IBM Software Group

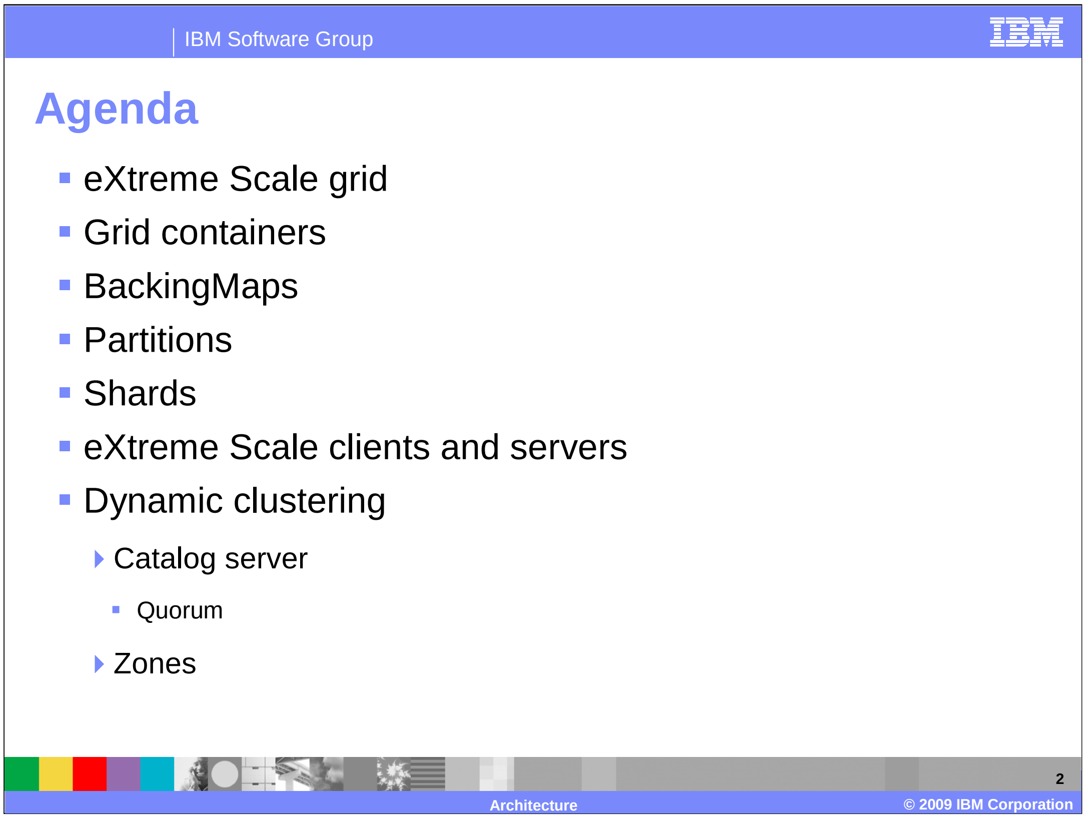
IBM WebSphere eXtreme Scale V7.0

Architecture

business on demand.

Updated July 6, 2009

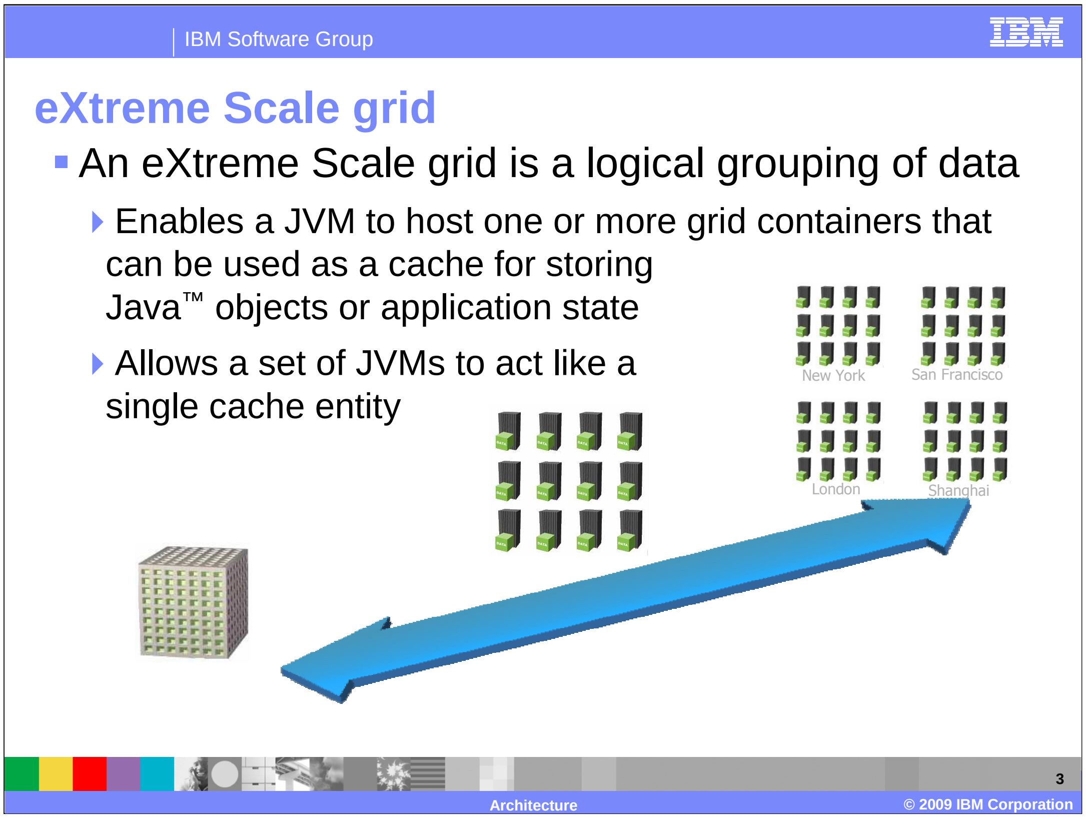
This presentation will cover the WebSphere® eXtreme Scale V7.0 architecture.



Agenda

- eXtreme Scale grid
- Grid containers
- BackingMaps
- Partitions
- Shards
- eXtreme Scale clients and servers
- Dynamic clustering
  - Catalog server
    - Quorum
  - Zones

The agenda is to cover the eXtreme Scale terminology in order to understand how an eXtreme Scale grid is structured and how applications use the grid.

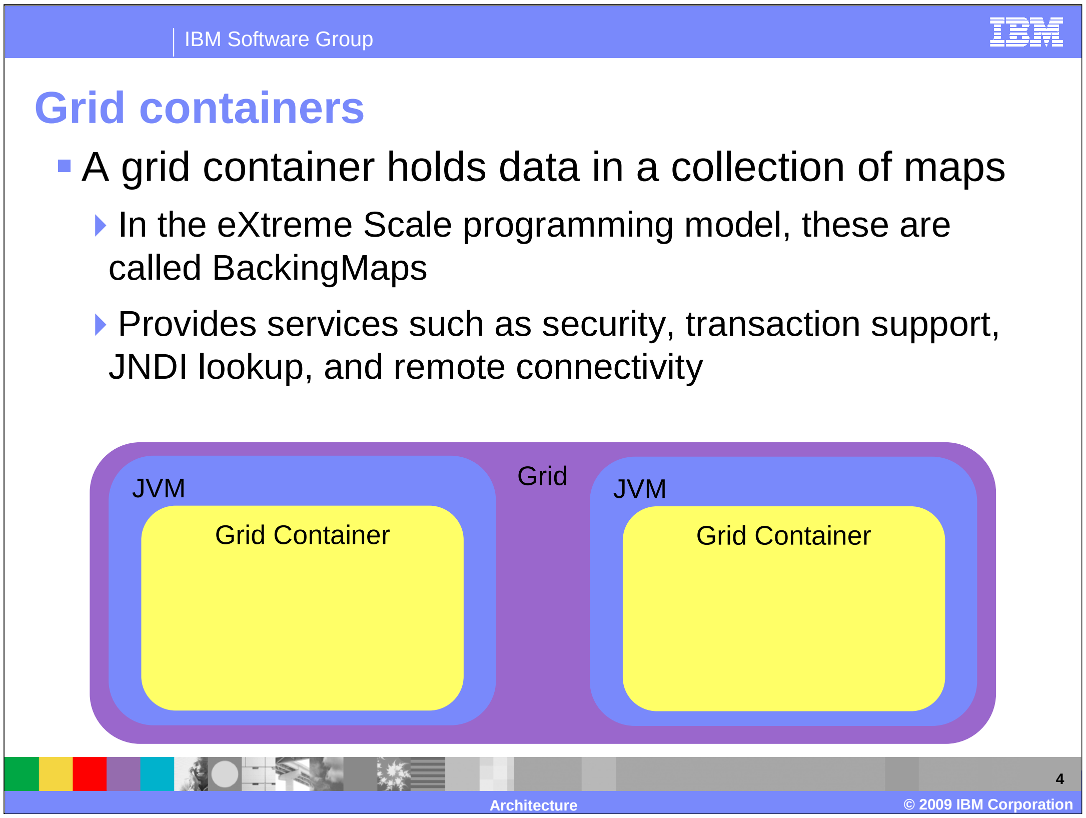

New York
San Francisco
London
Shanghai

eXtreme Scale grid

- An eXtreme Scale grid is a logical grouping of data
  - Enables a JVM to host one or more grid containers that can be used as a cache for storing Java™ objects or application state
  - Allows a set of JVMs to act like a single cache entity

WebSphere eXtreme Scale enables a Java Virtual Machine to host one or more grid containers that can be used as a cache for storing Java objects or application state. An eXtreme Scale grid is a self managed group of connected grid containers that act like a single cache entity.

