Big Data Sharing with the Cloud - WebSphere eXtreme Scale and IBM Integration Bus Integration

Dave Gorman – IBM Integration Bus Performance Team Lead

IBM Hursley – gormand@uk.ibm.com

Thursday 15th August 2013

13873

SHARE
• in Boston

Agenda

- Scenarios

- WebSphere eXtreme Scale concepts

- Programming model

- Operational model

Agenda

- Scenarios
- WebSphere eXtreme Scale concepts
- Programming model
- Operational model

Scenario 1 - Storing state for integrations

- When IIB is used to integrate 2 asynchronous systems, the broker needs to record some state about the requester in order to correlate the replies correctly.
- How can this scale across multiple brokers?
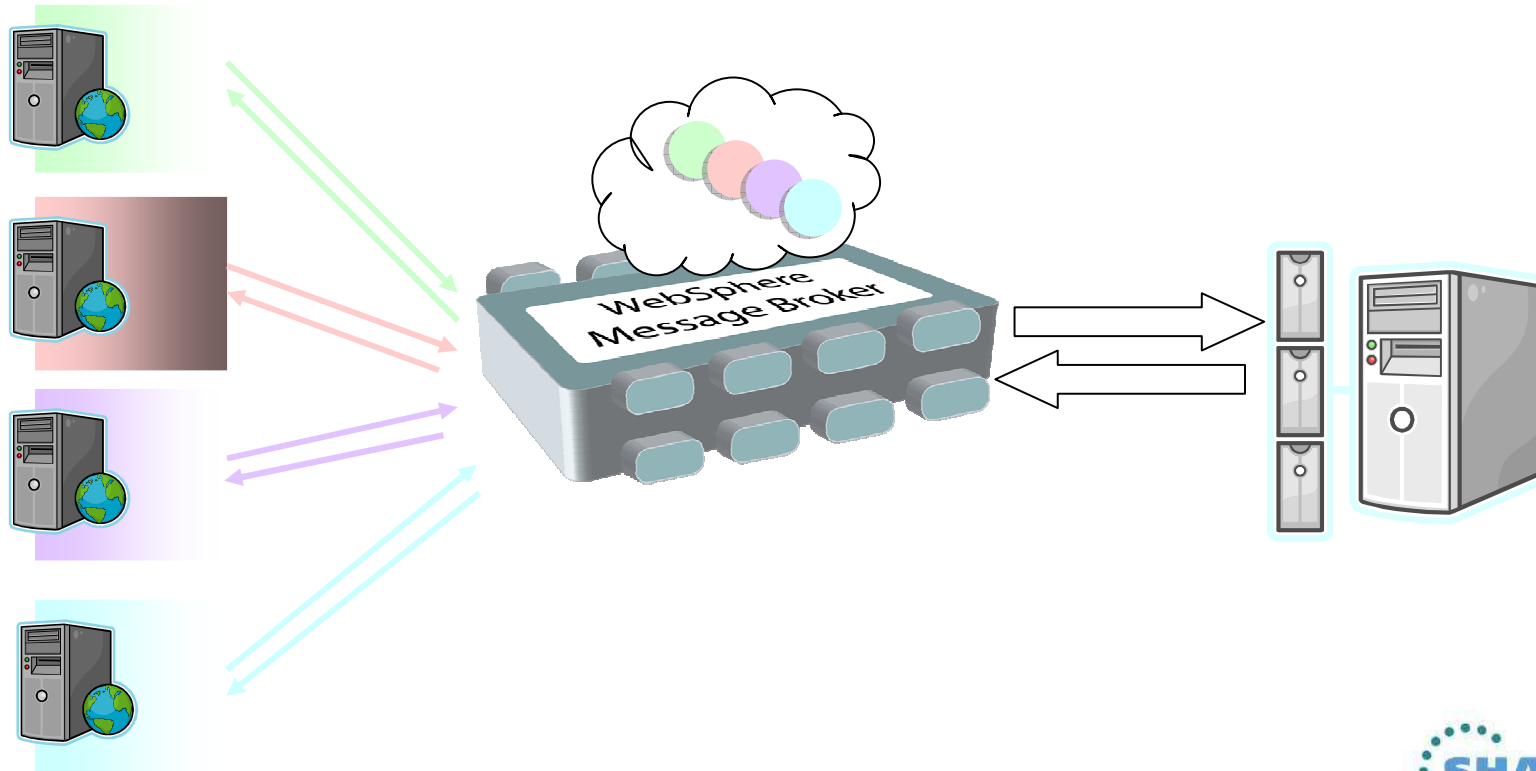
Scenario 1 - Storing state for integrations

- With a global cache, each broker can handle replies – even when the request was processed by another broker.
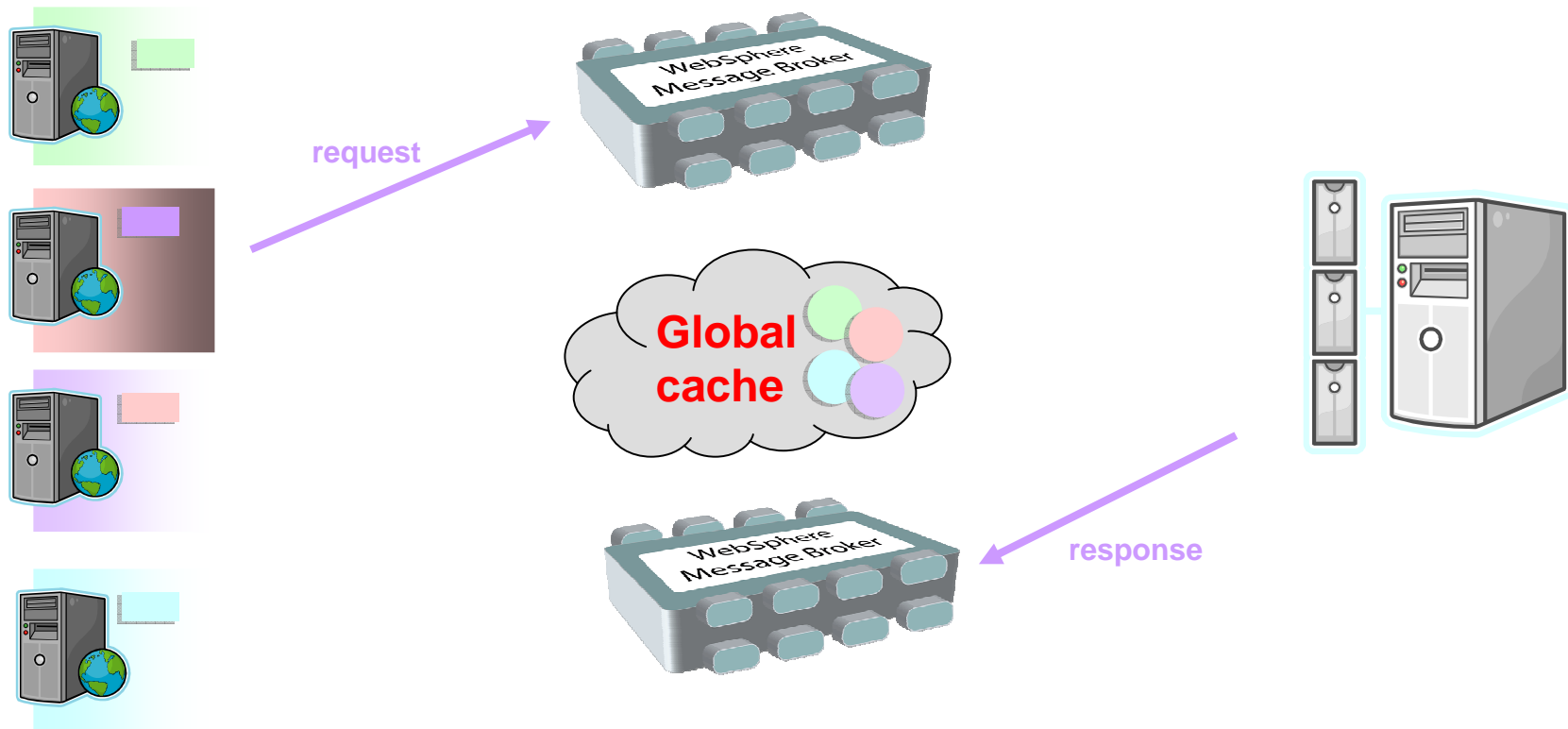
Complete your sessions evaluation online at SHARE.org/BostonEval

