#SHAREorg

SHARE
Technology · Connections · Results

Big Data Sharing with the Cloud - WebSphere eXtreme Scale and WebSphere Message Broker Integration

David Coles – WebSphere Message Broker Level 3 Technical Lead,
IBM Hursley – dcoles@uk.ibm.com

Thursday 7th February 2013
12627

Agenda

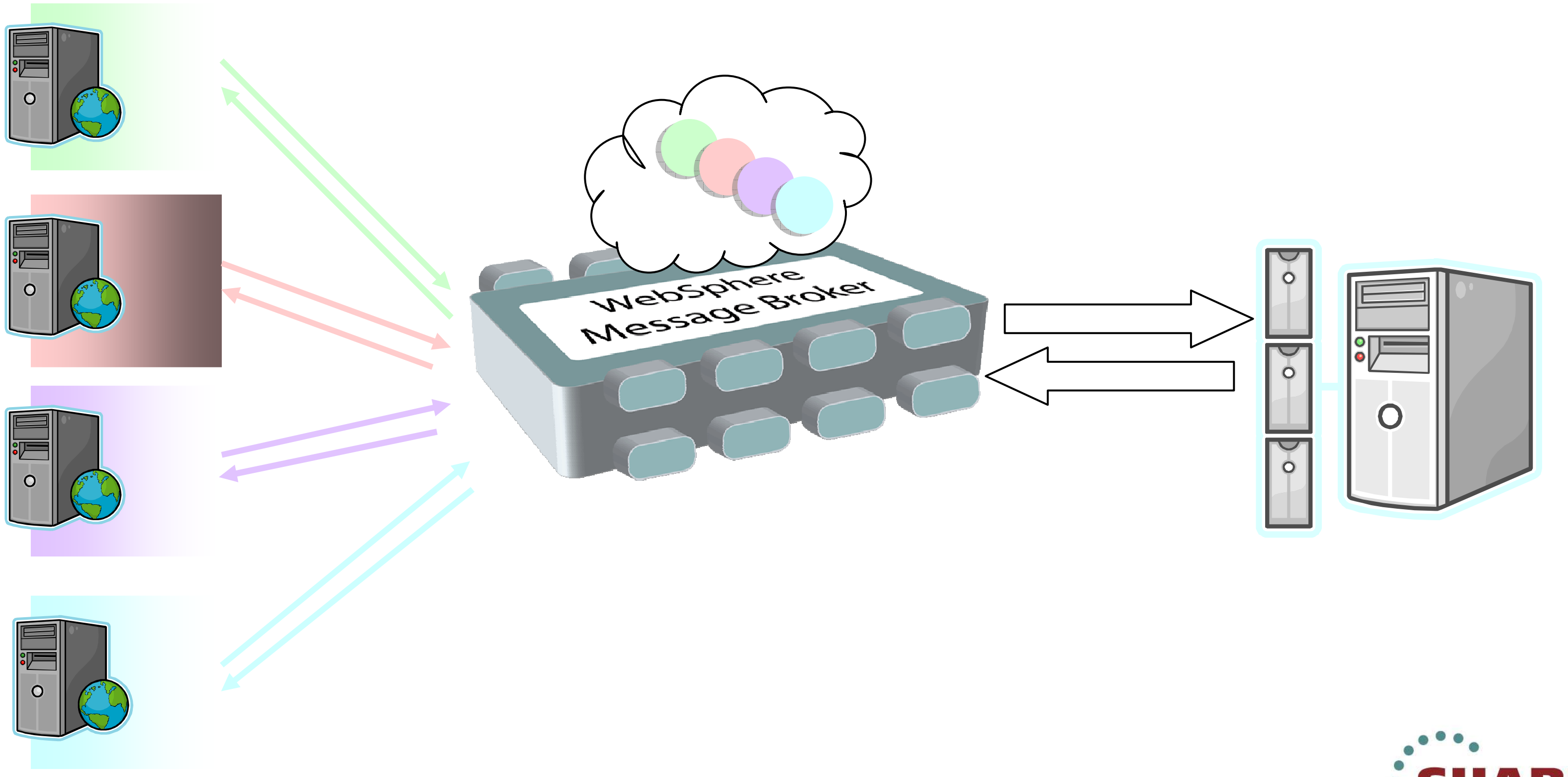
- Scenarios
- WebSphere eXtreme Scale concepts
- Programming model
- Operational model