Because grid containers can be added to the eXtreme Scale grid as more and more data needs to be stored, WebSphere eXtreme Scale provides a very scalable caching solution.



IBM Software Group

Grid containers

- A grid container holds data in a collection of maps
  - In the eXtreme Scale programming model, these are called BackingMaps
  - Provides services such as security, transaction support, JNDI lookup, and remote connectivity

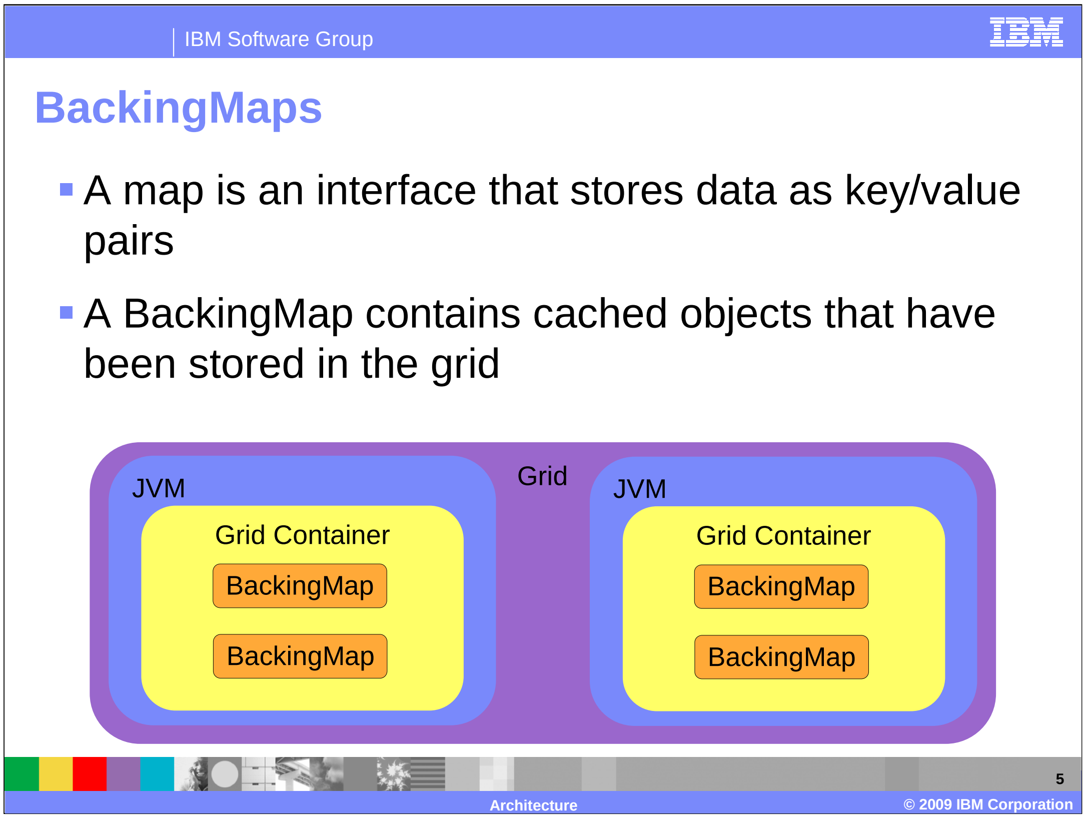
Grid
JVM
Grid Container
JVM
Grid Container

4

Architecture

© 2009 IBM Corporation

Much like J2EE containers, like Web or EJB containers, a grid container provides the grid application services such as security, transaction support, JNDI lookup service, remote connectivity and so forth. It houses data in a collection of maps called BackingMaps.

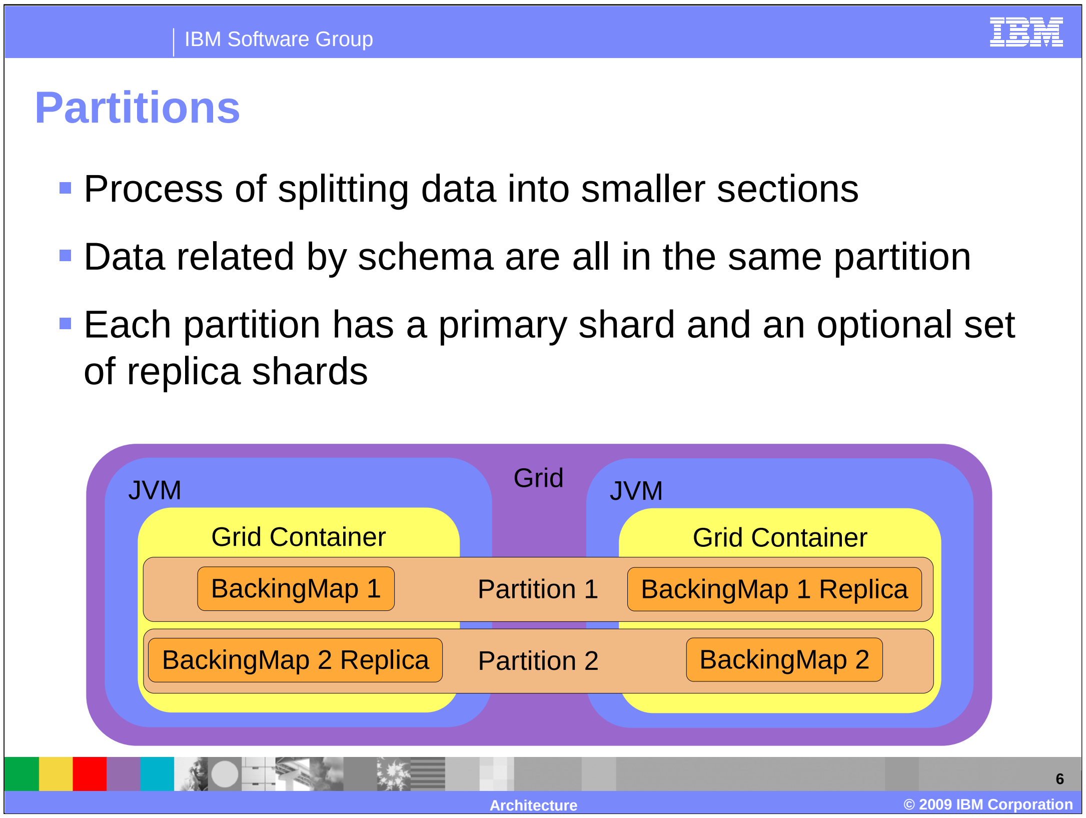


BackingMaps

- A map is an interface that stores data as key/value pairs
- A BackingMap contains cached objects that have been stored in the grid

Grid
JVM
Grid Container
BackingMap
BackingMap
JVM
Grid Container
BackingMap
BackingMap

A BackingMap is a type of map that contains cached objects that have been stored in the grid. Each BackingMap has it’s own loader instance. It will request any needed data that it does not contain from the loader, which in turn, will retrieve it from the back end data store.