Scenario 2 - Caching static data

- When acting as a façade to a back-end data base, IIB needs to provide short response time to client, even though the backend has high latency.
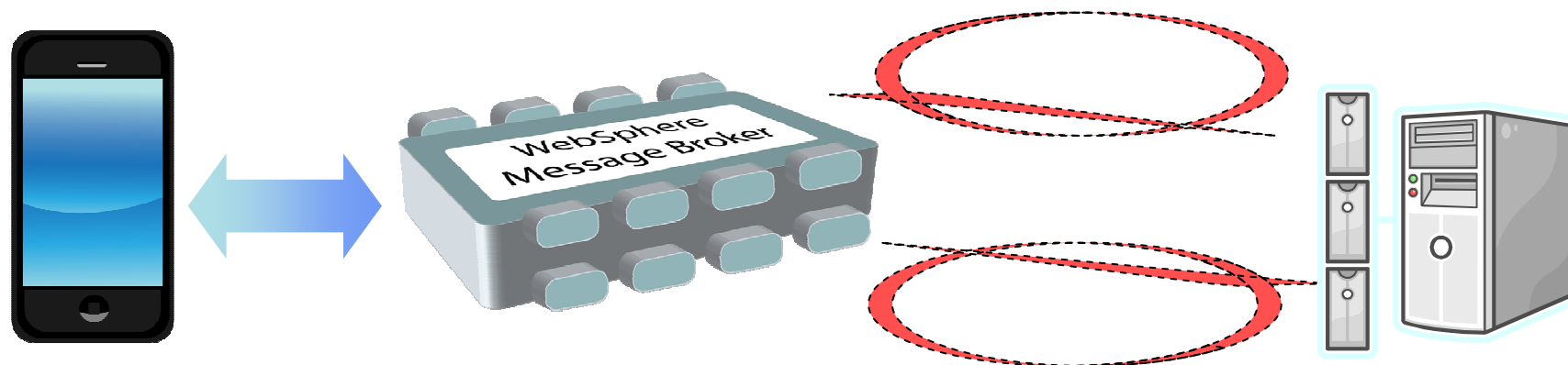
Scenario 2 - Caching static data
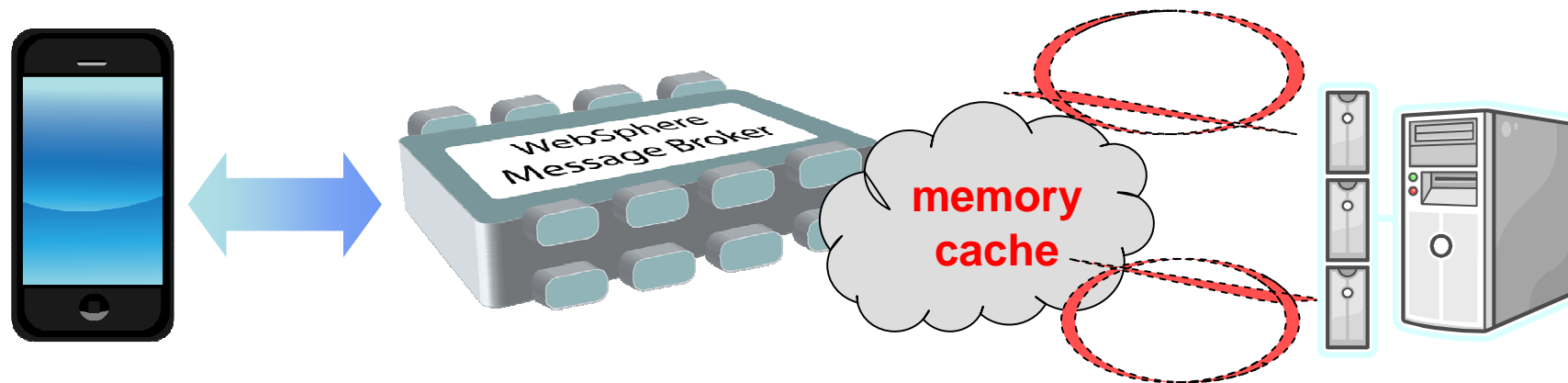
- When acting as a façade to a back-end data base, IIB needs to provide short response time to client, even though the backend has high latency.
- This can be solved by caching results in memory (e.g. ESQL shared variables).
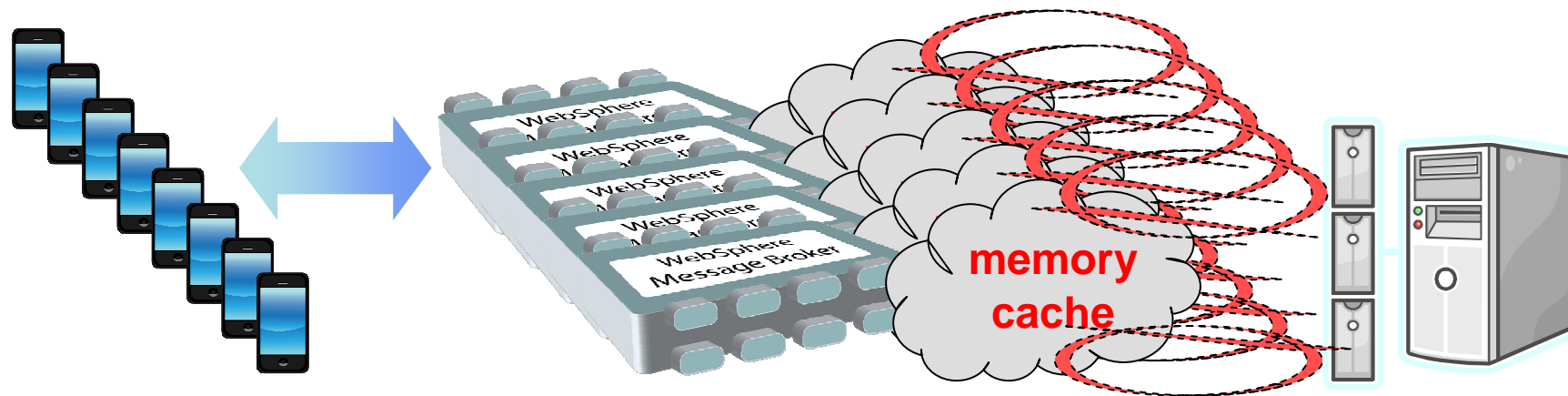
Scenario 2 - Caching static data

- When acting as a façade to a back-end data base, IIB needs to provide short response time to client, even though the backend has high latency.
- This can be solved by caching results in memory (e.g. ESQL shared variables).
- However, this does not scale well horizontally.
- When the number of clients increase, the number of brokers/execution groups can be increased to accommodate – but each one has to maintain a separate in-memory cache

memory cache

Scenario 2 - Caching static data

- With a global cache, the number of clients can increase while maintaining a predictable response time for each client.
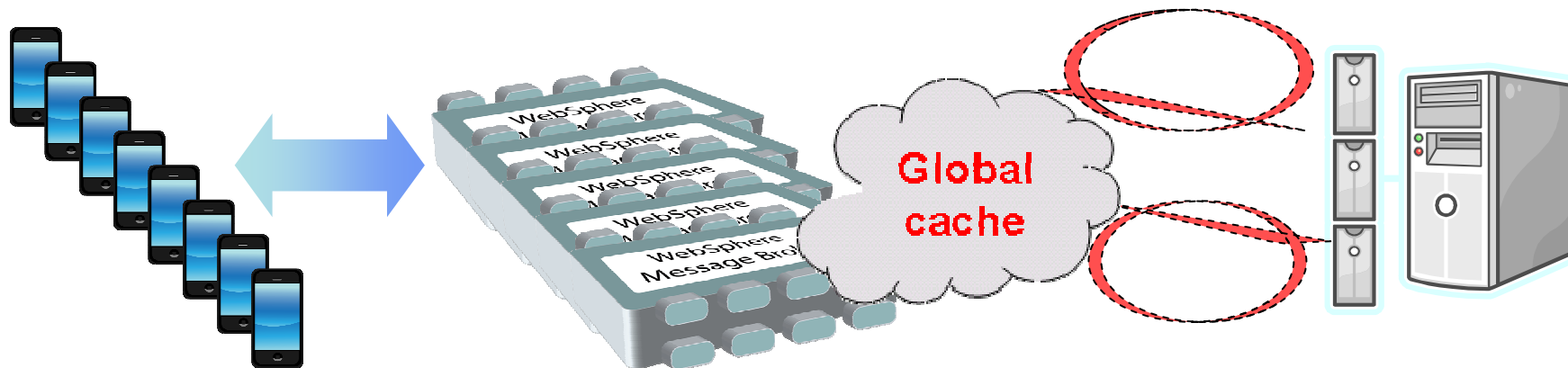
Agenda

- Scenarios
- WebSphere eXtreme Scale concepts
- Programming model
- Operational model

Complete your sessions evaluation online at SHARE.org/BostonEval

WXS Concepts - Overview

- Elastic In-Memory Data Grid

- Virtualizes free memory within a grid of Java Virtual Machines (JVMs) into a single logical space which is accessible as a partitioned, key addressable space for use by applications

- Provides fault tolerance through replication with self monitoring and healing

- The space can be scaled out by adding more JVMs while it's running without restarting

- Elastic means it manages itself! Auto scale out, scale in, failover, failure compensation, everything!

Why an in-memory cache?