Agenda

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

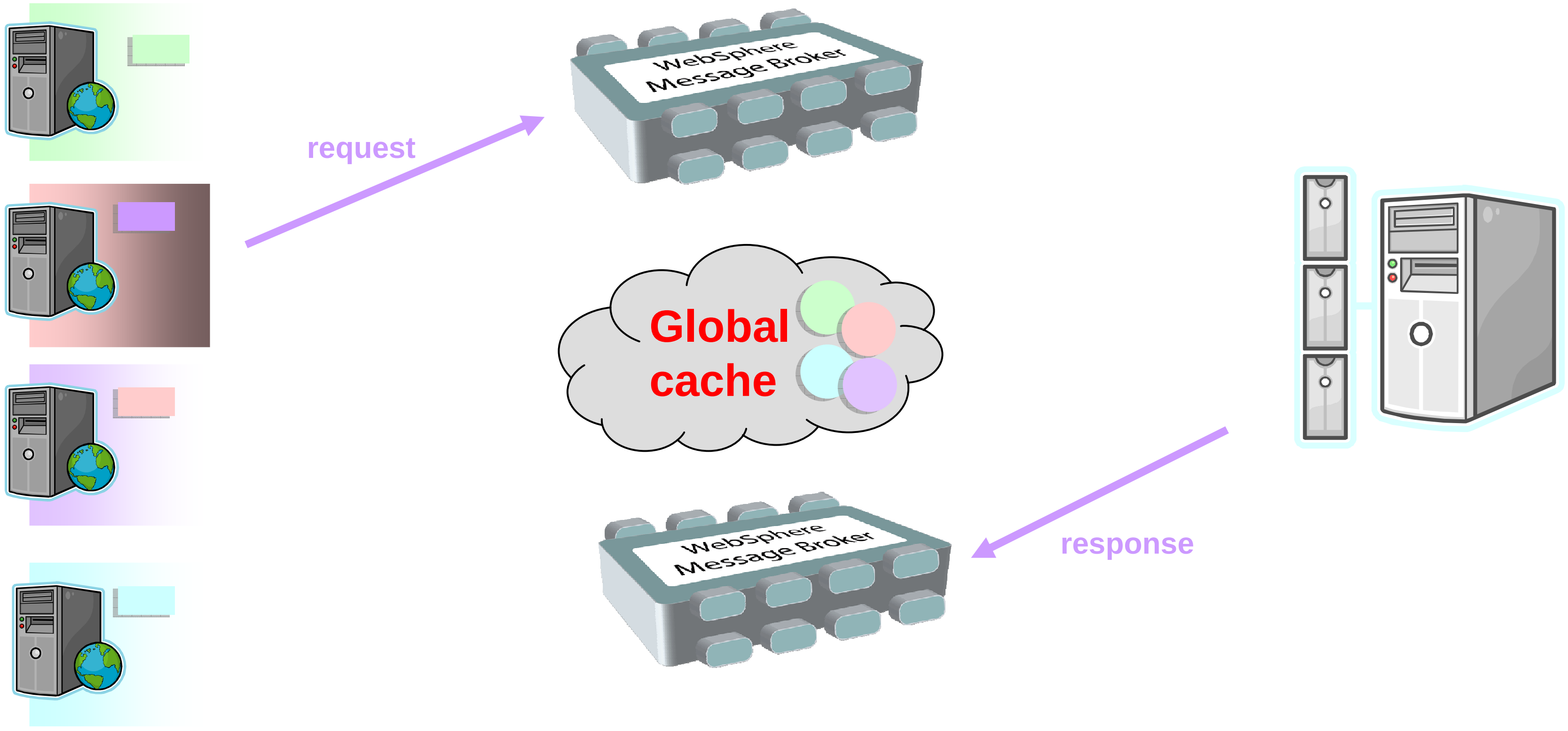
Scenario 1 - Storing state for integrations

- When WMB 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?



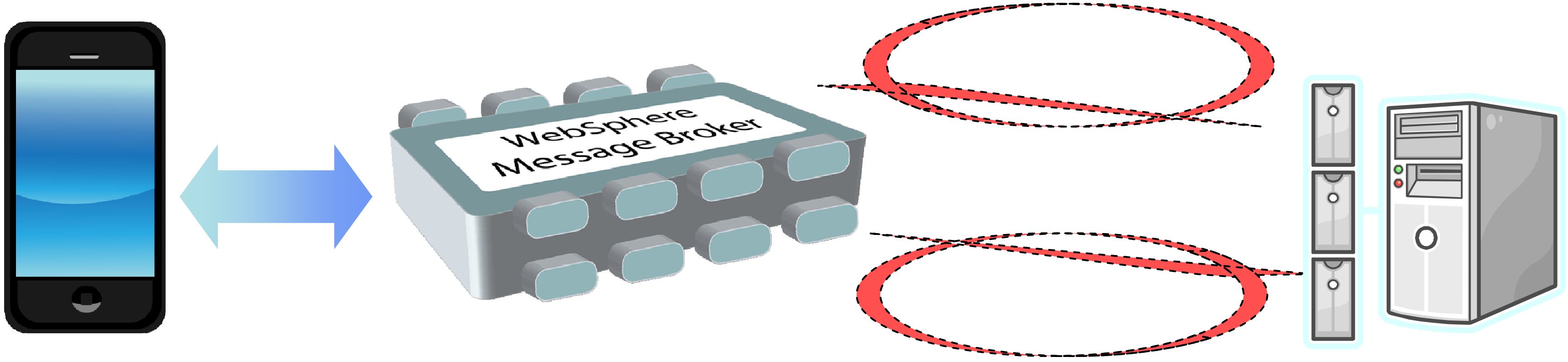
Scenario 1 - Storing state for integrations

- With a global cache, each broker can handle replies – even when the request was processed by another broker.



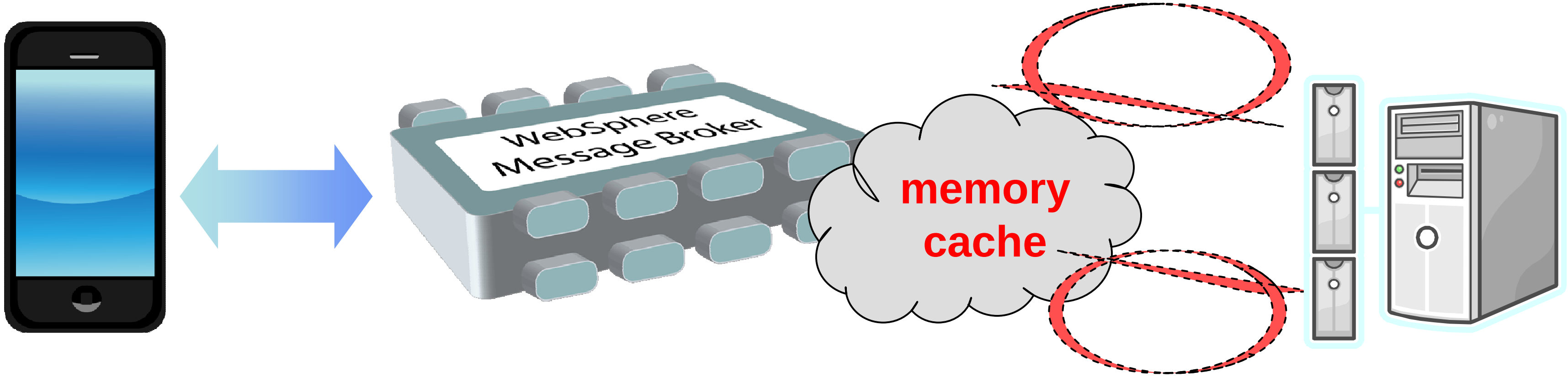
Scenario 2 - Caching static data

- When acting as a façade to a back-end data base, WMB needs to provide short response time to client, even though the backend has high latency.



