This presentation will introduce ObjectGrid, a new cache framework in WebSphere Extended Deployment V6.

Agenda

- ObjectGrid overview
- Configuring ObjectGrid
- ObjectGrid topologies

The agenda is to provide an overview of ObjectGrid, and then briefly cover ObjectGrid configuration and topology options.

Section

Overview

This section will provide an overview of ObjectGrid technology.

ObjectGrid overview

- ObjectGrid is a high performance cache framework for storing Java™ objects
- Supports a variety of cache topologies
  - Scalable from a local cache to a distributed, partitioned cluster of hundreds of object grid nodes
    - Peer Network of Object Cluster caches
    - Cross platform with hundreds of servers on distributed platforms
- Transaction support
  - Operations can take place within the scope of a transaction for consistency
  - 1-phase commit supported

ObjectGrid is a WebSphere Extended Deployment technology that provides a high performance, transactional cache for Java objects. ObjectGrid is highly scalable, from a local cache as a single instance all the way up to fully replicated caches distributed across numerous nodes. These object grid nodes can be either in a peer to peer network or use a server located on a platform other than z/OS.

Overview (cont.)

- Customizable cache lifecycle features
  - Declaration, configuration, invalidation, size management, cache loading
- Can be backed by hardened storage
  - Cache loader interface enables hardening using storage technology of your choice (like a database)
- Securable using Java Authentication and Authorization Service (JAAS) API
- Support for hierarchical, tag-based invalidation and custom eviction policies

An ObjectGrid is highly customizable. For example, interfaces are provided for implementing custom cache loading, size management, and invalidation schemes.

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. An ObjectGrid instance can also be secured using the standard Java Authentication and Authorization Service API. Hierarchical, tag-based invalidation using keywords is another feature of an ObjectGrid cache.

Scalability

- ObjectGrid can scale to hold hundreds of data caches for very large data sets
  - ObjectGrid clusters can be replicated and partitioned for fault-tolerance and high performance
  - Scales nearly linearly with additional hardware
  - Supports thousands of concurrent clients
  - Supports basic and 64-bit Java Developer Kits
- Cluster members communicate with each other using an ultra-high speed publish/subscribe system
  - High performance and low processor overhead

ObjectGrid technology scales from a local cache within a single server to a vast array of distributed and scalable data services spread across ObjectGrid clusters throughout an entire enterprise. ObjectGrid cluster members communicate with each other using a highly optimized, dedicated publish/subscribe messaging system.

Runtime environment support

- ObjectGrid Caches supported in:
  - WebSphere Extended Deployment V6.0 (or greater) runtime environment
  - WebSphere Application Server V5.0.2 (or greater) runtime environment
    - Include wsobjectgrid.jar in application classpath
- ObjectGrid Libraries provided with
  - WebSphere Extended Deployment
  - Not WebSphere Application Server

While ObjectGrid technology is provided only with WebSphere Extended Deployment, ObjectGrid caching is also supported in a WebSphere Application Server Network Deployment version 5.0.2 server by including the ObjectGrid library, wsobjectgrid.jar, in your application classpath.

Runtime environment support (cont.)

- ObjectGrid can be installed without WebSphere Extended Deployment using the “mixed server environment” installation option on distributed platforms
- ObjectGrid can also be run ‘stand-alone’ outboard
  - Server runtime requires J2SE 1.4.2 or higher JVM
  - Client and local runtime requires J2SE 1.3.1 or higher JVM
  - Not on z/OS

ObjectGrid can run in a stand-alone JVM or other application server product by using the ObjectGrid.jar file. The client/server object grid model employs a dedicated single-purpose object grid server. This server does not run in a WebSphere server; and it is itself a scaled-down server in its own right. The stand-alone object grid server is not available on z/OS in the 6.0.1 release. An application deployed to WebSphere Extended Deployment V6.01 for z/OS that requires access to an object grid server must use an off-platform object grid server, running, for instance, on Linux®.

Working with object grid data

- An ObjectGrid contains one or more Map-like objects (ObjectMaps)
  - 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

Java Objects are stored in an ObjectGrid 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 usual Map-like methods. The Map can be solely populated by the application, or it can be loaded from a back-end store by implementing a custom cache loader class.