- Speed
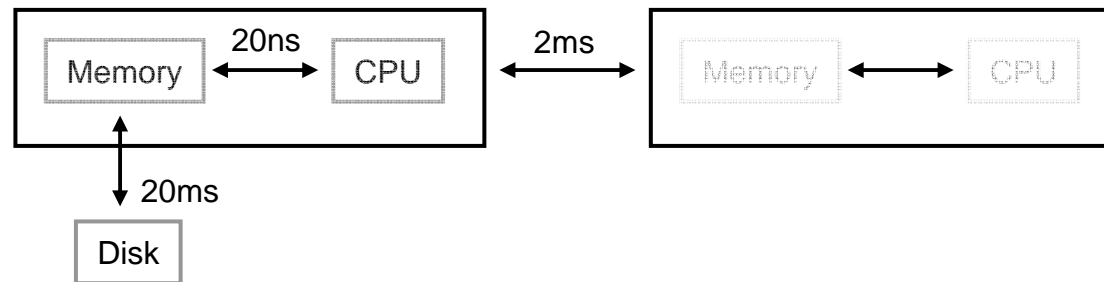
- Challenges
  - Physical memory size
  - Persistence
  - Resilience

Complete your sessions evaluation online at SHARE.org/BostonEval

Notes

- If we look at the speed differences of the various interfaces on a computer we can see that looking up data in memory is much quicker than looking it up from disk. So in a high performance system you want to try and retain as much data in memory as possible. However at some point you run out of available real memory on a system and then you have to start paging for virtual storage or you have to offload to disk, but that of course will incur the cost of disk access which won't perform.

- However a well setup network will allow you to read data from memory on another machine quicker than you can look it up from the local disk. So why don't all systems do this?

- Persistence, we often want data to survive outages, but this requires writing to a disk so that the data can be reloaded in to memory after an outage.

Scaling

- Vertical or horizontal?
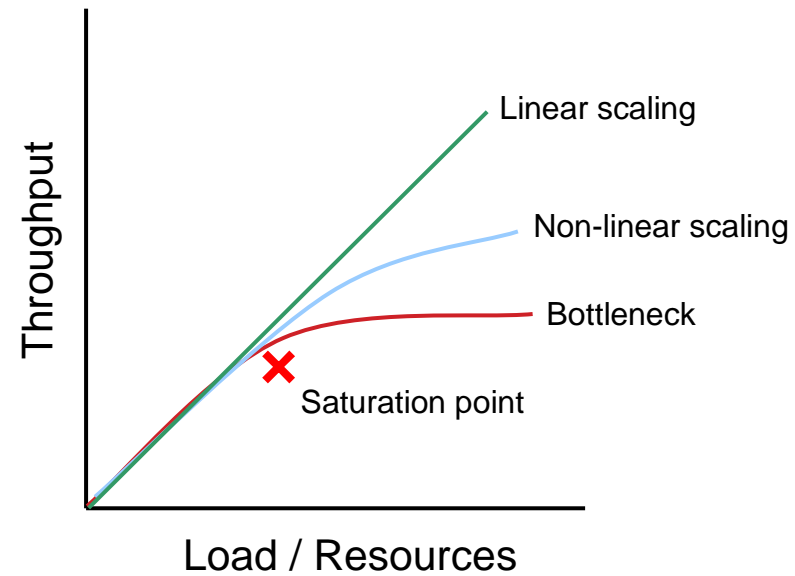
Complete your sessions evaluation online at SHARE.org/BostonEval

Notes

- We're next going to look at the different approaches to performance and scaling. In an ideal world you would have no bottlenecks in a system and as you add more resources you can get a higher throughput in a linear fashion. In reality in most instances as you add more resources you get less and less benefit and eventually you may also reach a point where throughput stops increasing as you reach a bottleneck after the saturation point. At this point you need to eliminate the bottle neck to get the throughput to increase again.

- In most cases the obvious approach to increase throughput is to throw more processors or memory at a system to get the throughput to scale. This is vertical scaling. Eventually, perhaps with a traditional database system, you will hit a bottleneck, perhaps with IO to the disk, that stops the scaling completely. You could try to buy more faster and expensive hardware to overcome this bottleneck, but you may need a different approach. The next step is to try distributing the work over more machines and at this point you get to horizontal scaling.

- With a traditional database system this may not be possible because you may not be able to scale a database across multiple systems without still having a disk IO bottleneck.

- At this point you may want to think about partitioning the data so that half is stored in one place and half in another, thus allowing horizontal scale of the lookup.
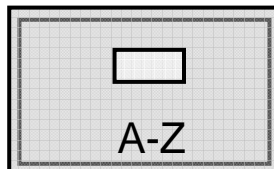
WXS Concepts

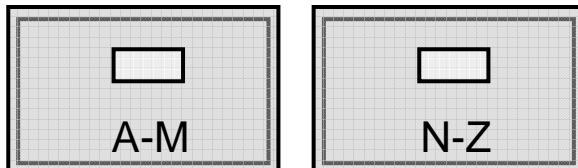- Partition
- Container
- Catalog
- Shard
- Zone

Complete your sessions evaluation online at SHARE.org/BostonEval
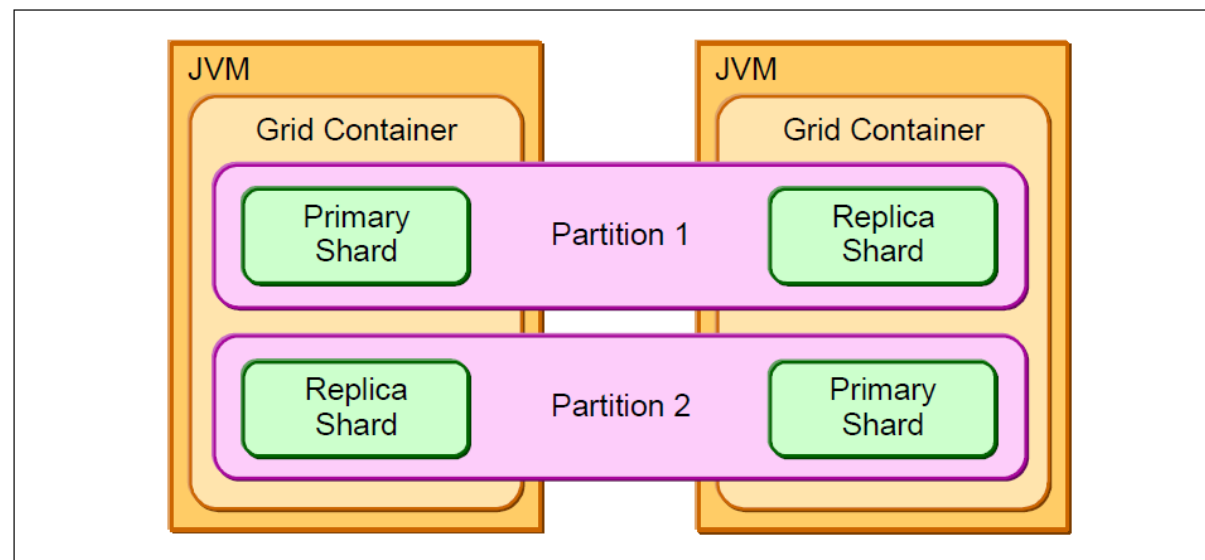
WXS Concepts - Partitions
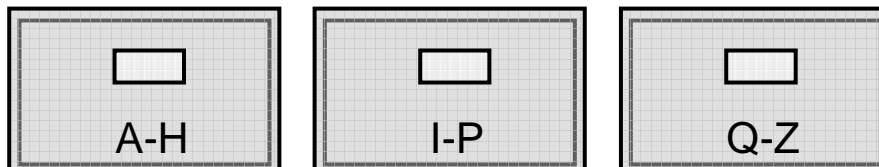
- Partitions
  - Think of a filing cabinet
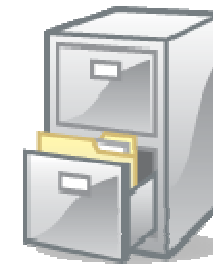 - To start off with you store everything in one draw

    A-Z

  - This fills up, so split across 2 draws

    A-M   N-Z

  - Then that fills up so you split across 3, etc
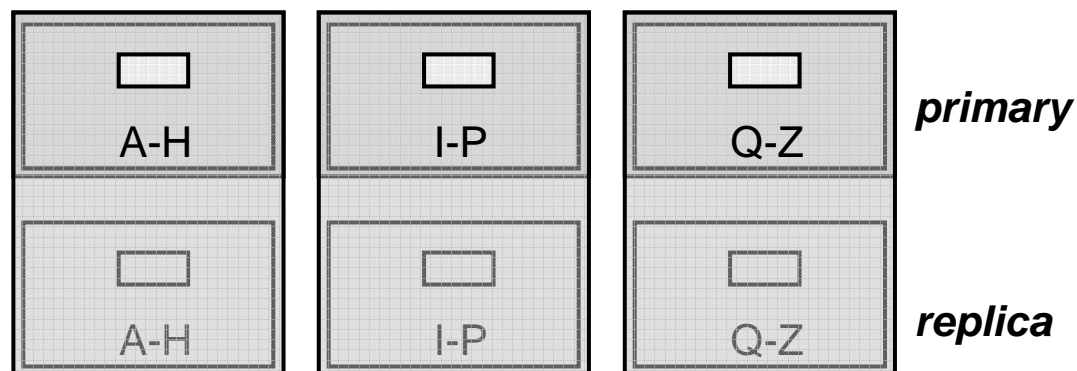
    A-H   I-P   Q-Z