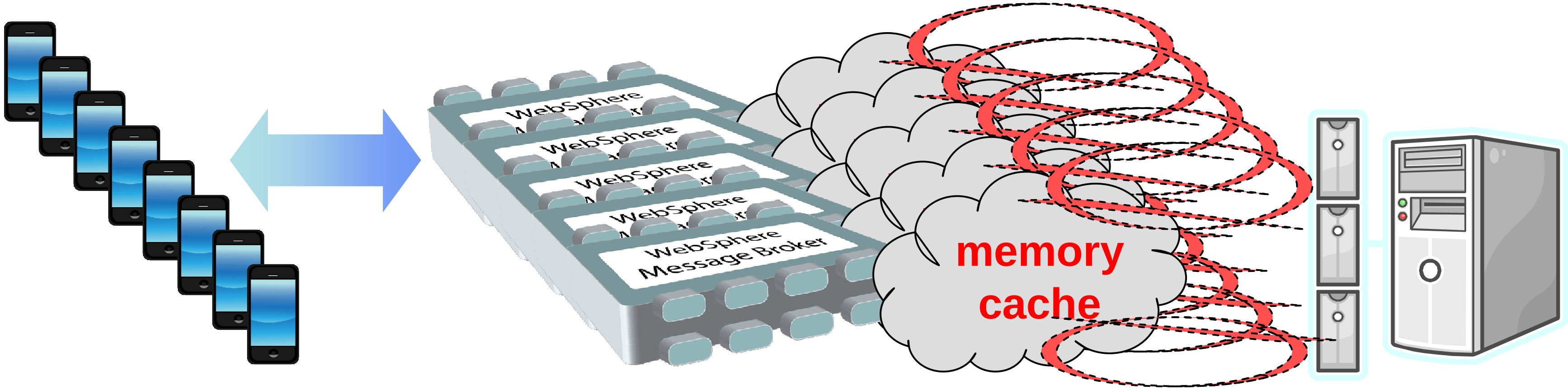
Scenario 2 - Caching static data

- When acting as a façade to a back-end data base, WMB 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).



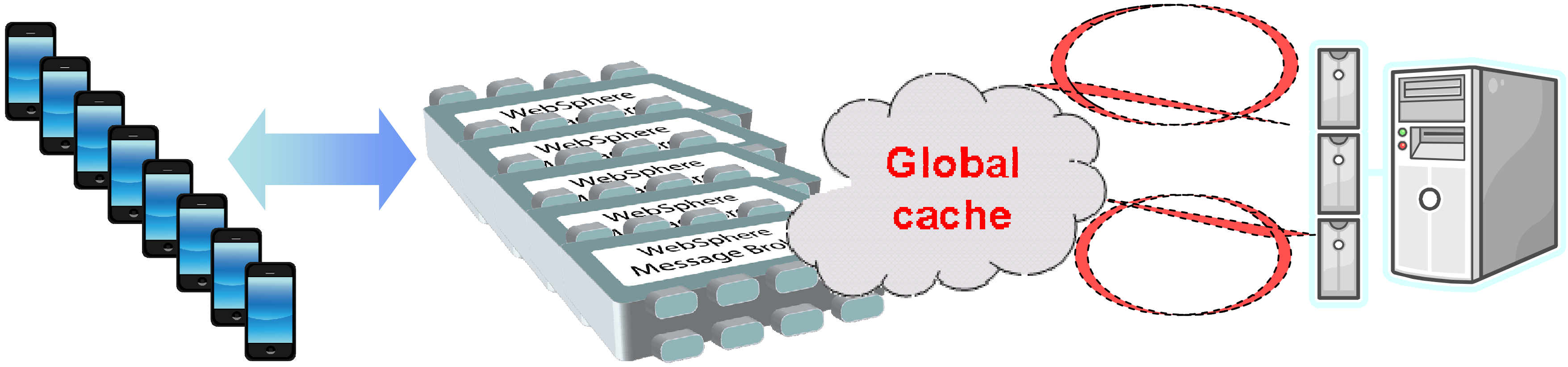
Scenario 2 - Caching static data

- When acting as a façade to a back-end data base, WMB 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



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

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

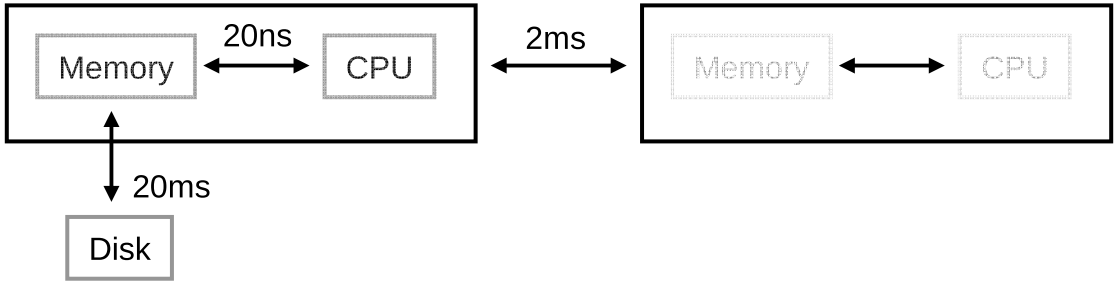
- Provides fault tolerance through replication with self monitoring and healing