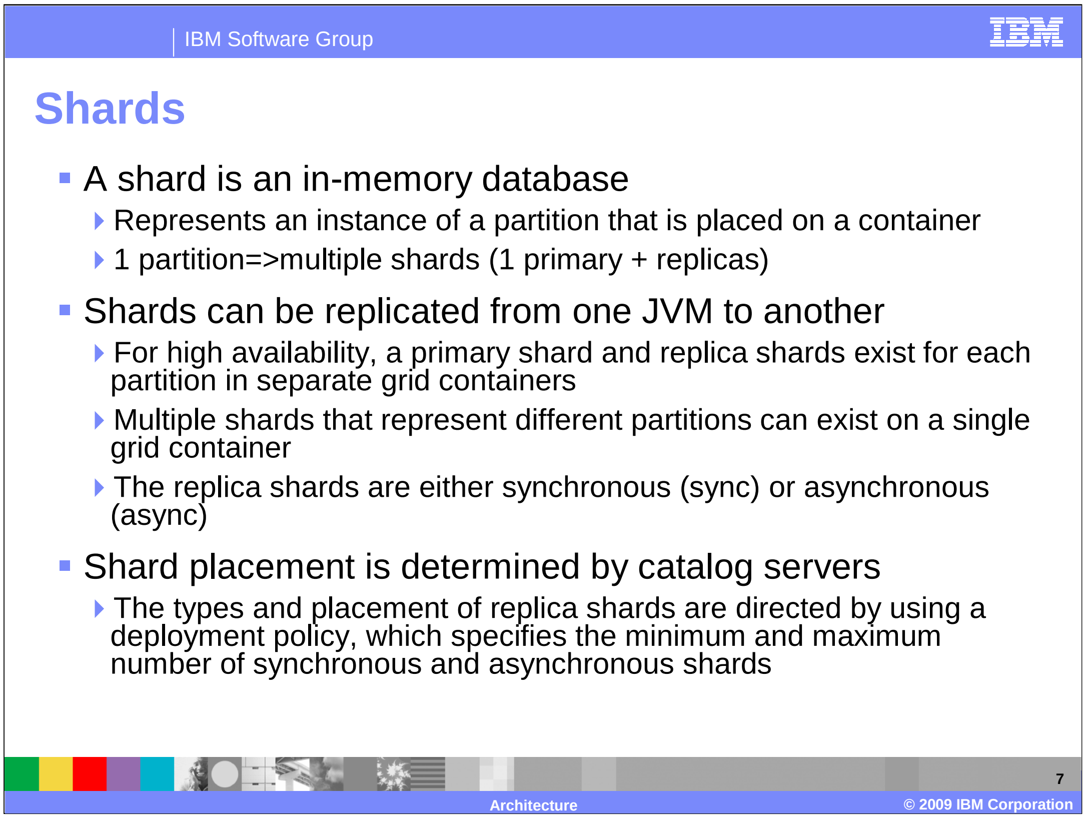


Partitions

- Process of splitting data into smaller sections
- Data related by schema are all in the same partition
- Each partition has a primary shard and an optional set of replica shards

Grid
JVM
Grid Container
BackingMap 1
Partition 1
BackingMap 1 Replica
BackingMap 2 Replica
Partition 2
BackingMap 2
JVM
Grid Container

Partitioning is the process of splitting data into smaller sections. It allows the grid to store more data than can be accommodated in a single Java Virtual Memory heap. The data partitioning is accomplished by using an application-defined schema. Each partition is made up of a primary shard and one or more replica shards that can be replicated to multiple servers. (Note that some applications may allow only one partition.)

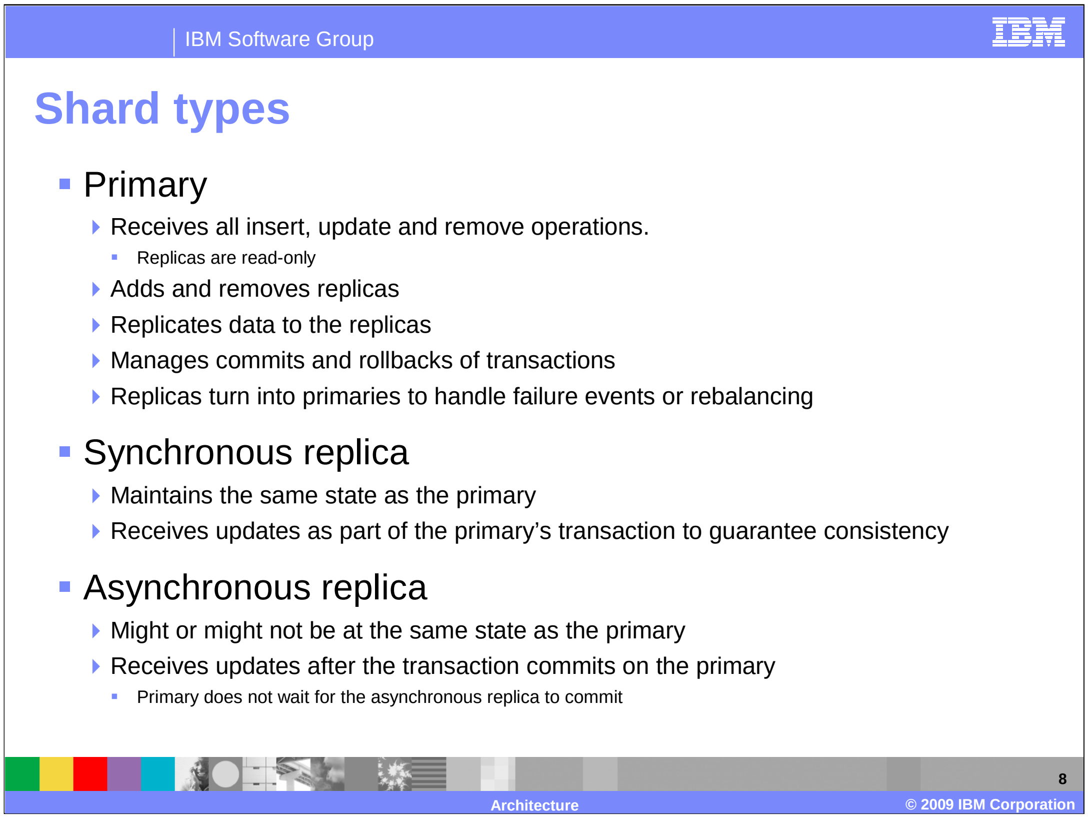


Shards

- A shard is an in-memory database
  - Represents an instance of a partition that is placed on a container
  - 1 partition=>multiple shards (1 primary + replicas)
- Shards can be replicated from one JVM to another
  - For high availability, a primary shard and replica shards exist for each partition in separate grid containers
  - Multiple shards that represent different partitions can exist on a single grid container
  - The replica shards are either synchronous (sync) or asynchronous (async)
- Shard placement is determined by catalog servers
  - The types and placement of replica shards are directed by using a deployment policy, which specifies the minimum and maximum number of synchronous and asynchronous shards

The term shard is used to define a single instance of a partition. Each partition is made up of a primary shard and zero or more replica shards. The shard distribution algorithm ensures that the primary and replica shards are never in the same container to ensure fault tolerance and high availability. Shards that represent different partitions can exist in a single grid container however.

Shard placement is the responsibility of the catalog servers. The type and placement of replica shards are directed by a deployment policy. So, as the grid members change and JVMs are added to or removed from the grid, the catalog server will redistribute shards accordingly to best fit the available JVMs. The ability of the grid to scale out or scale in provides tremendous flexibility to the changing nature of the eXtreme Scale infrastructure.



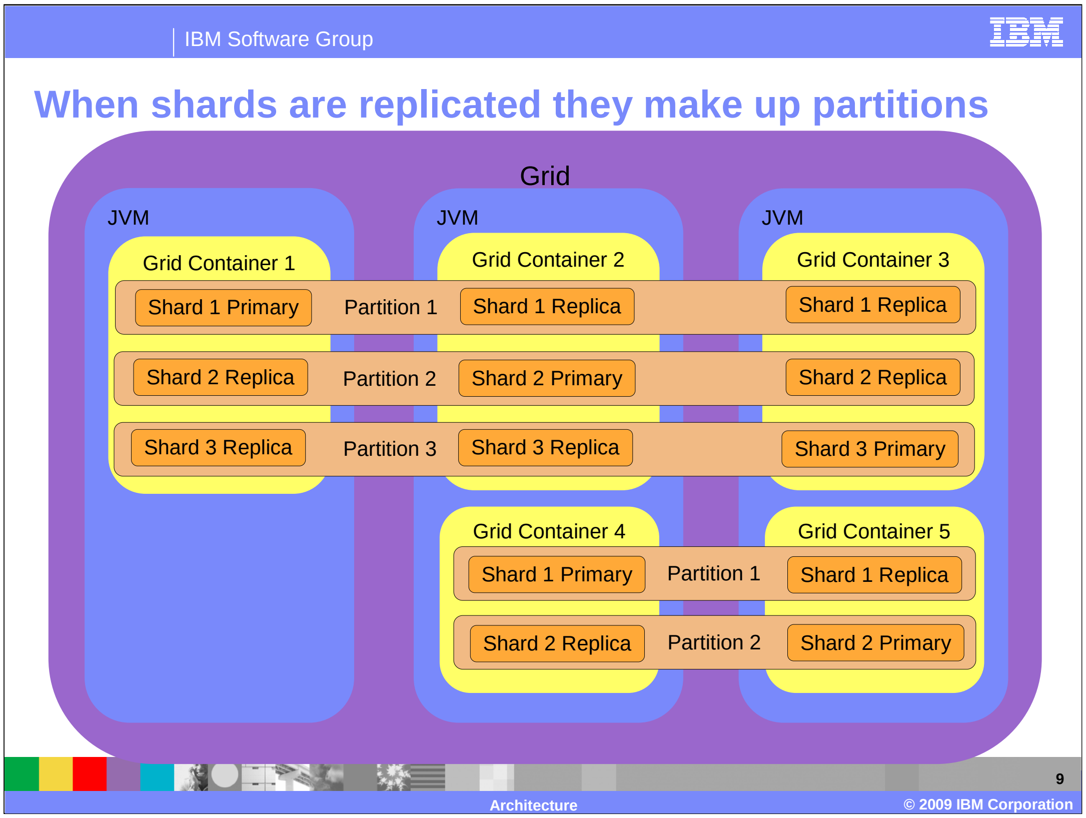
Shard types

- Primary
  - Receives all insert, update and remove operations.
    - Replicas are read-only
  - Adds and removes replicas
  - Replicates data to the replicas
  - Manages commits and rollbacks of transactions
  - Replicas turn into primaries to handle failure events or rebalancing
- Synchronous replica
  - Maintains the same state as the primary
  - Receives updates as part of the primary’s transaction to guarantee consistency
- Asynchronous replica
  - Might or might not be at the same state as the primary
  - Receives updates after the transaction commits on the primary
    - Primary does not wait for the asynchronous replica to commit

Primary and replica shards are balanced across containers. Replicas turn into primaries to handle failure events. This slide discusses some of the details of each shard type.