  - We're partitioning the data to spread it across multiple 'servers'
  - WXS groups your data in partitions

WXS Concepts - Shards

- Shard
  - Think of a copy of you data in another filing cabinet draw
  - This provides a backup of your data


primary

replica

- In WXS terms a shard is what actually stores your data
  - A shard is either a *primary* or a *replica*
- Each partition contains at least 1 shard depending on configuration
  - A partition is a group of shards

WXS Concepts – Container / Zone

- Container
  - Think of the room(s) you store your filing cabinet(s) in
  - They could all be in 1 room
  - Or split across several rooms in your house
  - Or perhaps some in your neighbours house
  - Depending on your access and resilience goal each option has a benefit

  - In WXS terms a container stores shards
    - A partition is split across containers
      - *(Assuming more than 1 shard per partition)*
  - Container position defines resilience

- Zone
  - A Zone is the name of your room
    - Or your house, or your neighbours house, or the town, etc
  - WXS uses zones to define how shards are placed
    - eg primary & replicas in different zones
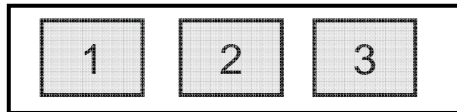 - You can have multiple containers in a zone

A Very Simple WXS Topology
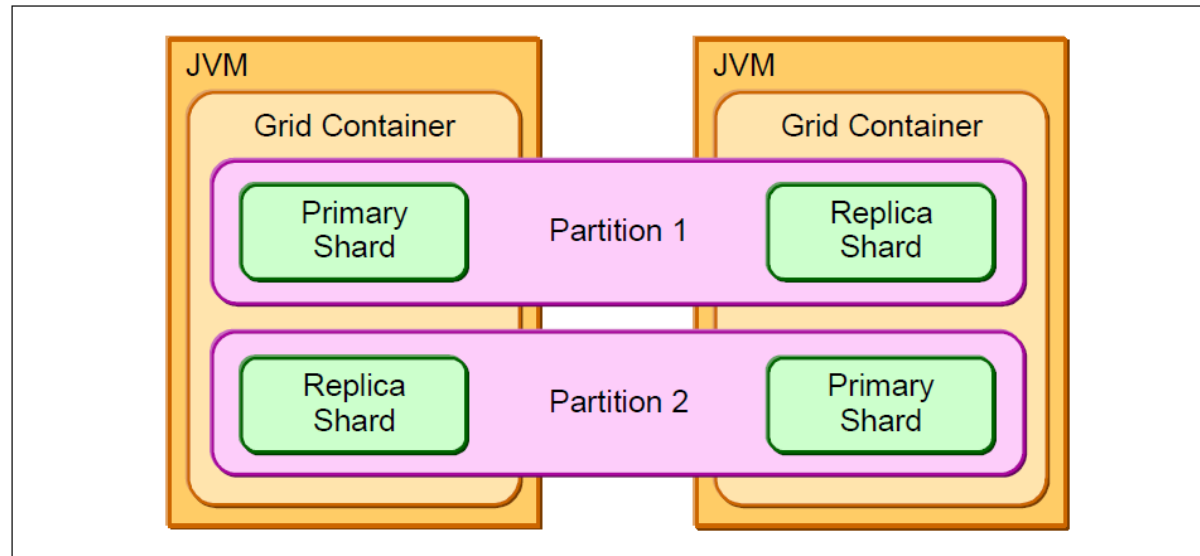
- You can replace JVM with
  - WebSphere Application Server
  - Datapower XC10
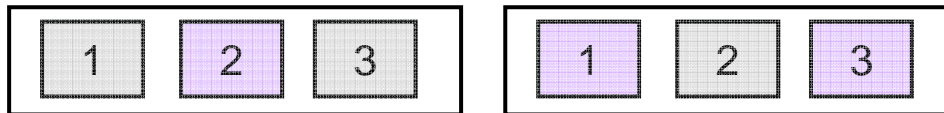 - IBM Integration Bus Execution Group

Elastic In-Memory Data Grid

- Dynamically reconfigures as JVMs come and go
- 1 JVM

1	2	3

  - No replicas
- 2 JVMs

1	2	3

1	2	3

  - Replicas are created and distributed automatically
- 3 JVMs

1	2

2	3

1	3
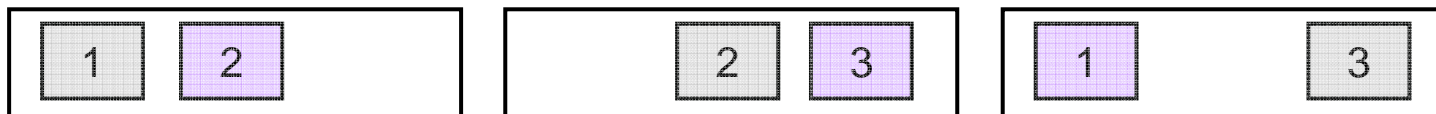
  - In the event of a failure the data is still available and is then rebalanced:

1	2	3

1	2	3

WXS Concepts - Catalog

Catalog

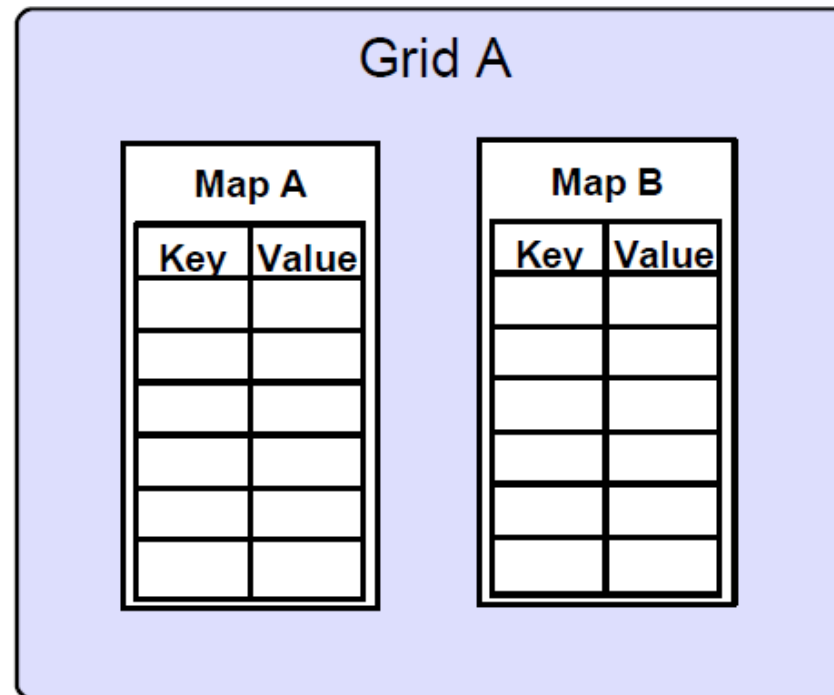|	Pri	Rep
A-H	A	C
I-P	B	A
Q-Z	C	B

- Catalog
  - A catalog is like an index
    - It tells you where to find stuff from a lookup

- In WXS the catalog server keeps a record of where the shards for each partition are located
  - The catalog is key, if you lose it, you lose the cache

JVM A

| 1 | 2 |

JVM B

| 2 | 3 |

JVM C

| 1 | 3 |
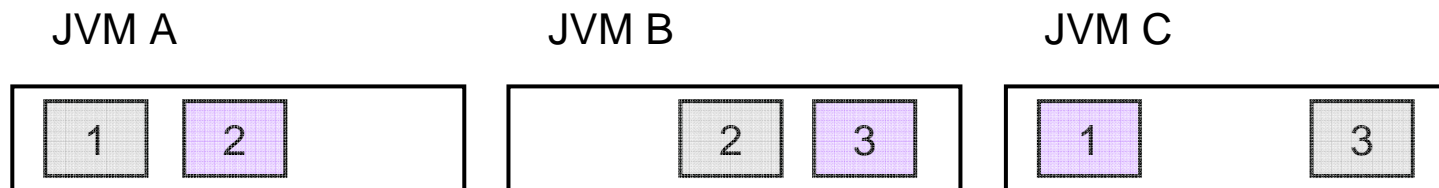
User/Developer view

- A WXS developer sees only different maps of key value pairs

IIB Global Cache implementation

- IIB contains an embedded WXS grid
  - For use as a global cache from within message flows.
  - WXS components are hosted within execution group processes.