- 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

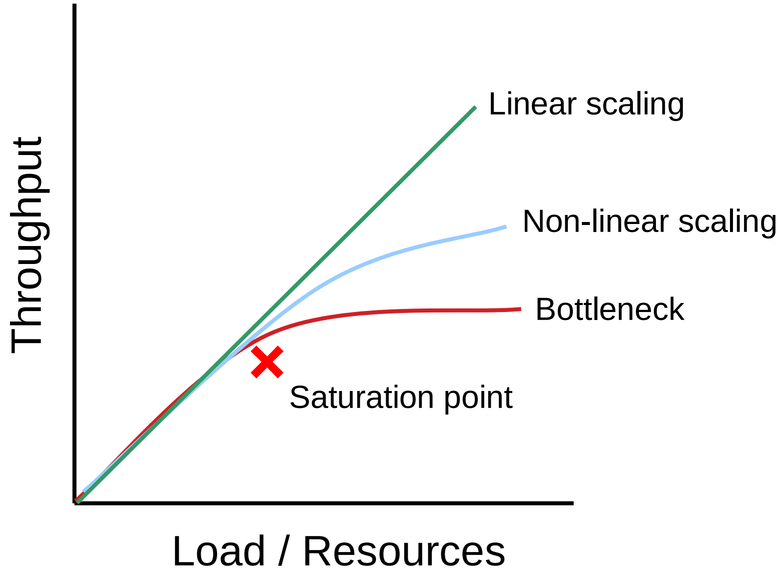


- Challenges
  - Physical memory size
  - Persistence
  - Resilience

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



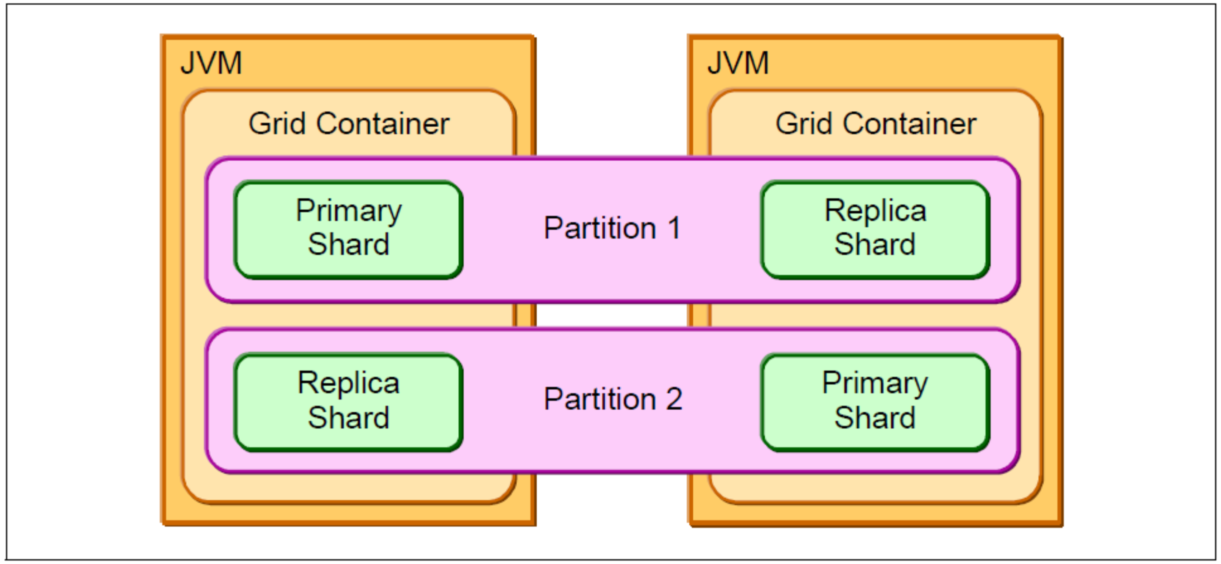
- Vertical or horizontal?

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.

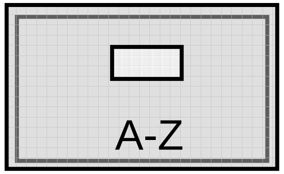
WXS Concepts

- Partition
- Container
- Catalog
- Shard
- Zone

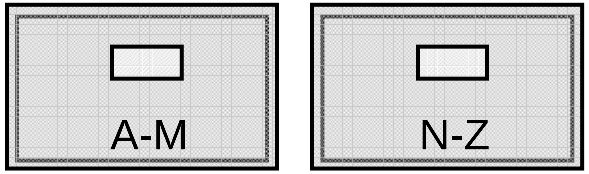


WXS Concepts - Partitions

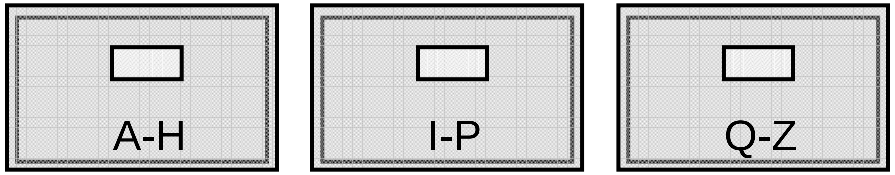
- Partitions
  - Think of a filing cabinet
  - To start off with you store everything in one draw



  - This fills up, so split across 2 draws



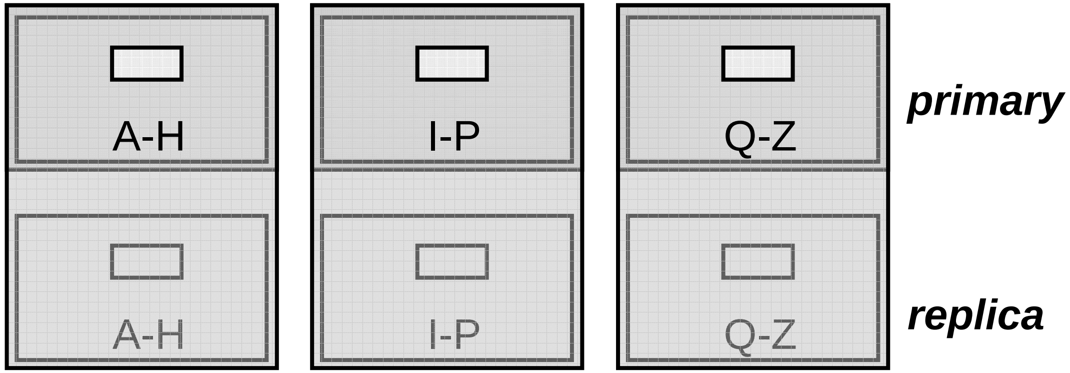
  - Then that fills up so you split across 3, etc