The synchronous replica shard data remains at the same level as the primary shard data. Whenever a transaction commits on the primary, the synchronous replica shard will update its contents. The primary shard waits until the replica shard responds that it got the data.

The asynchronous replica data shard is not guaranteed to remain at the same level as the primary shard data. When the primary shard sends data to the asynchronous replica, it does not wait for a response.



When shards are replicated they make up partitions

Grid

JVM

JVM

JVM

Grid Container 1

Grid Container 2

Grid Container 3

Shard 1 Primary

Partition 1

Shard 1 Replica

Shard 1 Replica

Shard 2 Replica

Partition 2

Shard 2 Primary

Shard 2 Replica

Shard 3 Replica

Partition 3

Shard 3 Replica

Shard 3 Primary

Grid Container 4

Grid Container 5

Shard 1 Primary

Partition 1

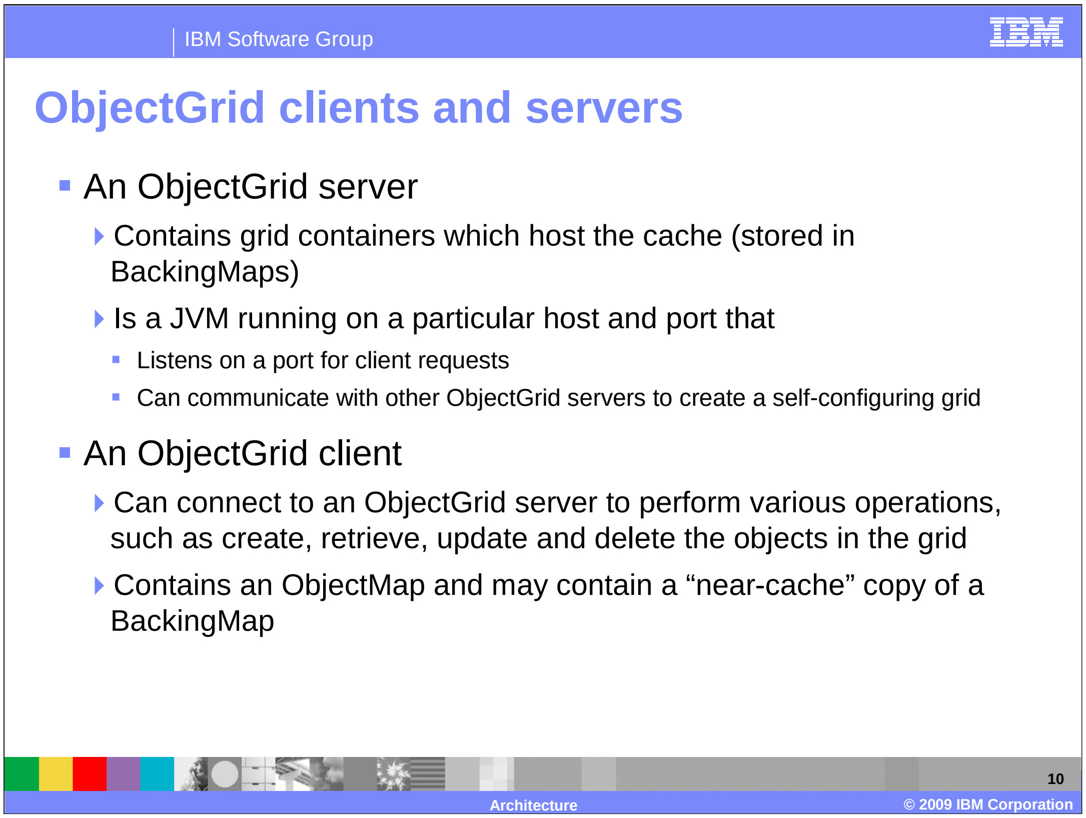
Shard 1 Replica

Shard 2 Replica

Partition 2

Shard 2 Primary

When shards are replicated they make up partitions. This figure shoes three partitions on the top with two replicas each and on the bottom two partitions each with a single replica, making 13 shards total. All of the shards are located in grid containers. Grid containers 1, 2, and 3 each hold three shards, while grid containers 4 and 5 hold two shards each.



ObjectGrid clients and servers

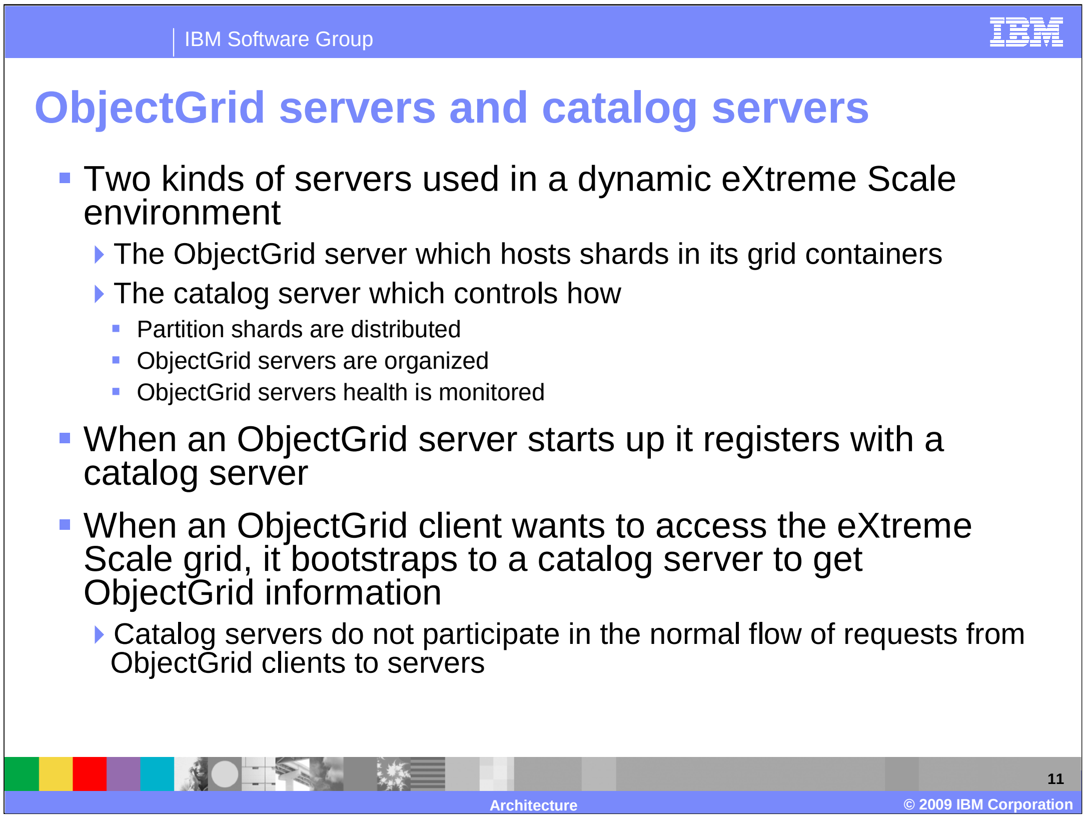
- An ObjectGrid server
  - Contains grid containers which host the cache (stored in BackingMaps)
  - Is a JVM running on a particular host and port that
    - Listens on a port for client requests
    - Can communicate with other ObjectGrid servers to create a self-configuring grid
- An ObjectGrid client
  - Can connect to an ObjectGrid server to perform various operations, such as create, retrieve, update and delete the objects in the grid
  - Contains an ObjectMap and may contain a “near-cache” copy of a BackingMap

There are two components that can contain BackingMaps within the eXtreme Scale framework, the ObjectGrid server and the ObjectGrid client. The ObjectGrid server, or simply grid server, contains grid containers which host the cache, stored in BackingMaps. The grid server is a JVM running on a particular host and port that listens for client requests and can communicate with other ObjectGrid servers and catalog servers to maintain a self-configuring, self-healing eXtreme Scale grid.

The ObjectGrid client, or grid client, will connect to a grid server to perform operations such as create, retrieve, update and delete to the objects in the eXtreme Scale grid within the scope of a single transaction. The client contains an ObjectMap and may contain a copy of a BackingMap.

The grid server BackingMap is always shared between clients, while the client-side BackingMap, or near-cache, is shared only between threads of the grid client.