- It works out of the box, with default settings, with no configuration. You just have to switch it on.

- The default scope of one cache is across one Broker (i.e. multiple execution groups) but it is easy to extend this across multiple brokers.

- Advanced configuration available via execution group properties.

- The message flow developer has new artefacts for working with the global cache, and is not aware of the underlying technology (WXS).

- Access and store data in external WXS grids
  - Datapower XC10 / WebSphere Application Server

IIB Global Cache implementation

- Our implementation has
  - 13 partitions
  - one synchronous replica for each primary shard
  - changes are sync'd from primary to replica immediately.
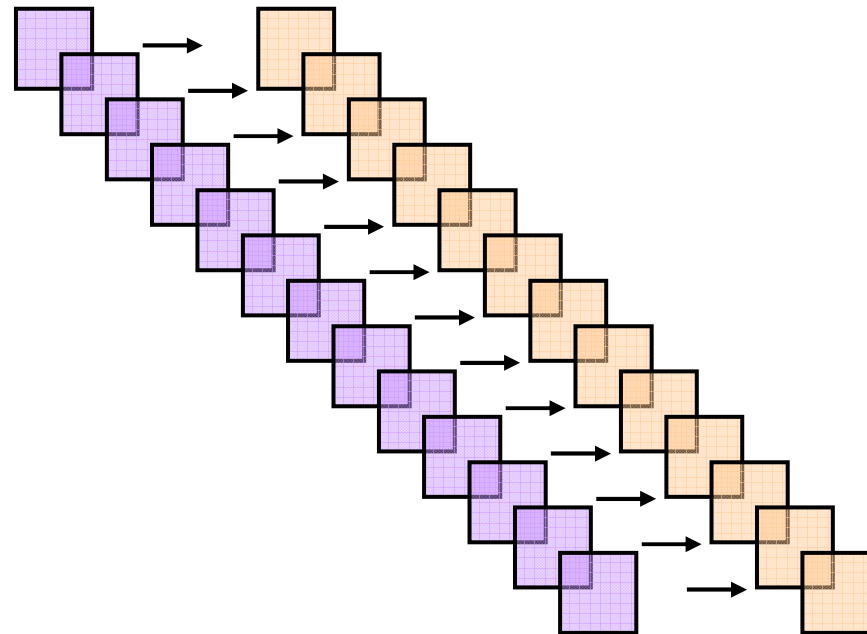
Agenda

- Scenarios
- WebSphere eXtreme Scale concepts
- Programming model
- Operational model

Complete your sessions evaluation online at SHARE.org/BostonEval

IIB Programming model

- Java Compute Node framework has new MbGlobalMap class for accessing the cache

- Underlying implementation (WXS) not immediately apparent to the flow developer

- Each cache action is autocommitted, and is visible immediately

- Other WMB language extensions to follow
  - Possible to call Java from most other interfaces so can still access cache now
    - eg ESQL calling Java

IIB Programming model - Java

JavaCompute

```
public class jcn extends MbJavaComputeNode {
  public void evaluate(MbMessageAssembly assembly) throws MbException {
    ...
    MbGlobalMap myMap = MbGlobalMap.getGlobalMap("myMap");
    ...
    myMap.put(varKey, myValue);
    myMap.put("aKey", myValue);
    ...
    myValue = (String)myMap.get(varKey);

  }
}
```

New MbGlobalMap object. With static 'getter' acting as a factory.

Can also getMap() to get the default map.

Getter handles client connectivity to the grid.

Data is PUT on the grid

Data is read from the grid

IIB Programming model - Java

- Each statement is autcommitted

- Consider doing your cache actions as the last thing in the flow (after the MQOutput, for instance) to ensure you do not end up with data persisted after flow rollback

- put() will fail if the value already exists in the map. use containsKey() to check for existence and update() to change the value

- Clear out data when you are finished! (Using remove())

- clear() is not supported – but can be achieved administratively using mqsicacheadmin, described later

- All mapnames are allowed apart from SYSTEM.BROKER.*

- Consider implementing own naming convention, similar to a pubsub topic hierarchy

Map Eviction

- Can specify a session policy when retrieving a map

  MbGlobalMap.*getGlobalMap*(String, MbGlobalMapSessionPolicy)

- MbGlobalMap session policy takes one parameter in the constructor - time to live (in seconds!)

```
MbGlobalMap.getGlobalMap("myMap", new MbGlobalMapSessionPolicy(20));
```

- Time to live specifies how long data stays in the map before it's evicted

- This time applies to data put to or updated in the cache in that session

Map Eviction - Notes

- Multiple calls to *getGlobalMap*() with any signature will return an MBGlobalMap representing the same session with the map in question

- "getting" a new MbGlobalMap object will update the underlying session

- Consider the scenarios:

```
MbGlobalMap map1 = MbGlobalMap.getGlobalMap("myMap", new MbGlobalMapSessionPolicy(20));
MbGlobalMap map2 = MbGlobalMap.getGlobalMap("myMap", new MbGlobalMapSessionPolicy(40));
MbGlobalMap map3 = MbGlobalMap.getGlobalMap("myMap", new MbGlobalMapSessionPolicy(60));

map1.put("key1", "value1");
map2.put("key2", "value2");
map3.put("key3", "value3");
```

All three pieces of data will live for 60 seconds, as the session is updated at each call to *getGlobalMap*

```
MbGlobalMap map1 = MbGlobalMap.getGlobalMap("myMap", new MbGlobalMapSessionPolicy(20));
map1.put("key1", "value1");

MbGlobalMap map2 = MbGlobalMap.getGlobalMap("myMap", new MbGlobalMapSessionPolicy(40));
map2.put("key2", "value2");

MbGlobalMap map3 = MbGlobalMap.getGlobalMap("myMap", new MbGlobalMapSessionPolicy(60));
map3.put("key3", "value3");
```

Session only updated after each put, so each piece of data is put with the value that the MbGlobalMap object was retrieved with

Storing Java Objects

- The cache plugin interface looks the same:

  - MbGlobalMap.put(Object key, Object value)

- Support extends to Java Serializable and Externalizable objects

```
public class MyPojo implements Serializable
```

```
MbGlobalMap mbgm = MbGlobalMap.getGlobalMap();
mbgm.put("key", new MyPojo(5, 10));
```

```
MbGlobalMap mbgm = MbGlobalMap.getGlobalMap();
MyPojo mypojo = (MyPojo) mbgm.get("key");
```

- .jar files should be placed in shared classes

  - System level   -   /shared-classes
  - Broker level   -   /config/<node_name>/shared-classes
  - Execution group level   -   /config/<node_name>/<server_name>/shared-classes

Accessing external grids

- External grid can be
  - A DataPower® XC10 device
  - Embedded WebSphere eXtreme Scale servers in WebSphere Application Server network deployment installations
  - A stand-alone WebSphere eXtreme Scale software grid

- Create a WXSServer configurable service
  - Need to know the following about the external grid
    - The name of the grid
    - The host name and port of each catalog server for the grid
    - (Optional) The object grid file that can be used to override WebSphere eXtreme Scale properties
  - Set grid login properties using mqsisetdbparms
  - Can enable SSL on the grid connections

- When retrieving the grid in a Java Compute node, specify the configurable service name

  ```
  MbGlobalMap.getGlobalMap(GridName, ConfigurableServiceName)
  ```

  - Cannot specify eviction time for external grid as this is configured on the grid by the grid administrator

Complete your sessions evaluation online at SHARE.org/BostonEval

Agenda

- Scenarios

- WebSphere eXtreme Scale concepts

- Programming model

- Operational model

Complete your sessions evaluation online at SHARE.org/BostonEval

Topologies - introduction

- Broker-wide topology specified using a cache policy property.
  - The value Default provides a single-broker embedded cache, with one catalog server and up to four container servers.

- The initial value is Disabled - no cache components run in any execution groups. Switch to Default and restart in order to enable cache function.

- Choose a convenient port range for use by cache components in a given broker.

- Specify the listener host name, which informs the cache components in the broker which host name to bind to.

Diagram of the default topology

- Demonstrates the cache components hosted in a 6-EG broker, using the default policy.