  - 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



- 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

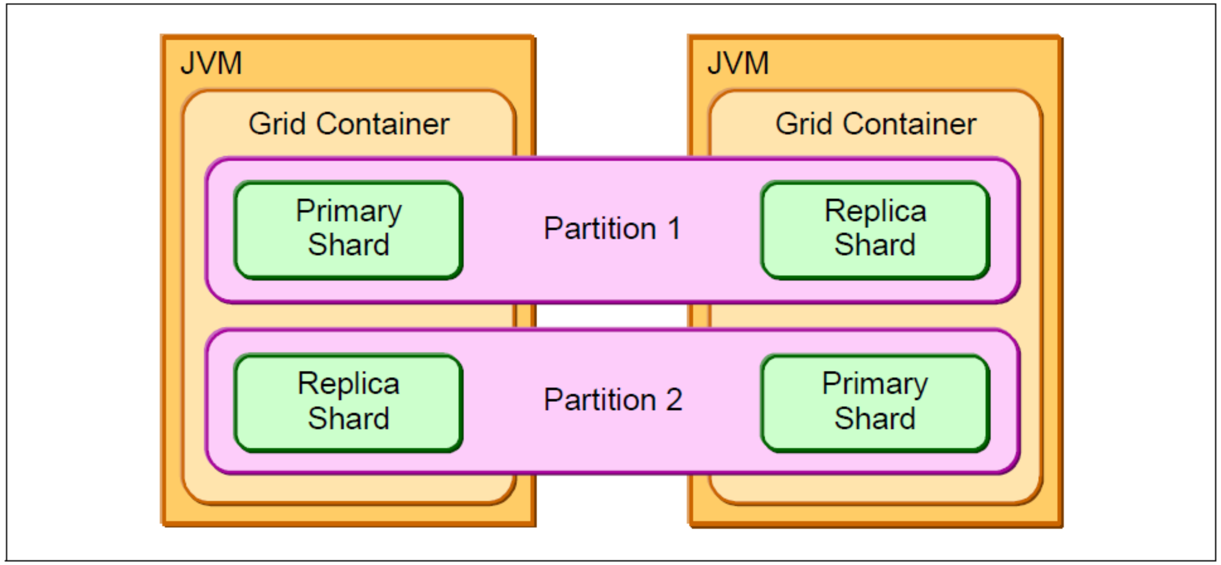
WXS Concepts – Container / Zone

- 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

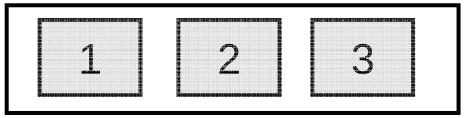
A Very Simple WXS Topology



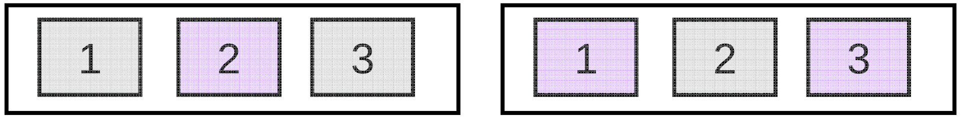
- You can replace JVM with
  - WebSphere Application Server
  - Datapower XC10
  - WebSphere Message Broker Execution Group

Elastic In-Memory Data Grid

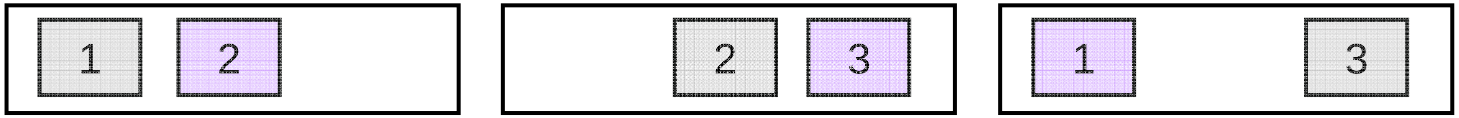
- Dynamically reconfigures as JVMs come and go
- 1 JVM