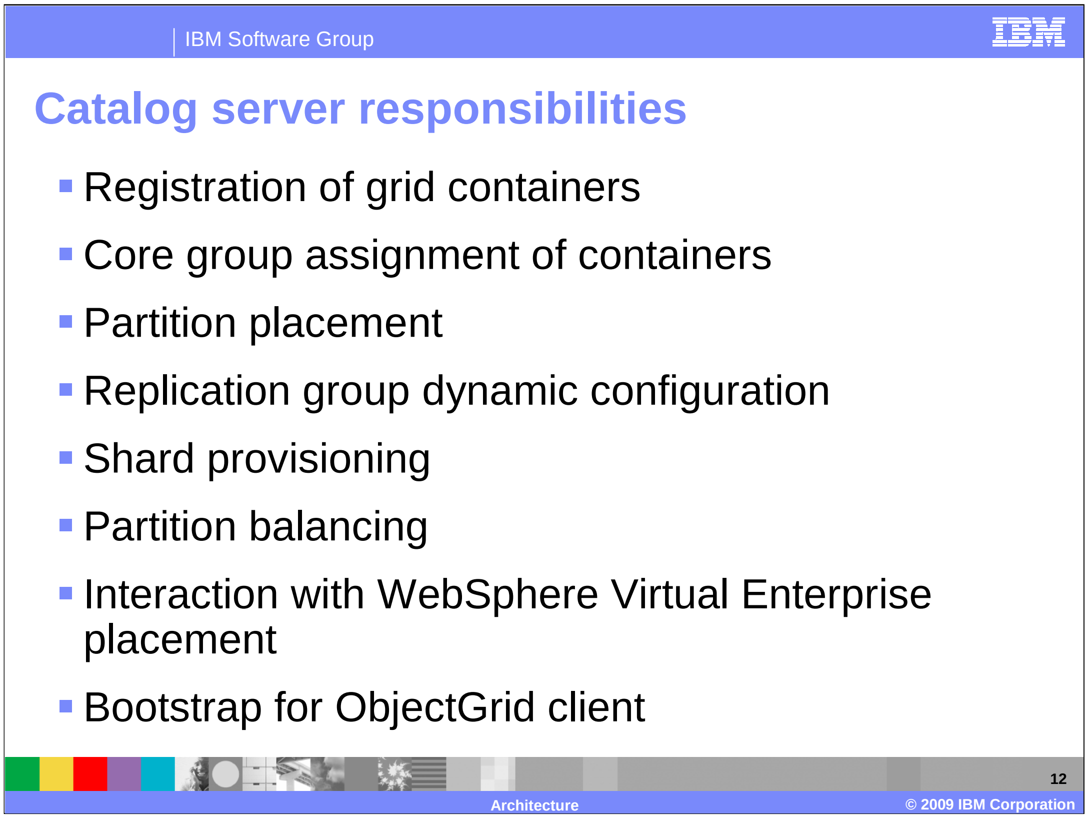
The terms “grid server” and “ObjectGrid server” can be used interchangeably.



ObjectGrid servers and catalog servers

- Two kinds of servers used in a dynamic eXtreme Scale environment
  - The ObjectGrid server which hosts shards in its grid containers
  - The catalog server which controls how
    - Partition shards are distributed
    - ObjectGrid servers are organized
    - ObjectGrid servers health is monitored
- When an ObjectGrid server starts up it registers with a catalog server
- When an ObjectGrid client wants to access the eXtreme Scale grid, it bootstraps to a catalog server to get ObjectGrid information
  - Catalog servers do not participate in the normal flow of requests from ObjectGrid clients to servers

The ObjectGrid servers that hold data in their grid containers are not the only type of eXtreme Scale servers. Catalog servers also play a role in the management of the eXtreme Scale grid. When an ObjectGrid server starts up it registers with a catalog server. The catalog server manages how the partition shards are distributed, monitors the health and status of the ObjectGrid servers, and enables the ObjectGrid clients to locate the primary shard for a requested object.



IBM Software Group

IBM

Catalog server responsibilities

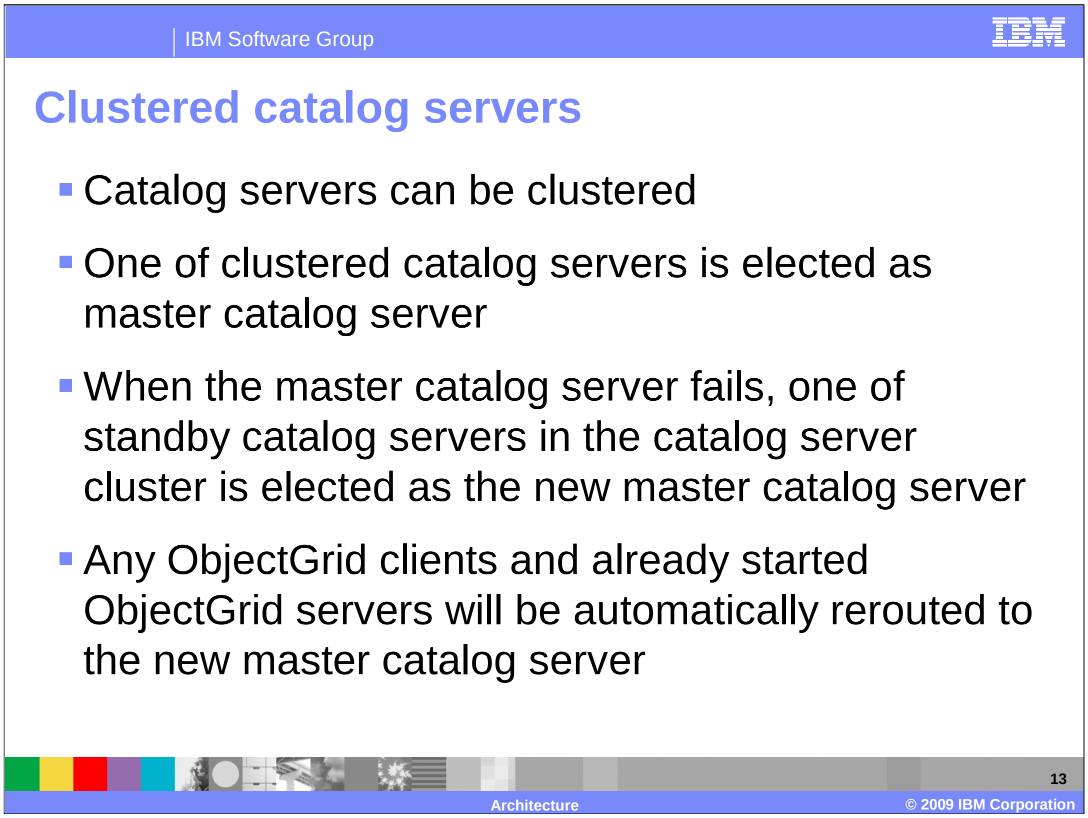
- Registration of grid containers
- Core group assignment of containers
- Partition placement
- Replication group dynamic configuration
- Shard provisioning
- Partition balancing
- Interaction with WebSphere Virtual Enterprise placement
- Bootstrap for ObjectGrid client

12

Architecture

© 2009 IBM Corporation

The catalog server is the engine that drives the grid operations. It provides services essential to maintaining a scalable self-healing grid.



IBM Software Group

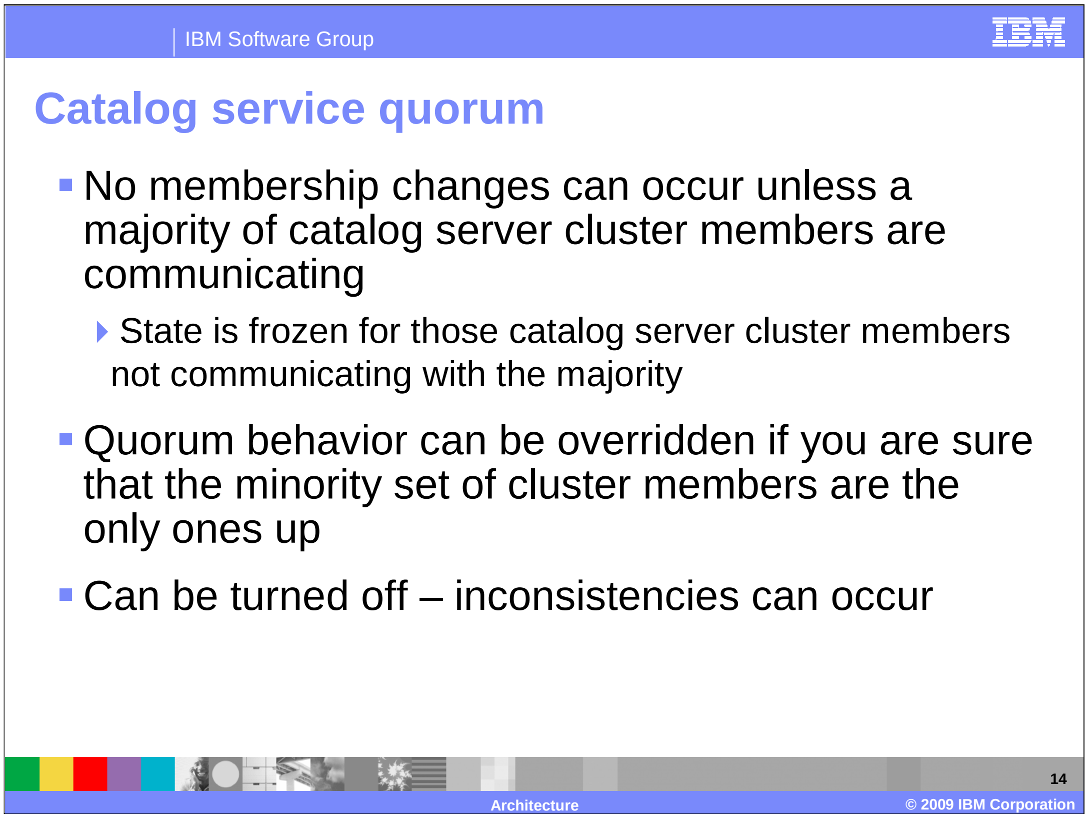
Clustered catalog servers

- Catalog servers can be clustered
- One of clustered catalog servers is elected as master catalog server
- When the master catalog server fails, one of standby catalog servers in the catalog server cluster is elected as the new master catalog server
- Any ObjectGrid clients and already started ObjectGrid servers will be automatically rerouted to the new master catalog server