Section

ObjectGrid configuration

This section will cover configuring an ObjectGrid instance.

Configuration options

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

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

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


To cache objects using ObjectGrid, you must create an ObjectGrid instance within your application. The instance can be configured programmatically, or created based on configuration data stored in an XML file. The code snippet shown here illustrates how to instantiate an ObjectGrid based on a configuration file, using the ObjectGridManager class. You can learn about ObjectGrid configuration files by exploring the samples provided in the “optional libraries” directory after installing WebSphere Extended Deployment.

Sample configuration syntax

```
<objectGrids>
  <objectGrid name="ObjectGrid">

        <bean id="TransactionCallback"
        className="com.ibm.websphere.objectgrid.sample.HeapTransacti
        onCallback" />

        <backingMap name="employees" readOnly="false"
        pluginCollectionRef="employees" preloadMode="false"
        lockStrategy="optimistic"
        copyMode="COPY_ON_READ_AND_COMMIT" />
  </objectGrid>
</objectGrids>
```


This snippet shows what a simple ObjectGrid configuration looks like. The code specifies that the runtime use a custom TransactionCallback implementation by pointing to the implementation class. The grid will contain a Map named ‘employees’, and you can see that data preloading has been disabled and a locking strategy and copy mode have been set for the map.

Sample configuration syntax (cont.)

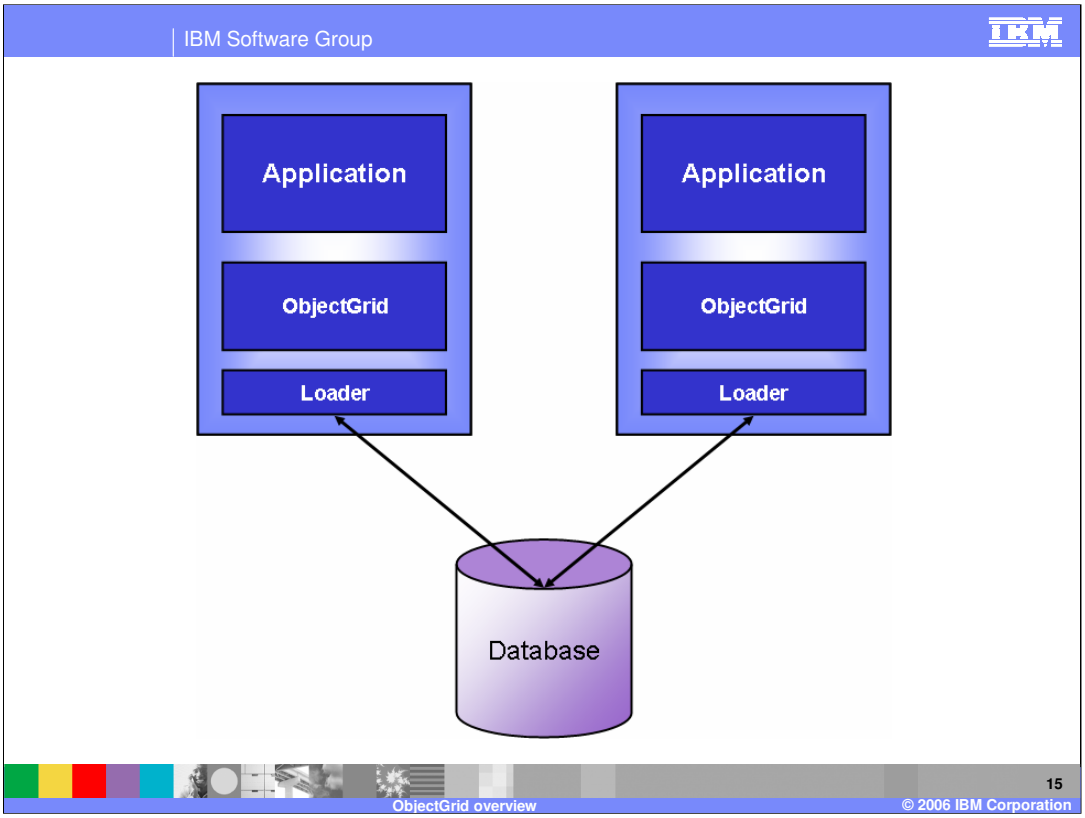
```
<backingMapPluginCollections>

  <backingMapPluginCollection id="employees">
    <bean id="Loader"
    className="com.ibm.websphere.objectgrid.sample.HeapCacheLoader">
      <property name="preLoadClassName" type="java.lang.String"
      value="com.ibm.websphere.objectgrid.sample.EmployeePreload"
      description="name of class that implements HeapPreloadData
      interface" />
    </bean>
    <bean id="ObjectTransformer"
    className="com.ibm.websphere.objectgrid.sample.EmployeeObjectTra
    nsformer" />
    <bean id="OptimisticCallback"
    className="com.ibm.websphere.objectgrid.sample.EmployeeOptimisti
    cCallbackImpl" />
  </backingMapPluginCollection>


</backingMapPluginCollections>
```