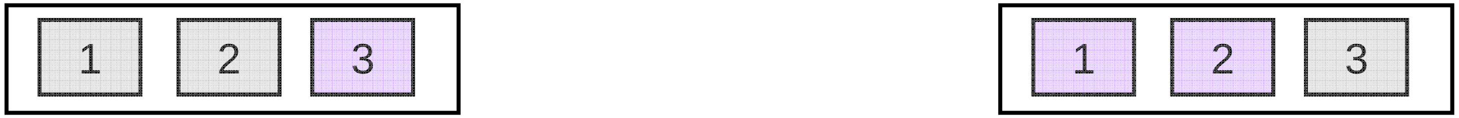
  - No replicas
- 2 JVMs



  - Replicas are created and distributed automatically
- 3 JVMs



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



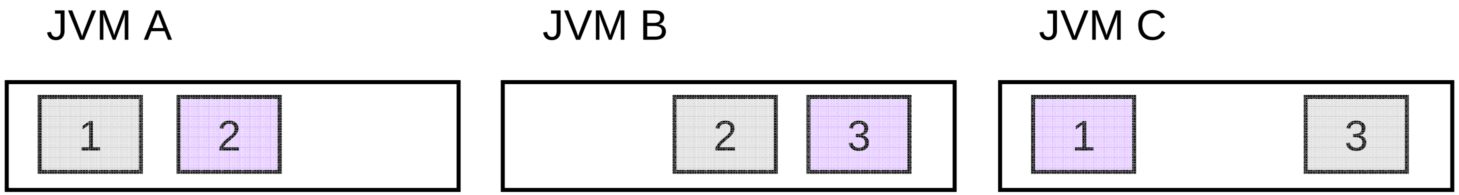
WXS Concepts - Catalog

Catalog

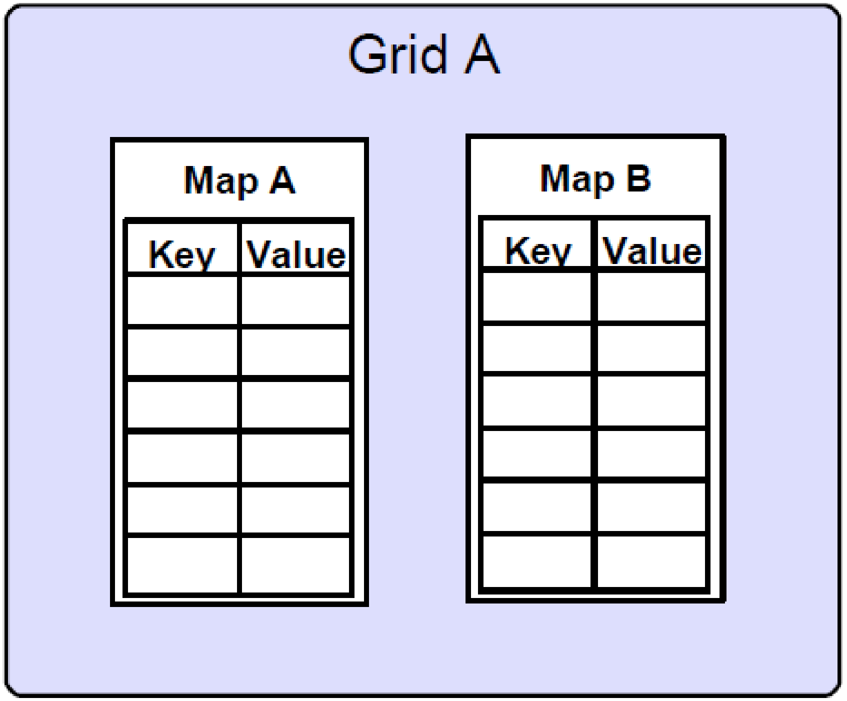
	Pri	Rep
A-H	A	C
I-P	B	A
Q-Z	C	B

- 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



User/Developer view



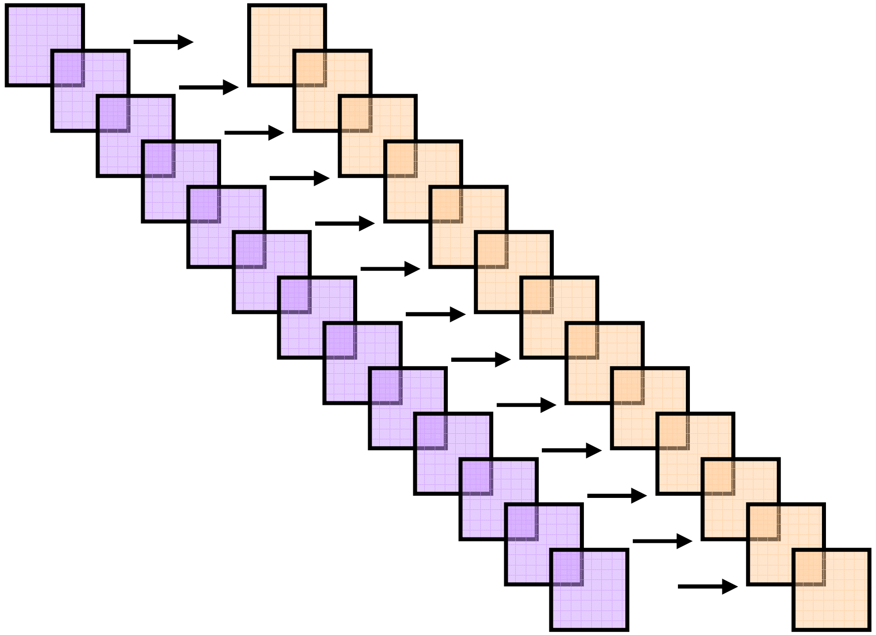
- A WXS developer sees only different maps of key value pairs

WMB Global Cache implementation

- WMB 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).