13

Architecture

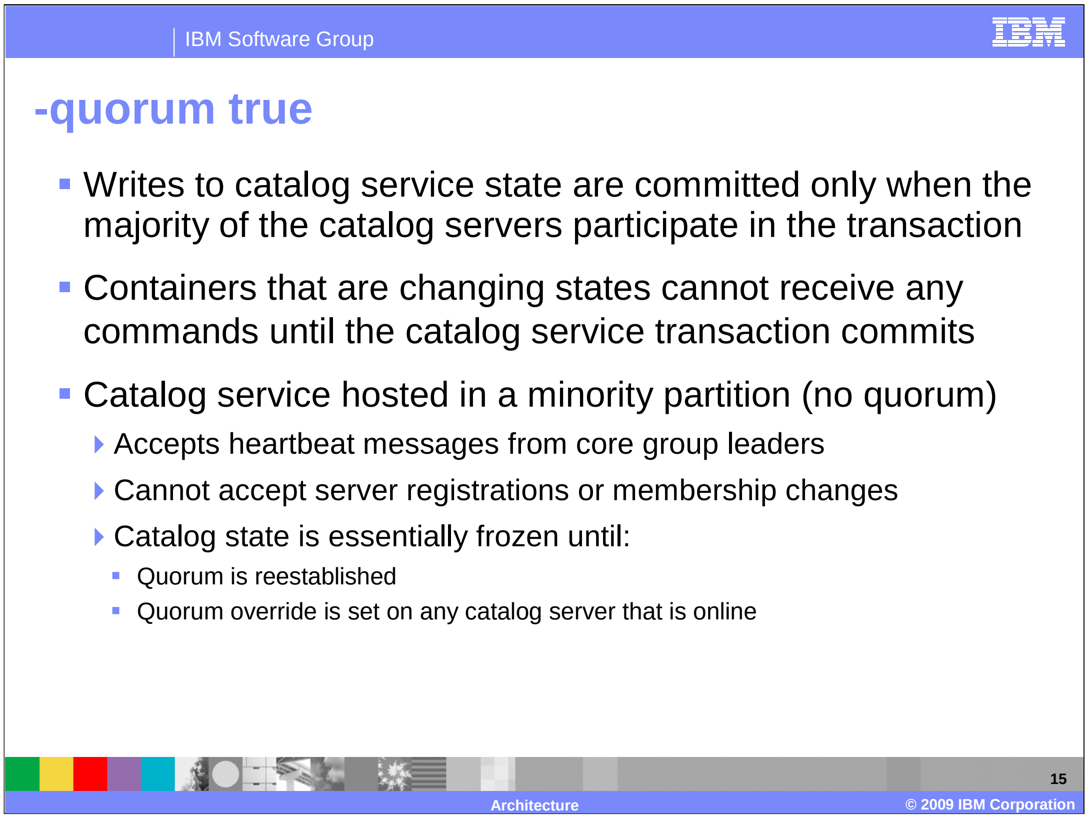
Catalog servers can be clustered to ensure high availability. Initially, one of the catalog servers is elected as the master catalog server. If the master fails, a standby catalog server is elected as the new master. ObjectGrid clients and servers are automatically rerouted to new master catalog server.



Catalog service quorum

- No membership changes can occur unless a majority of catalog server cluster members are communicating
  - State is frozen for those catalog server cluster members not communicating with the majority
- Quorum behavior can be overridden if you are sure that the minority set of cluster members are the only ones up
- Can be turned off – inconsistencies can occur

A catalog service quorum can be required when the catalog servers in your environment are clustered. Having more than one catalog server means that information about the ObjectGrid server members in your eXtreme Scale grid will be maintained and replicated across multiple catalog servers. The quorum specifies that a minimum number of active catalog servers are required for the eXtreme Scale grid to accept membership registrations and changes to grid membership to ensure proper routing and shard placement. So, when a new server is added to your eXtreme Scale grid for example, the grid membership update is only allowed if the majority of catalog servers are online to acknowledge the membership change. The catalog service will wait for any shard placement changes to occur as a result of the addition of the new JVM and for those changes to be replicated across the active catalog servers in the cluster before taking any further action.



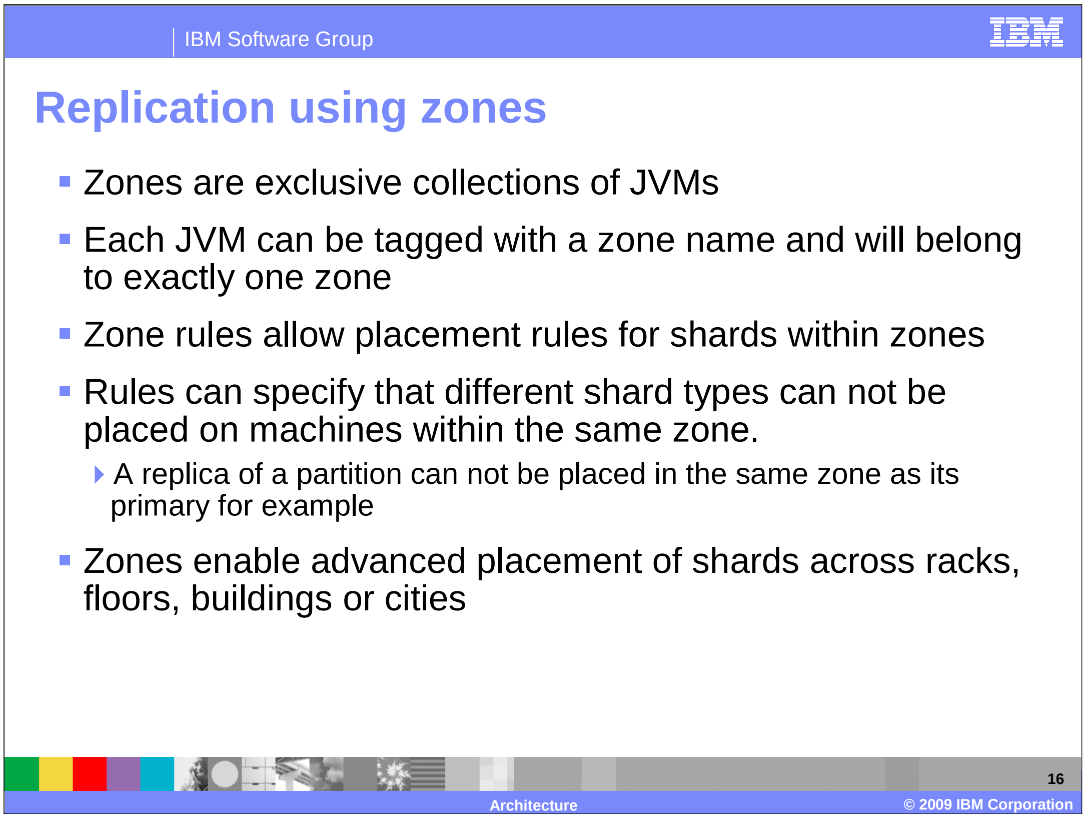
-quorum true

- Writes to catalog service state are committed only when the majority of the catalog servers participate in the transaction
- Containers that are changing states cannot receive any commands until the catalog service transaction commits
- Catalog service hosted in a minority partition (no quorum)
  - Accepts heartbeat messages from core group leaders
  - Cannot accept server registrations or membership changes
  - Catalog state is essentially frozen until:
    - Quorum is reestablished
    - Quorum override is set on any catalog server that is online

By default, quorum mode is enabled. It allows writes to the catalog service state to be committed only when the majority of the catalog servers participate in the transaction. Containers that are changing states cannot receive any commands, unless the catalog service transaction commits first.

If the catalog service is hosted in a minority partition, meaning no quorum has been establish, it will accept liveness messages. It cannot, however, accept server registrations or membership changes, because the state is essentially frozen until one of following two things occurs: quorum is reestablished or quorum override is set on any catalog server that is on-line.