This snippet contains the remainder of the configuration that began on the previous slide, and it defines the custom plug-ins that will be used on the map named ‘employees’. This particular map uses custom Loader, ObjectTransformer, and TransactionCallback implementations. These plug-ins are all associated with a map, so if you have multiple maps within an ObjectGrid instance, they can use different sets of custom plug-ins.

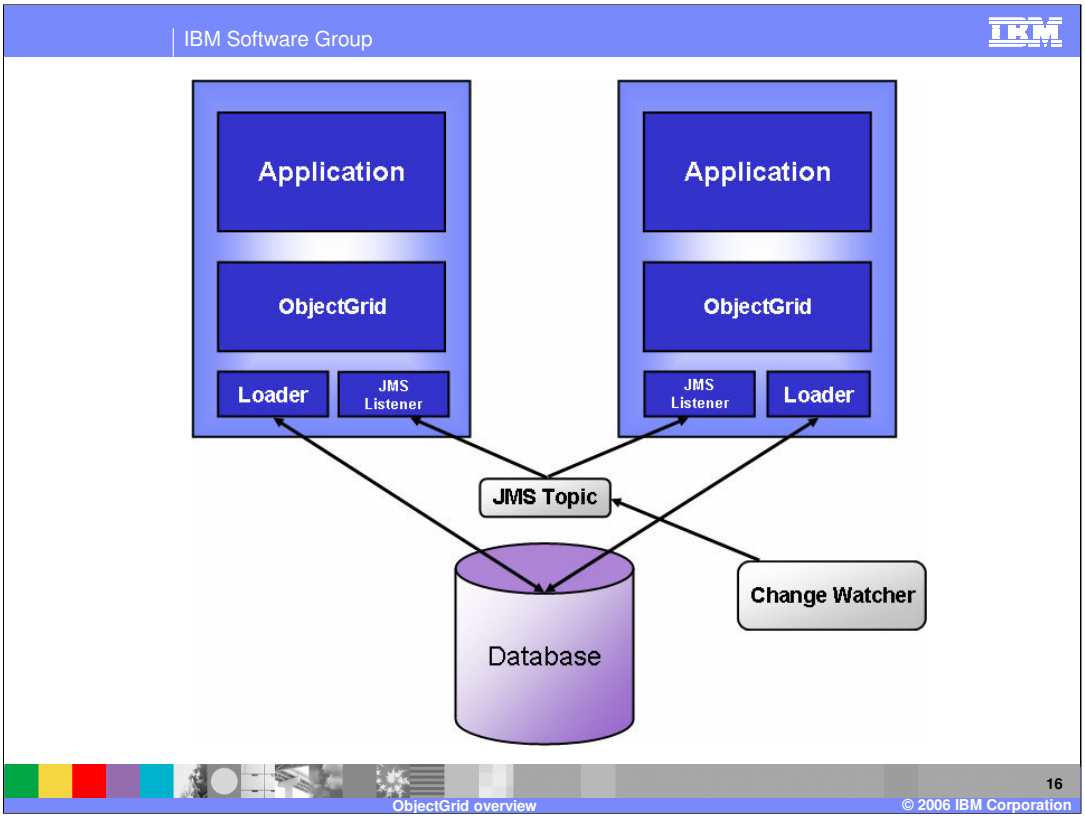
Section

ObjectGrid topology options

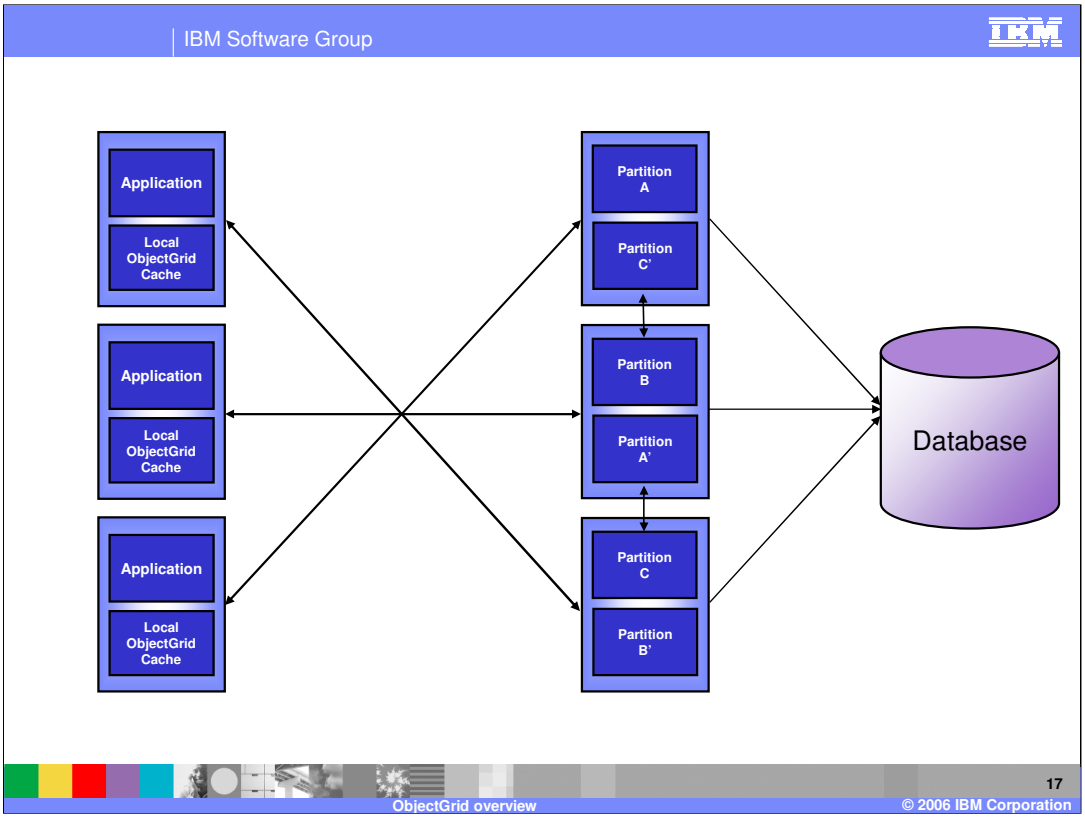
This section will discuss different ObjectGrid topology options.