Complete your sessions evaluation online at SHARE.org/BostonEval
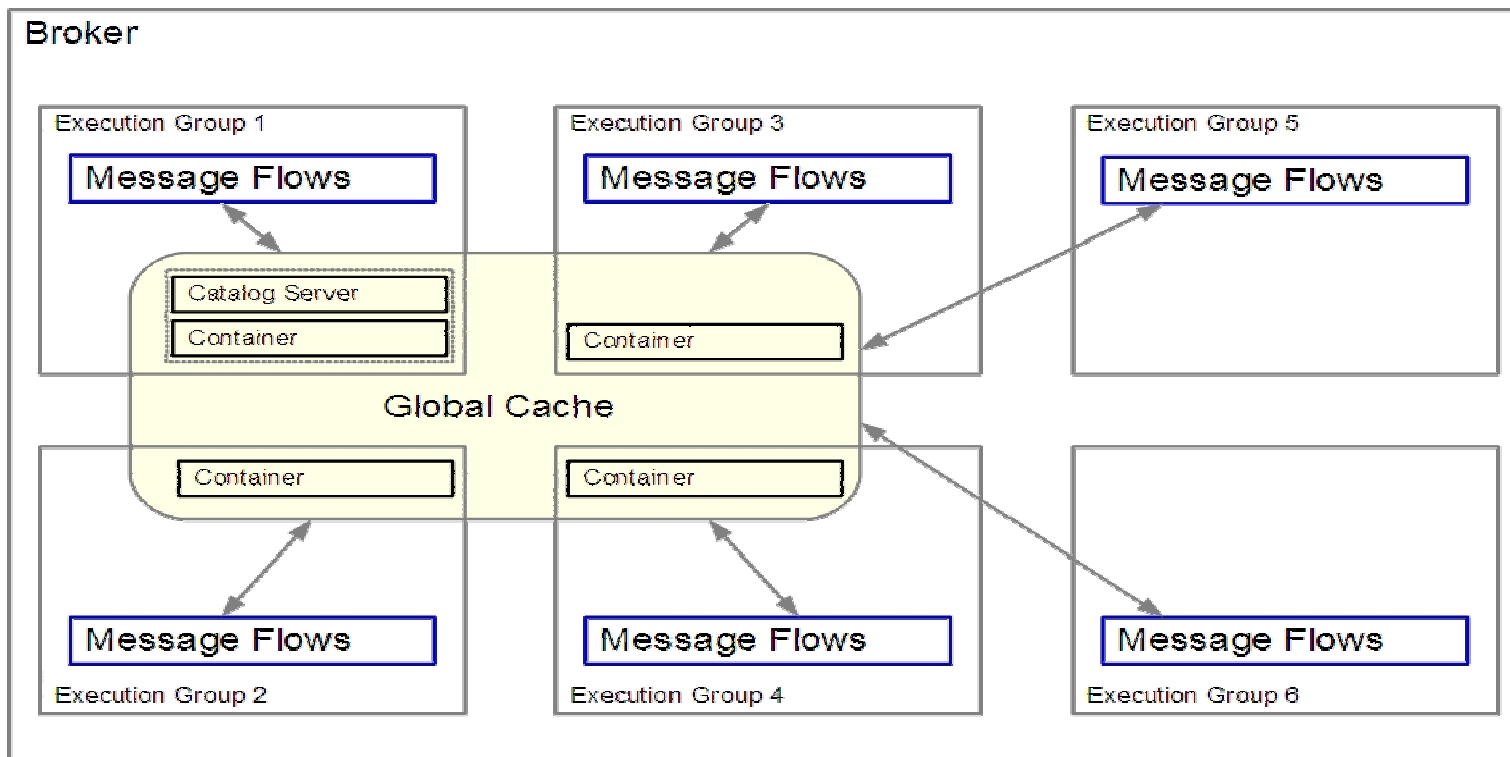
Topologies – default policy

- Default
  - All connections shown in the diagram below have out-of-the-box defaults!

= Clients connecting to catalog server
= WXS internal connections between catalog server and containers

EG1

Catalog Service

Container Service

Client (message flows)

EG2

Container Service

Client (message flows)

EG3

Container Service

Client (message flows)
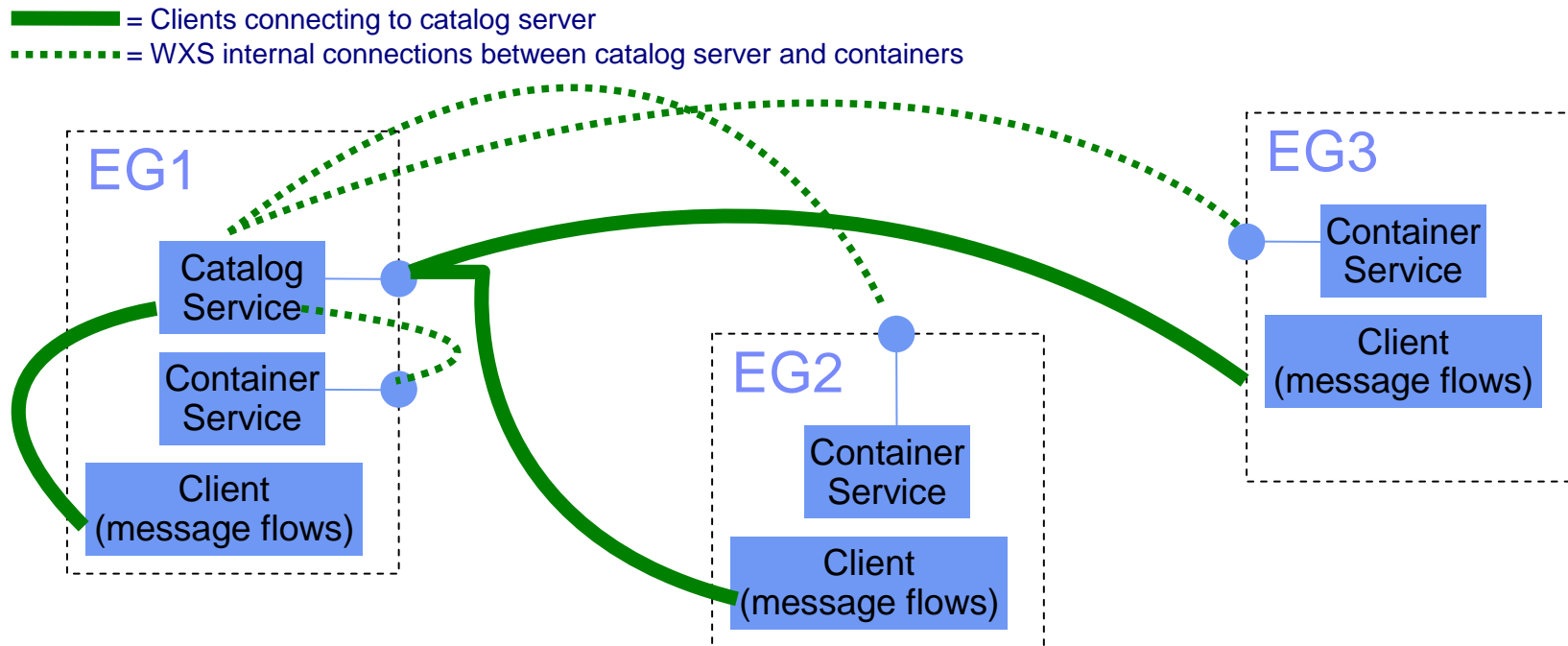
Topologies – default policy

Notes:

- Multiple execution groups/brokers participate together as a grid to host the cache
- Cache is distributed across all execution groups – with some redundancy
- One execution group is configured as the catalog server
- Distribution of "shards" handled by WXS
- Catalog server is required to resolve the physical container for each key
- Other execution groups are configured with the location of the catalog server
- Execution groups determine their own roles in the cache dynamically on startup
- Flows in all execution groups can connect to the cache. They explicitly connect to the catalog server, and are invisibly routed to the correct containers for each piece of data

Topologies – default policy, continued

- The execution group reports which role it is playing in the topology

Complete your sessions evaluation online at SHARE.org/BostonEval

Topologies – policy files

- Specify the fully-qualified path to an XML file to create a multi-broker cache
- In the file, list all brokers that are to participate in the cache
- Specify the policy file as the policy property value on all brokers that are to participate
- Sample policy files are included in the product install
- Policy files are validated when set and also when used
- Each execution group uses the policy file to determine its own role, and the details of all catalog servers
- For each broker, list the number of catalog servers it should host (0, 1 or 2) and the port range it should use. Each execution group uses its broker's name and listener host property to find itself in the file
- When using multiple catalog servers, some initial handshaking has to take place between them. Start them at the same time, and allow an extra 30-90 seconds for the cache to become available

Topologies – policy files, continued

- The policy file below shows two brokers being joined, with only one catalog server.
  - MQ04BRK will host up to 4 container servers, using ports in the range 2820-2839.
  - JAMES will host a catalog server and up to 4 containers, using ports in the range 2800-2819.
  - All execution groups in both brokers can communicate with the cache.

```
MyPolicy.xml - WordPad

File  Edit  View  Insert  Format  Help

<?xml version="1.0" encoding="UTF-8"?>
<cachePolicy xmlns="http://www.ibm.com/xmlns/prod/websphere/messagebroker/globalcache/policy-1.0">
        <broker name="MQ04BRK" listenerHost="WINMVSD1.HURSLEY.IBM.COM">
                <catalogs>0</catalogs>
                <portRange>
                        <startPort>2820</startPort>
                        <endPort>2839</endPort>
                </portRange>
        </broker>
        <broker name="JAMES" listenerHost="lefkas.hursley.ibm.com">
                <catalogs>1</catalogs>
                <portRange>
                        <startPort>2800</startPort>
                        <endPort>2819</endPort>
                </portRange>
        </broker>
</cachePolicy>

For Help, press F1                                                    NUM
```