This approach allows the catalog service to manage shard distribution and to maintain a consistent grid server membership list.

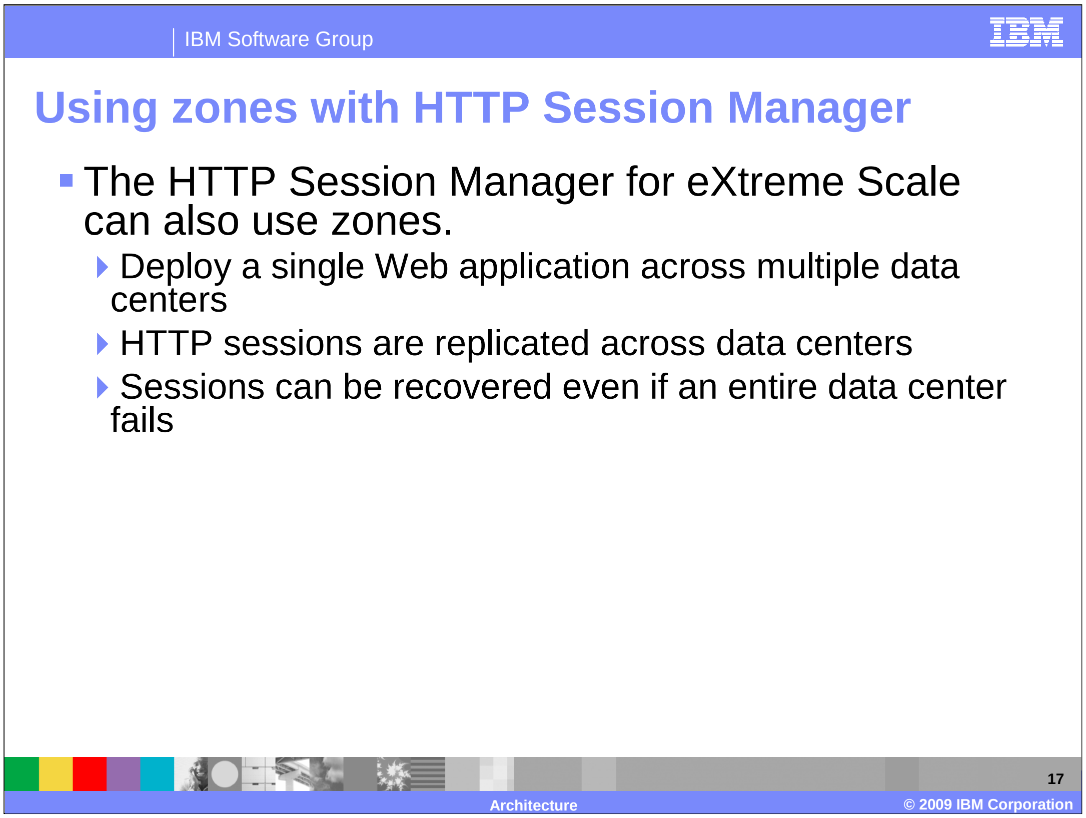


Replication using zones

- Zones are exclusive collections of JVMs
- Each JVM can be tagged with a zone name and will belong to exactly one zone
- Zone rules allow placement rules for shards within zones
- Rules can specify that different shard types can not be placed on machines within the same zone.
  - A replica of a partition can not be placed in the same zone as its primary for example
- Zones enable advanced placement of shards across racks, floors, buildings or cities

Zone capability is useful to ensure that primary and replica shards are placed in different physical locations for better fault tolerance. A zone is essentially an exclusive collection of JVMs. Rules are used to specify the possible set of zones that a shard can be placed in. The catalog server will follow the zone rules

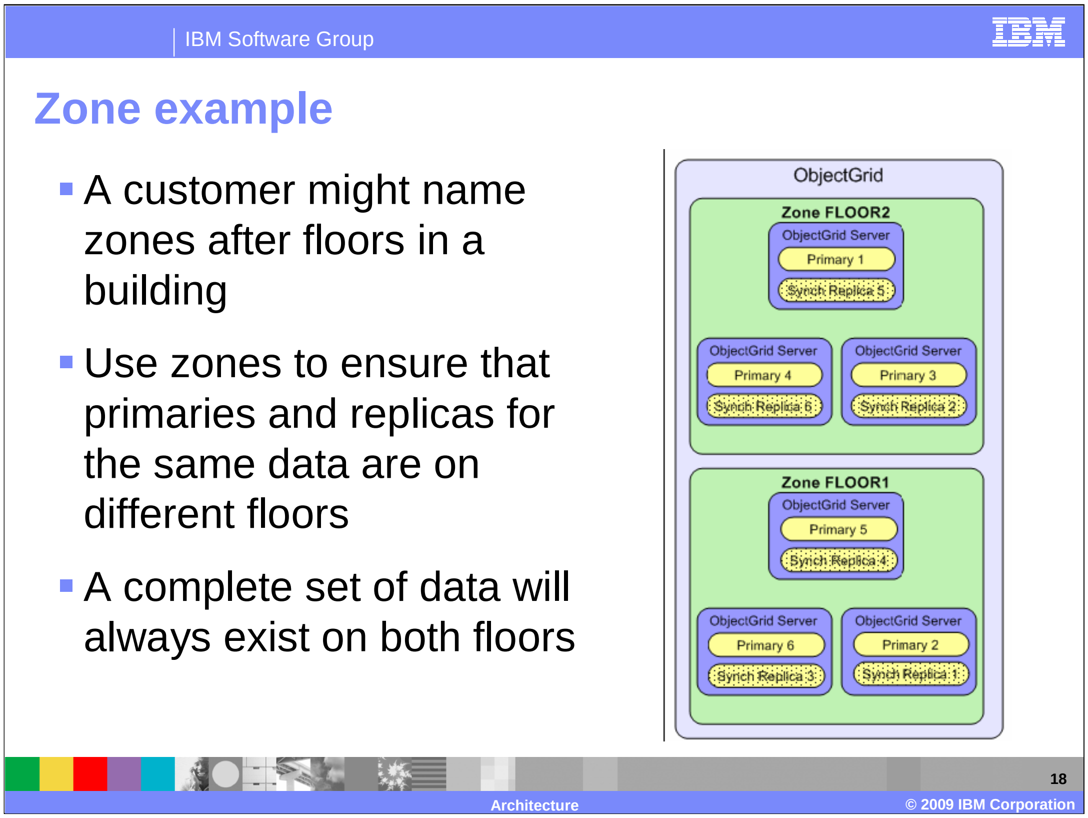
to place the primary and replica shards across these groups of JVMs, or zones. As a general practice, you should place only synchronous replicas in the same zone and asynchronous replicas in a different zone to ensure high availability for a JVM failure in a local zone (synchronous replica) and in the case of a complete data center failure (asynchronous replica). Zones can exist across racks, floors, buildings, or even cities.



Using zones with HTTP Session Manager

- The HTTP Session Manager for eXtreme Scale can also use zones.
  - Deploy a single Web application across multiple data centers
  - HTTP sessions are replicated across data centers
  - Sessions can be recovered even if an entire data center fails

Zone support is also provided for HTTP Session Manager, so HTTP sessions can be dispersed across different locations or zones for better fault tolerance.



IBM Software Group

Zone example

- A customer might name zones after floors in a building
- Use zones to ensure that primaries and replicas for the same data are on different floors
- A complete set of data will always exist on both floors

