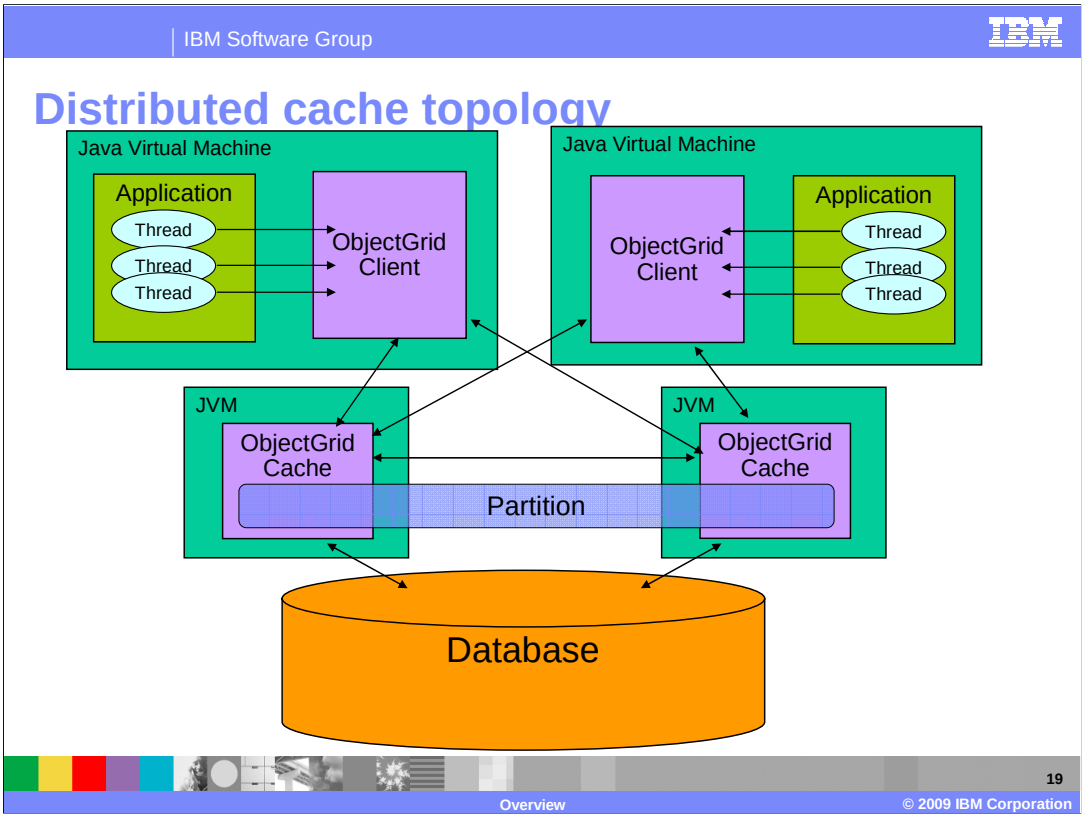
In the simplest case, WebSphere eXtreme Scale can be used as an in-memory “side cache”. This illustration shows multiple application servers loading data from the same database and caching it locally. The “side cache” topology can be used to provide consistent, transactional access to temporary data within a single Java Virtual Machine. This can especially benefit high-concurrency applications where multiple threads need to access and modify transient data. The data kept in a local eXtreme Scale grid can be indexed and retrieved using eXtreme Scale’s query support.

(Alt text – WebSphere eXtreme Scale is installed locally in the same Java Virtual Machine as the application logic. Each application thread will only access the local grid instance to store or retrieve data from its cache. Each local grid instance will access the database to retrieve data that it does not contain or to store changes.)

Collocated application and cache topology



In this topology, the application logic runs in the same Java Virtual Machine as the data in the grid. However, the data stored in the grid is spread across all the JVMs that have WebSphere eXtreme Scale installed and configured. This topology can reduce the load on your backend data store when the requested data is in the near cache or is stored on another server in the grid. Also, replica data can sit in a JVM separate from the primary data to ensure fault tolerance and high availability.



eXtreme Scale also supports more complicated topologies, supporting highly available distributed, partitioned and replicated cache which can scale to thousands of containers containing terabytes of data. Distributed eXtreme Scale caches offer increased performance, availability and scalability. Local and distributed eXtreme Scale topologies both provide the same application programming model for interacting with the cache.

In this distributed cache topology the application logic runs on application servers separate from the eXtreme Scale grid servers. The application servers host a grid client which can communicate with the grid servers to access data from the far cache. The data stored in the grid is spread across all the Java Virtual Machines that have WebSphere eXtreme Scale installed and configured.

Section

Summary

This section will provide a summary of this presentation.

Summary

- WebSphere eXtreme Scale provides a high performance, scalable cache system capable of performing massive volumes of transaction processing
- It operates as an in-memory data grid that dynamically caches, partitions, replicates, and manages application data and business logic across multiple servers
- eXtreme Scale features can be customized by implementing custom Java classes
  - Cache loading, invalidation, and more are extensible

WebSphere eXtreme Scale provides a high-performance, scalable, transactional cache technology. It operates as an in-memory data grid that can dynamically cache, partition, replicate, and manage application data and business logic across multiple servers.

The eXtreme Scale cache is designed to be highly extensible, so that you can implement a cache system that meets the needs of your specific environment.

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_WXS70_Overview.ppt

This module is also available in PDF format at: ../WXS70_Overview.pdf

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

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

WebSphere z/OS

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

J2SE, Java, JavaServer, JSP, JVM, and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or 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 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.

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.