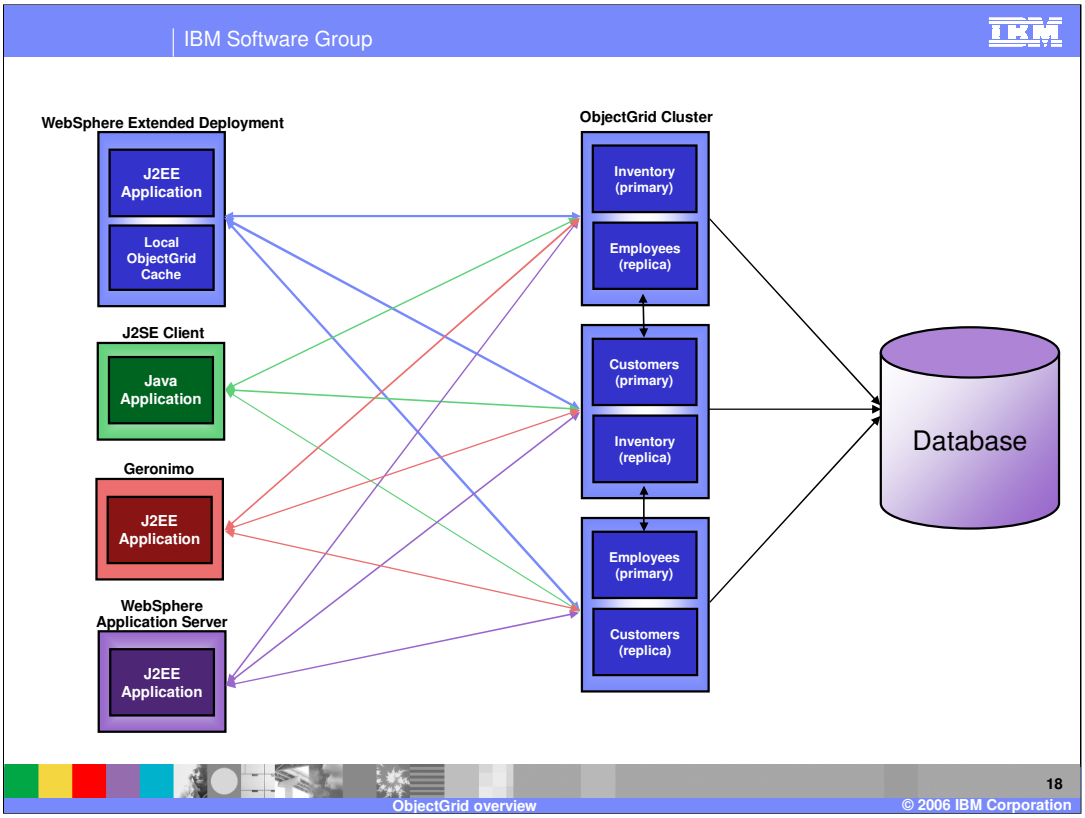
The first scenario shows multiple Application Servers loading data from the same database and caching it locally. In this scenario, each of the servers can have stale data in its local cache if the other server changes a value. Optimistic locking is used to ensure that stale data does not get written back to the database.



The scenario shown here is similar, except that a change watcher is employed to alert the servers whenever a change is made to the database by some other process. A lightweight JMS system is used for notification, in which both servers subscribe to a JMS topic that receives messages from the change watcher. When they receive messages on this topic, the servers invalidate the specified data. Again, optimistic locking is used to ensure that stale data is not written back to the database.



In this multi-tiered example, each application server contains an ObjectGrid instance that is used as a near cache. For objects that are not found in the local cache, a query can be made to the tier of ObjectGrid servers. This tier hosts a much larger set of data in memory, and Objects can be retrieved from this tier much faster than by accessing the database. This example shows a replicated and partitioned tier of ObjectGrid servers. Partitioning enables higher performance access to the data, and replication ensures availability of each partition. The ObjectGrid tier can be configured to use a Loader that queries the database when the requested data is not found in the cache.



This example illustrates a topology in which several different client types have access to a set of data being hosted by an ObjectGrid cluster. The three maps – Inventory, Customers, and Employees – each have a primary instance and a replica. Data changes are periodically sent from the primaries to the replicas to ensure that the replica remains up to date, and these replicas can serve client requests in case the primary fails.

Section

Summary

This section will provide a summary for this presentation.

Summary

- ObjectGrid is a new technology that provides a high-performance object cache for applications running on WebSphere Extended Deployment
- ObjectGrid features can be customized by implementing custom Java classes
  - Cache loading, invalidation, and more are extensible

ObjectGrid provides a new high-performance transactional cache technology for WebSphere Extended Deployment. Objects can be stored in and retrieved from the cache using map objects. The ObjectGrid cache is designed to be highly extensible, so that you can implement a cache system that meets the needs of your specific environment.

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.