2 Broker topology

- Demonstrates a more advance configuration which could be achieved using a policy file

Complete your sessions evaluation online at SHARE.org/BostonEval
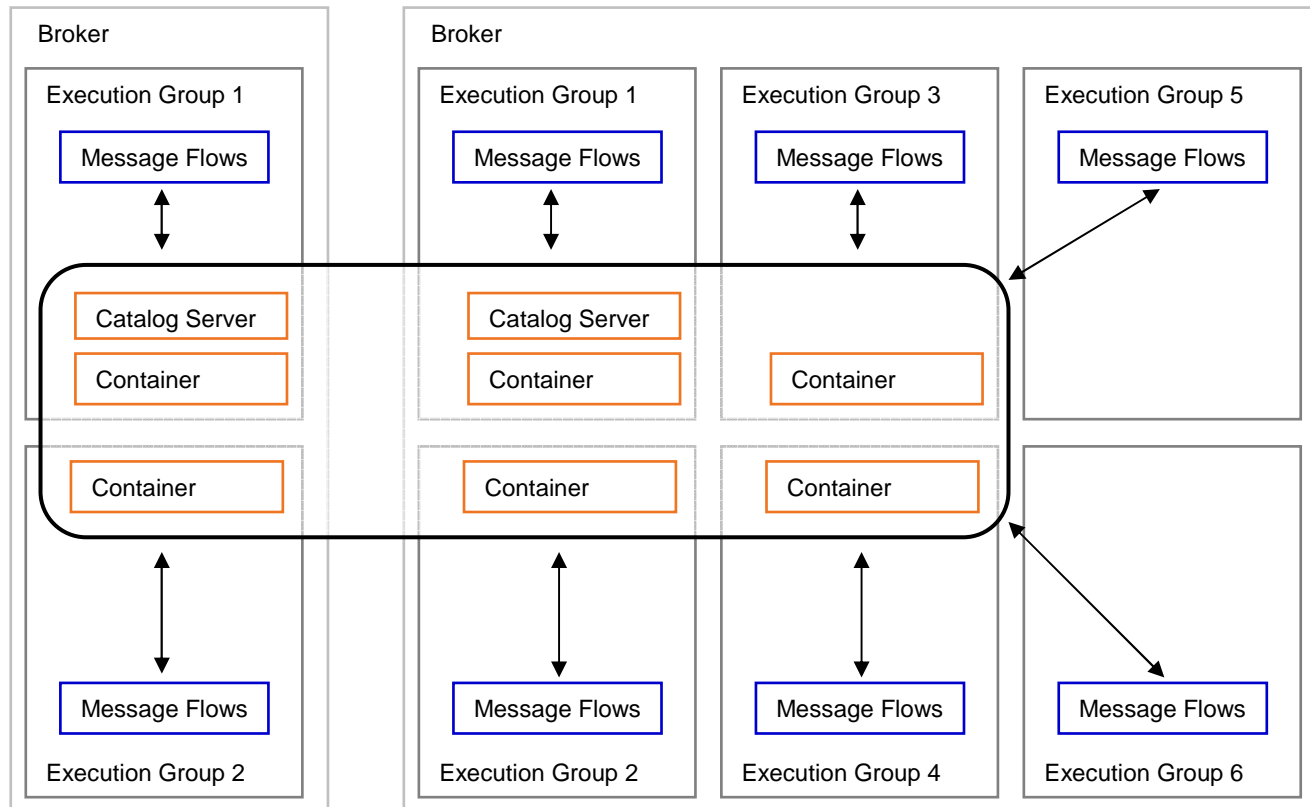
Notes

- This picture demonstrates a 2 broker topology that could be achieved by using a policy file or with manual configuration.
- This topology has more resilience than the previous default topology because there is more than 1 catalog server and they are split across multiple brokers.
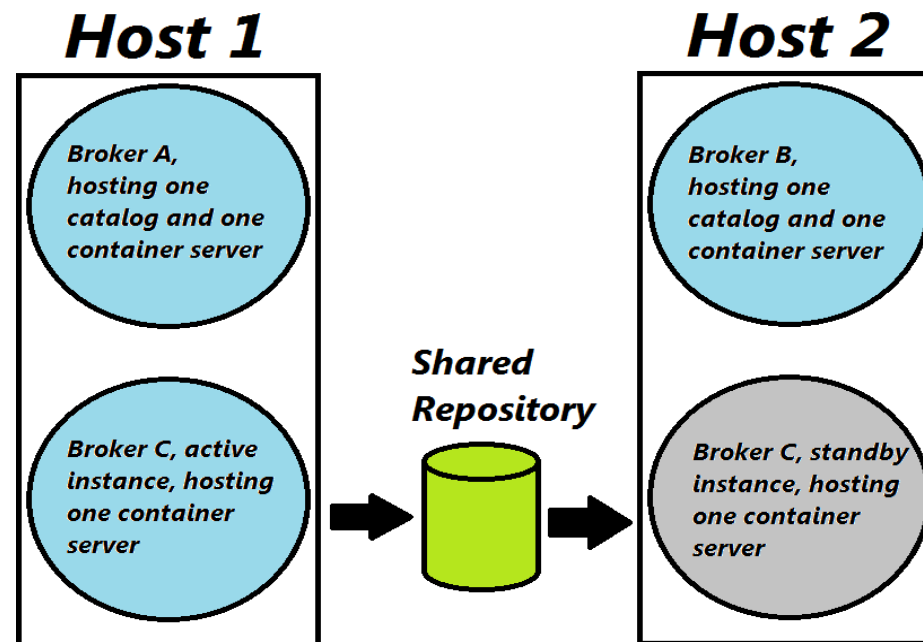
Complete your sessions evaluation online at SHARE.org/BostonEval

Topologies – policy "none"

- None. Switches off the broker level policy. Use this if you want to configure each execution group individually. The screenshot below shows the execution group-level properties that are available with this setting.
    - Useful for fixing specific cache roles with specific execution groups.
    - You may wish to have dedicated catalog server execution groups.
    - Tip – start with "Default" or policy file, then switch to "None" and tweak the settings.
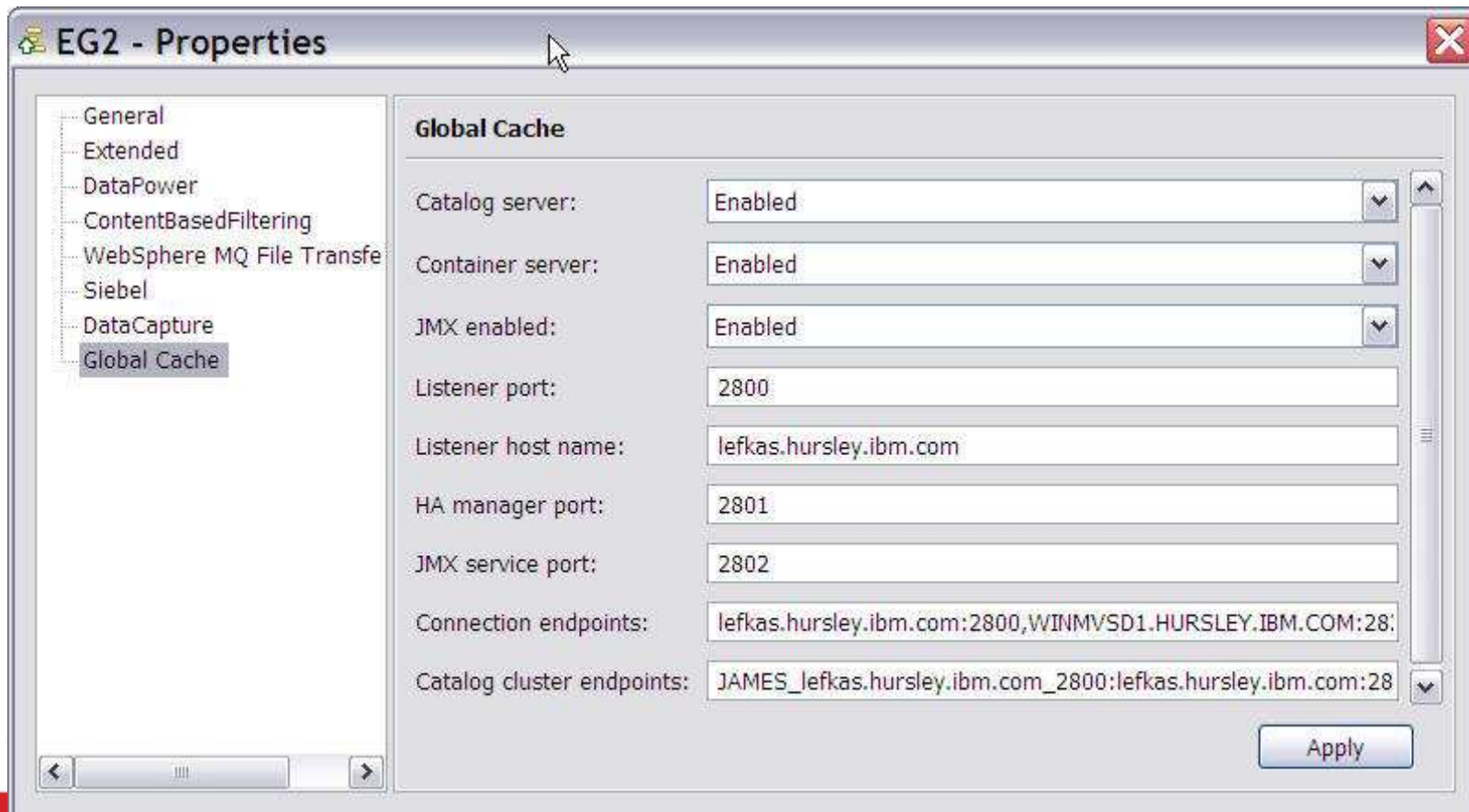
WXS Multi-instance Support