WMB 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

WMB 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

WMB Programming model - Java

```
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

WMB Programming model - Java

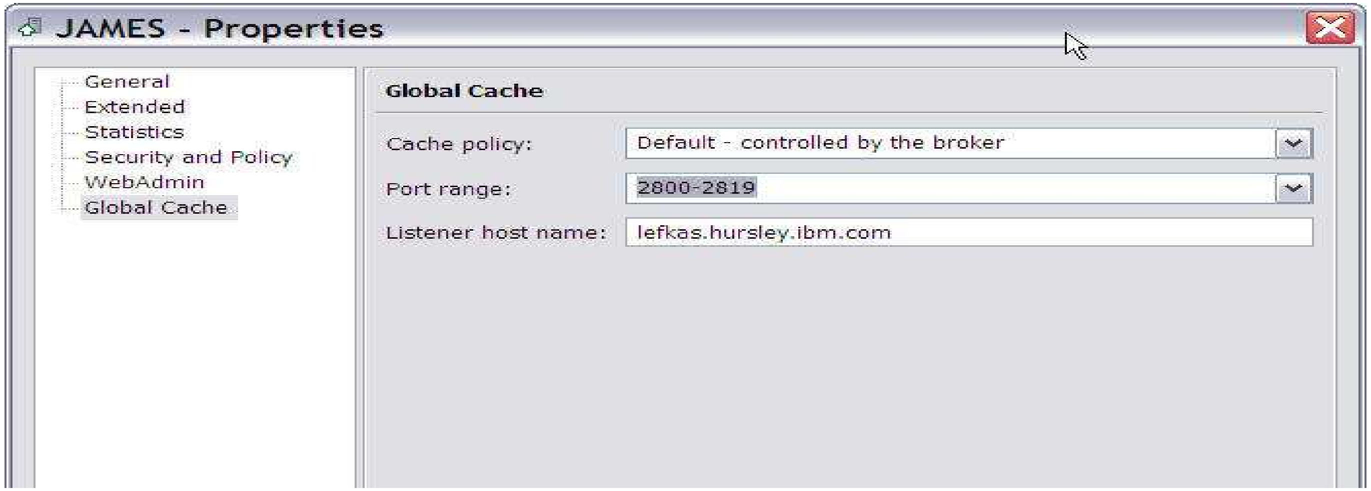
- Each statement is autcommitted
- Consider doing your cache actions as the last thing in the flow (after the MQOutput, for instance) to ensure you do not end up with data persisted after flow rollback
- put() will fail if the value already exists in the map. use containsKey() to check for existence 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

Agenda

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

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.