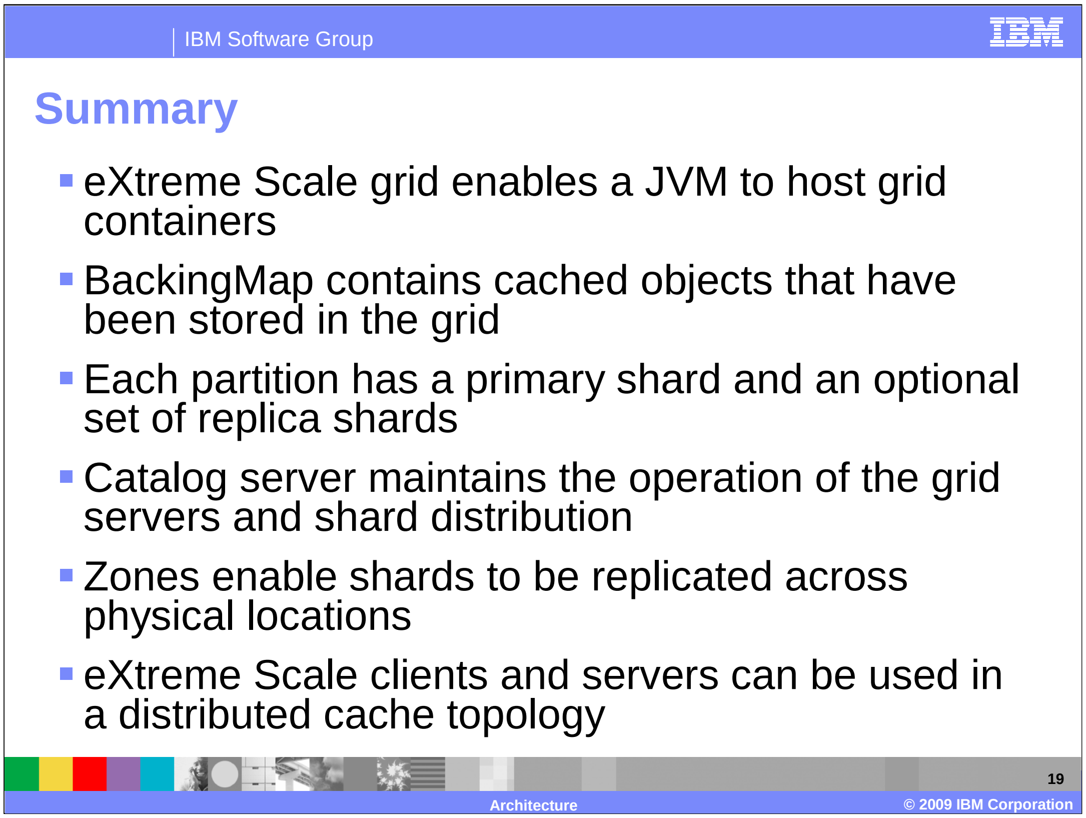
ObjectGrid
Zone FLOOR2
ObjectGrid Server
Primary 1
Synch Replica 5
ObjectGrid Server
Primary 4
Synch Replica 6
ObjectGrid Server
Primary 3
Synch Replica 2
Zone FLOOR1
ObjectGrid Server
Primary 5
Synch Replica 4
ObjectGrid Server
Primary 6
Synch Replica 3
ObjectGrid Server
Primary 2
Synch Replica 1

18

Architecture

© 2009 IBM Corporation

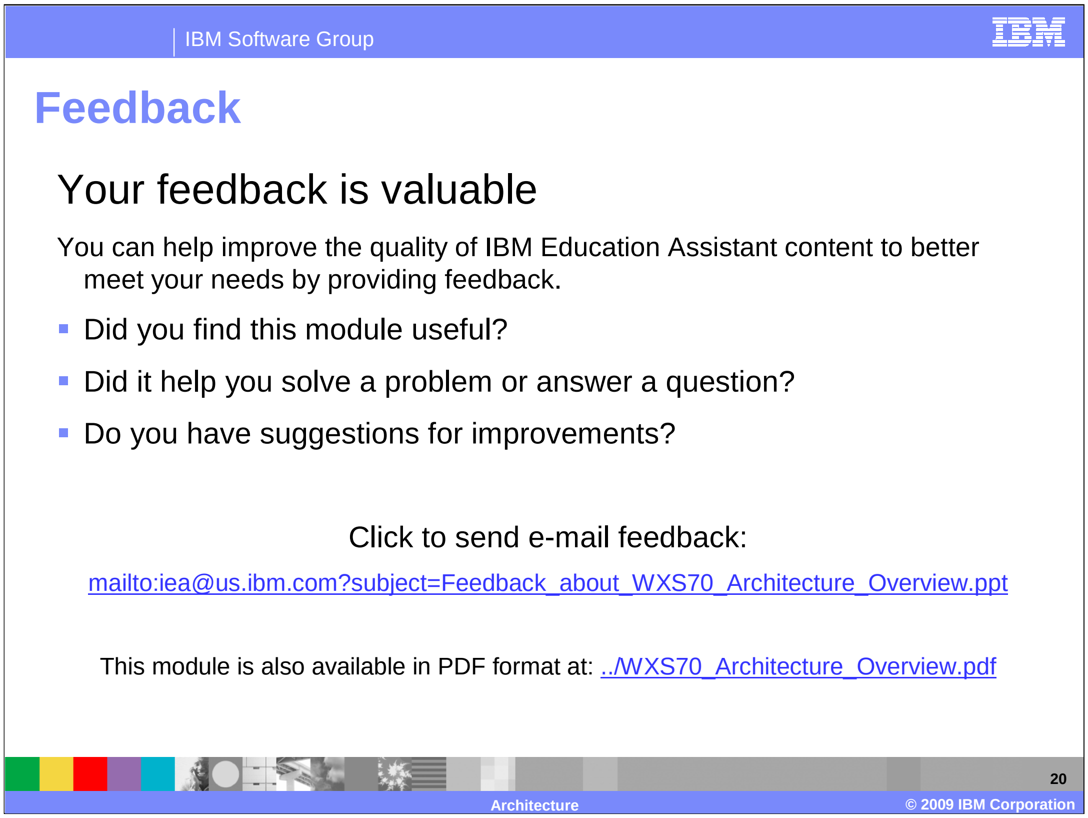
Here is an example of how to use zones. A customer using zones across two physical locations, such as floors in a building for example, would have the primaries distributed across both zones. The replica for each primary would be placed on a different floor than the primary. In this case, the floor name is used as the zone name. Zones ensure the data is adequately replicated between building floors and the data is available to the application regardless of its zoned location.



Summary

- eXtreme Scale grid enables a JVM to host grid containers
- BackingMap contains cached objects that have been stored in the grid
- Each partition has a primary shard and an optional set of replica shards
- Catalog server maintains the operation of the grid servers and shard distribution
- Zones enable shards to be replicated across physical locations
- eXtreme Scale clients and servers can be used in a distributed cache topology

This presentation covered the eXtreme Scale terminology to show how an eXtreme Scale grid is structured. WebSphere eXtreme Scale enables a JVM to host one or more grid containers that can be used as a cache for storing Java objects or application state. Cached objects are stored in a BackingMap. Partitions are used to split data into smaller groups and consist of a primary shard and zero or more replica shards. A catalog server plays an important role in the environment as the manager of shard distribution and the monitor of the availability and health of the grid servers. Zones can be used to ensure primary and replica shards are placed in separate physical locations. Finally, in a distributed cache topology, eXtreme Scale clients can be used to separate the application logic from the grid servers to reduce the load on your back-end data store.



Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

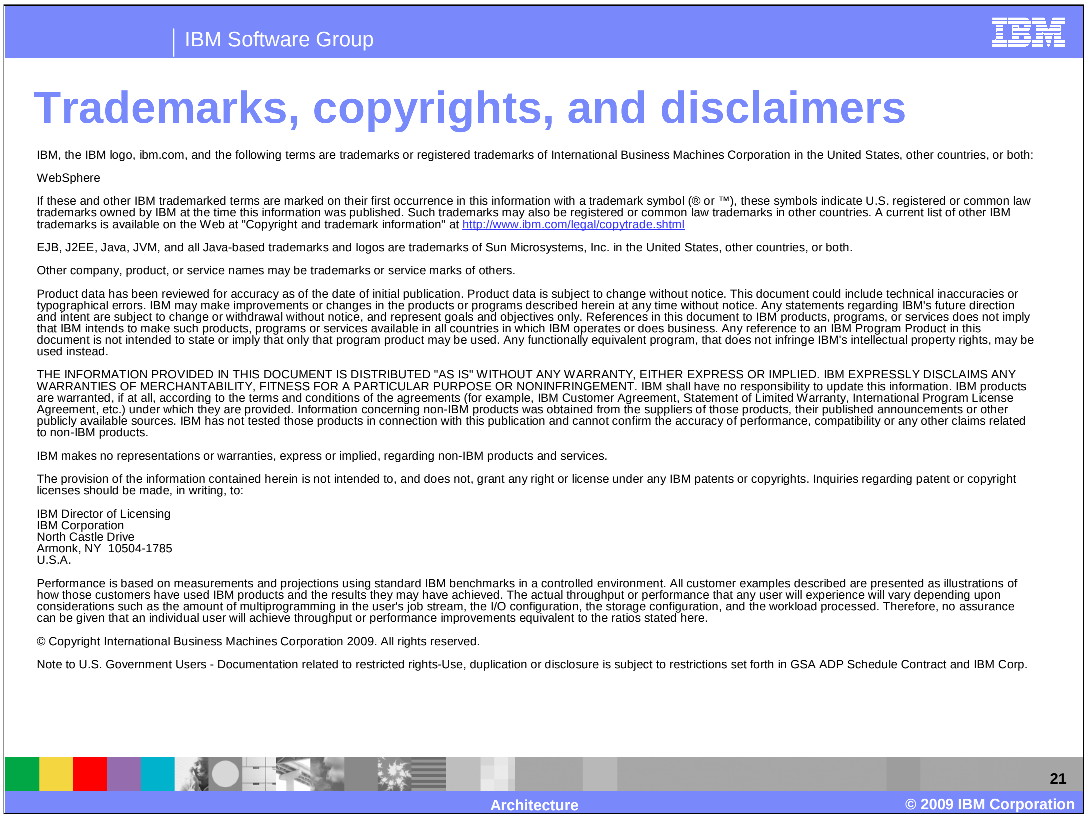
- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WXS70_Architecture_Overview.ppt

This module is also available in PDF format at: ../WXS70_Architecture_Overview.pdf

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



Trademarks, copyrights, and disclaimers