- Specify more than 1 listener host
- Configuration via policy file or IBX
- Policy file demonstrating a High Availability topology is kept in the product install
  - <install_dir>/sample/globalcache

Host 1

Broker A, hosting one catalog and one container server

Broker C, active instance, hosting one container server

Shared Repository

Host 2

Broker B, hosting one catalog and one container server

Broker C, standby instance, hosting one container server

- Broker reads policy file from absolute path to the file
  - Copy the directory structure
  - Mount to same place on both machines and keep on shared file system
- HA support is only for container servers

WXS Multi-instance Support
Cache Policy File

- "listenerHost" element as an alternative to "listenerHost" attribute

- "Old" 'listenerHost' attribute, now optional, can still be used to specify only one 'listenerHost'

```
<!-- The broker "CATBRK1" runs on "host1.ibm.com". -->
<broker name="CATBRK1" listenerHost="host1.ibm.com">
  <!-- This broker hosts one catalog server. -->
  <catalogs>1</catalogs>
```

- 'listenerHost' element can be used to specify one or two 'listenerHost' properties only if the Integration Node is not hosting a catalog server.

```
<!-- The multi-instance broker "MIBRK" runs on "host1.ibm.com" and "host2.ibm.com". -->
<broker name="MIBRK">
  <!-- This broker runs on "host1.ibm.com" and "host2.ibm.com". -->
  <listenerHost>host1.ibm.com</listenerHost>
  <listenerHost>host2.ibm.com</listenerHost>
  <!-- This broker hosts no catalog servers. -->
  <catalogs>0</catalogs>
```

WXS Multi-instance Support - Commands

- Both Integration Node and Integration Server level 'listenerHost' properties can be set to a comma separated list of values via mqsichangeproperties

  mqsichangeproperties MIBRK -b cachemanager -o CacheManager
           -n listenerHost -v **\"host1.ibm.com,host2.ibm.com\"**

  mqsichangeproperties MIBRK -e default -o ComIbmCacheManager
           -n listenerHost -v **\"host1.ibm.com,host2.ibm.com\"**

- All will be rejected if they are applied to a catalog server hosting broker
- Validation of other cache manager properties

Properties	Valid Values
enableCatalogService, enableContainerService, enableJMX	true, false
ListenerPort, haManagerPort, jmxServicePort	Integer: 1 – 65535 (inclusive)

WXS Multi-instance Support
IBM Integration Explorer

Setting multiple "listenerHosts" via IBM
Integration Explorer:

We're safely rejected if we try to set
this on a catalog server hosting
Integration Server:

Administrative information and tools

- Messages logged at startup to show global cache initialisation
- Resource statistics and activity trace provide information on map interactions.
- mqsicacheadmin command to provide advanced information about the underlying WXS grid.
  - Useful for validating that all specified brokers are participating in a multi-broker grid.
  - Also useful for checking that the underlying WXS grid has distributed data evenly across the available containers
  - Use with the "-c showMapSizes" option to show the size of your embedded cache.
  - Use with the "-c clearGrid -m <mapname>" option to clear data out of the cache.

Startup log messages

Event 7162, WebSphere Broker v8001

General | Details

(GLOBAL1.CACHE1) Catalog server "GLOBAL1_9.79.14.161_2840" in execution group "CACHE1" has started successfully.

A WebSphere eXtreme Scale catalog server has been started successfully in this execution group for use with the broker global cache.

None

Event 7155, WebSphere Broker v8001

General | Details

(GLOBAL1.CACHE3) The execution group has established a connection to the broker global cache.

A connection has been established to the broker global cache. The broker global cache is now available for use from message processing nodes.

Event 7163, WebSphere Broker v8001

General | Details

(GLOBAL1.CACHE3) Container server "GLOBAL1_9.79.14.161_2847" in execution group "CACHE3" has started successfully.

A WebSphere eXtreme Scale container server has been started successfully in this execution group for use with the broker global cache.

None

Resource statistics

- Resource statistics gives an overview of cache interactions

	GLOBAL1 Administration Log	GLOBAL1\CACHE2\GlobalCache - Activity Log		CACHE2 R

DotNet App Domains	CICS	DotNet GC	CORBA	ConnectDirect	FTEAgent	FTP	File
name	TotalMapActions		MapReads		MapWrites		MapRemoves
summary	7		3		3		1
WMB	7		3		3		1

esources Statistics (Snapshot time 18:17:31 - 18:17:51) ⊠

GlobalCache	JDBCConnectionPools	JMS	JVM	ODBC	Parsers	SOAPIr
FailedActions	MapsUsed		ConnectionFailures		Connects	
0	1		0		0	
0	1		0		0	

Activity Log

- Activity log gives a deeper view of cache interactions

Complete your sessions evaluation online at SHARE.org/BostonEval

Activity Log

- Cache activity is also interleaved with other flow activity

Complete your sessions evaluation online at SHARE.org/BostonEval

mqsicacheadmin command output

Global Cache Summary

- WXS Client hosted inside Execution Group
  - WXS Container can be hosted inside Execution Group
  - Catalog Server can be hosted inside Execution Group

- Type of topology controlled by broker cache "policy" property.
  - Options for default policy, disabled policy, xml-file specified policy, or no policy
  - All of the above available out of the box, with sensible defaults for a single-broker cache, in terms of number of containers, number of catalogs, ports used
  - Can be customized
  - Execution group properties / Message Broker Explorer for controlling which types of server are hosted in a given execution group, and their properties

- Access to map available in Java Compute node

- Access data in external grids

Gotchas

- JVM memory usage is higher with the global cache enabled
  - Typically at least 40-50mb higher
  - Consider having execution groups just to host the catalog and container servers

- 1 Catalog server is a single point of failure
  - Define a custom policy

- Cache update is autocommitted
  - Consider doing your cache actions as the last thing in the flow (after the MQOutput, for instance) to ensure you do not end up with data persisted after flow rollback.

- Port contention
  - Common causes of failure are port contention (something else is using one of the ports in your range), or containers failing because the catalog has failed to start.
  - In all cases, you should resolve the problem (choose a different port range, or restart the catalog EG) and then restart the EG.

Complete your sessions evaluation online at SHARE.org/BostonEval

Agenda

- Scenarios
- WebSphere eXtreme Scale concepts
- Programming model
- Operational model
- Questions?

?

Complete your sessions evaluation online at SHARE.org/BostonEval

This was session 13873 - The rest of the week ……

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00				Extending IBM WebSphere MQ and WebSphere Integration Bus to the Cloud	CICS and WMQ - The Resurrection of Useful
09:30	Introduction to MQ				Can I Consolidate My Queue Managers and Brokers?
11:00		MQ on z/OS - Vivisection	Hands-on Lab for MQ - take your pick!	MOBILE connectivity with Integration Bus	Migration and Maintenance, the Necessary Evil. Into the Dark for MQ and Integration Bus
12:15					
01:30	MQ Parallel Sysplex Exploitation, Getting the Best Availability From MQ on z/OS by Using Shared Queues	What's New in the MQ Family	MQ Clustering - The basics, advances and what's new	Using IBM WebSphere Application Server and IBM WebSphere MQ Together	
03:00	First Steps With Integration Bus: Application Integration for the Messy	What's New in Integration Bus	BIG Connectivity with mobile MQ	WebSphere MQ CHINIT Internals	
04:30	What's available in MQ and Broker for high availability and disaster recovery?	The Dark Side of Monitoring MQ - SMF 115 and 116 Record Reading and Interpretation	MQ & DB2 – MQ Verbs in DB2 & Q-Replication performance	Big Data Sharing with the Cloud - WebSphere eXtreme Scale and IBM Integration Bus	
06:00				WebSphere MQ Channel Authentication Records	

Complete your sessions evaluation online at SHARE.org/BostonEval