JAMES - Properties

Global Cache

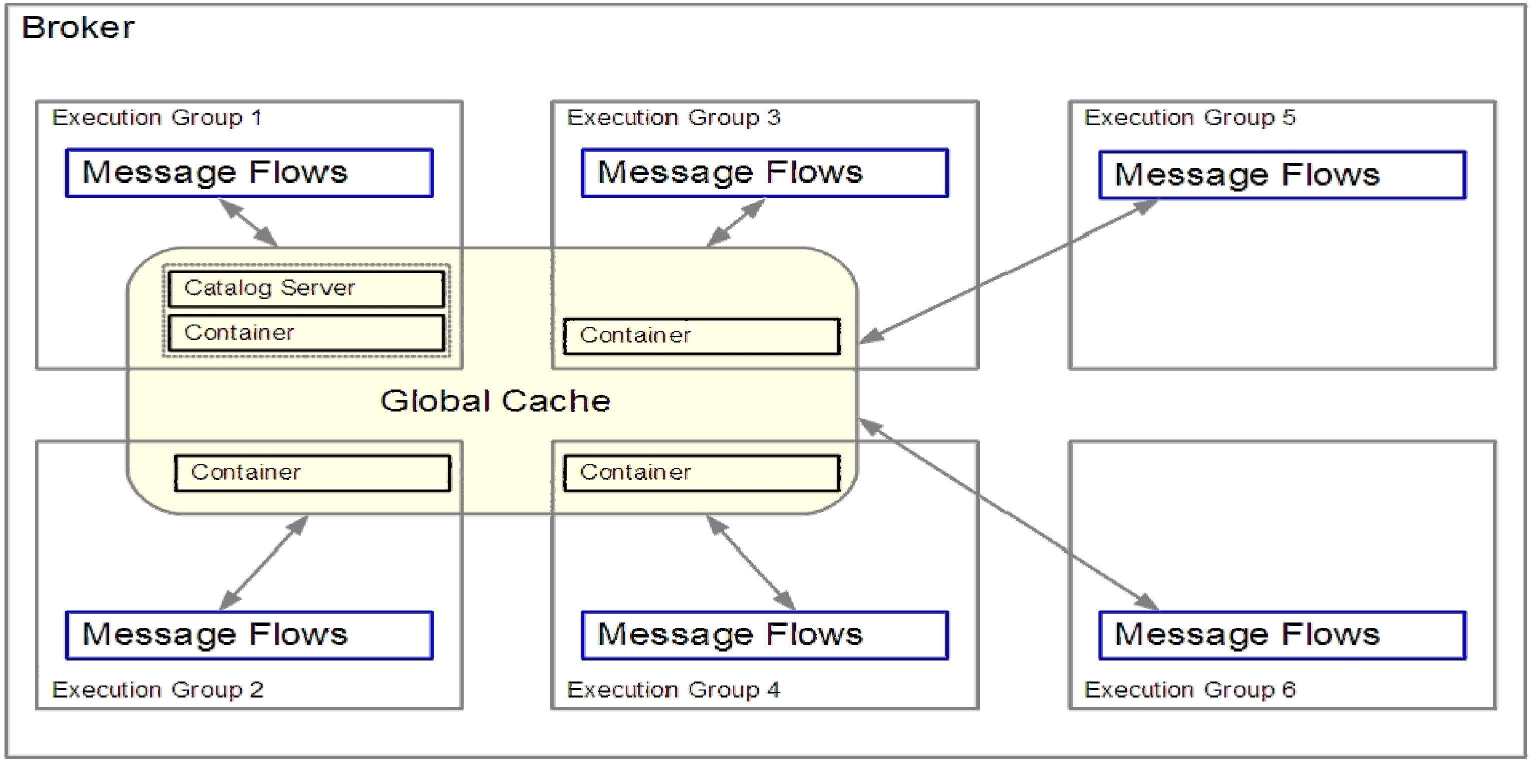
Cache policy: Default - controlled by the broker

Port range: 2800-2819

Listener host name: lefkas.hursley.ibm.com

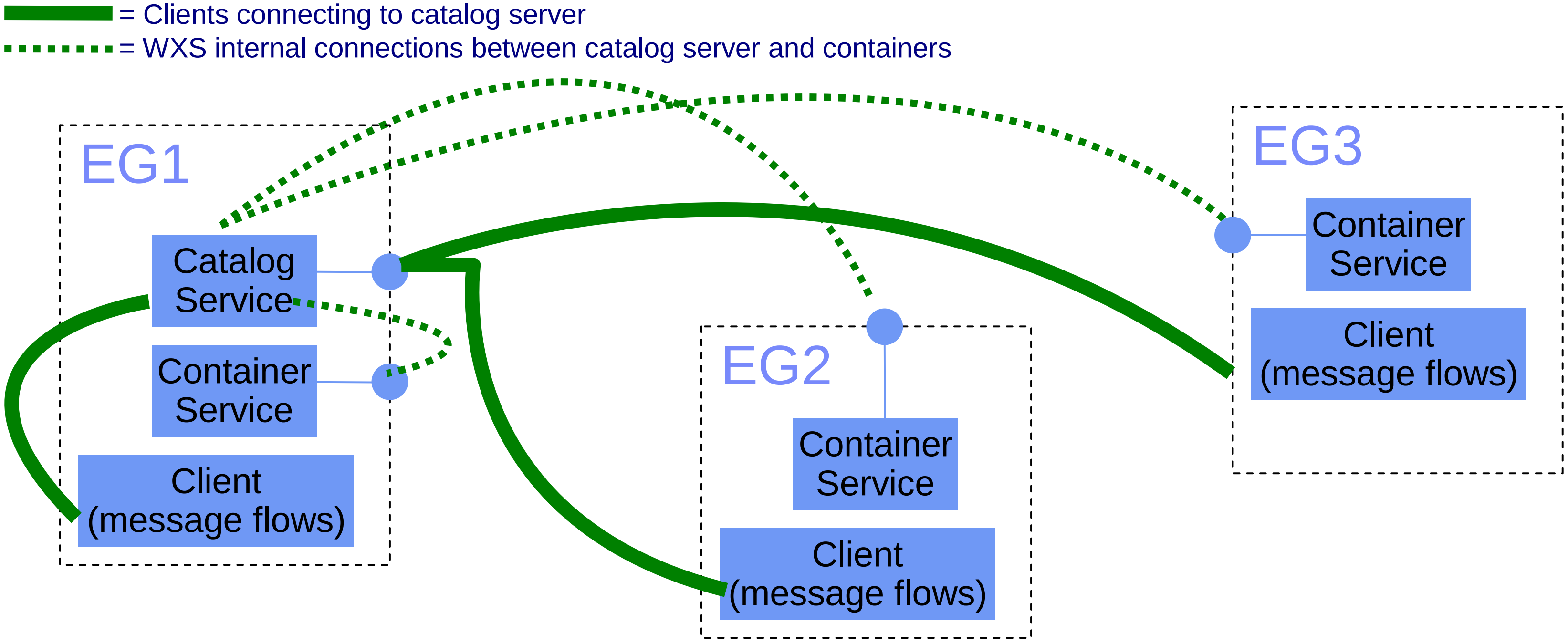
Diagram of the default topology

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



Topologies – default policy

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



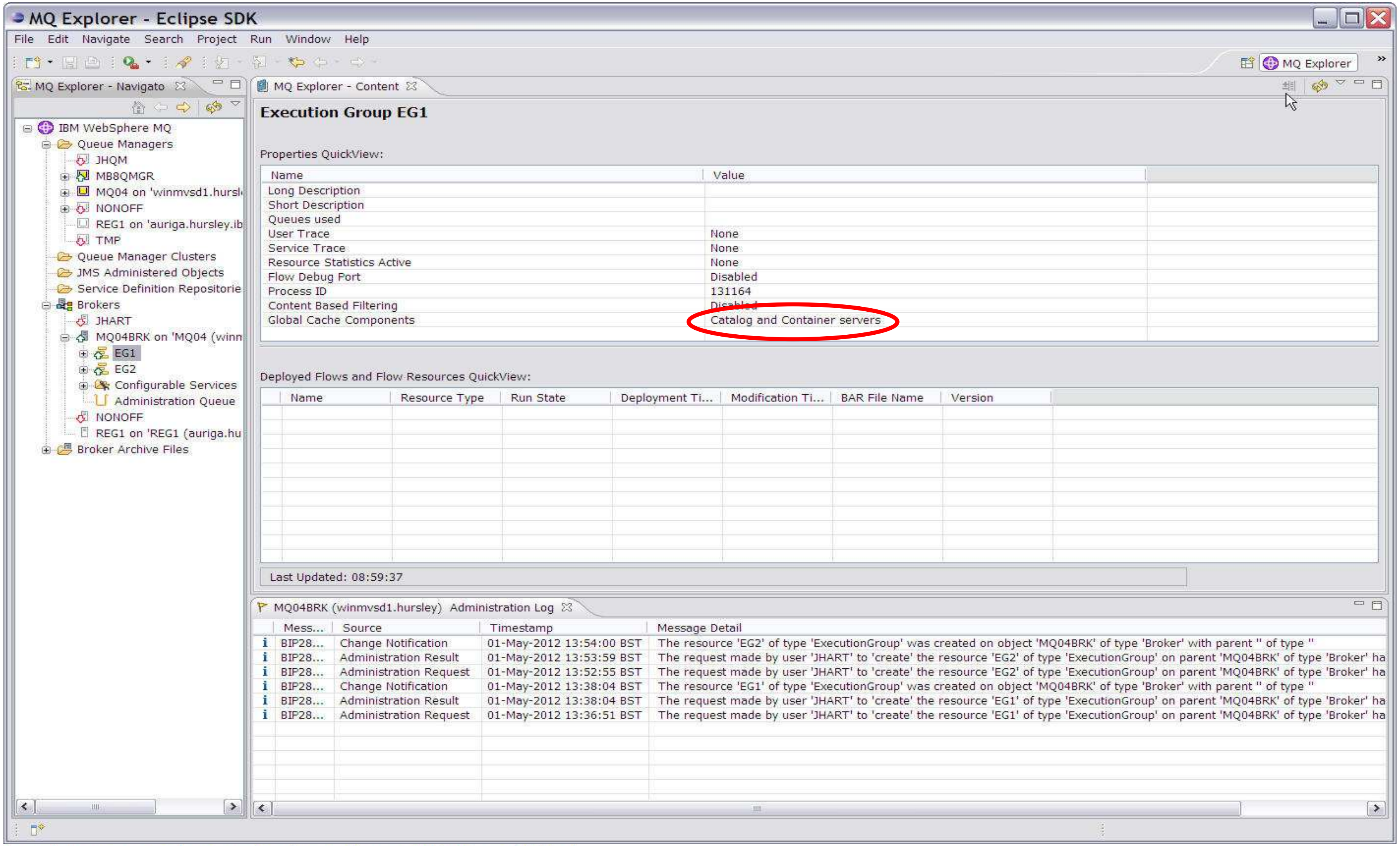
Topologies – default policy

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



Execution Group EG1

Properties QuickView:

Name	Value
Long Description	
Short Description	
Queues used	
User Trace	None
Service Trace	None
Resource Statistics Active	None
Flow Debug Port	Disabled
Process ID	131164
Content Based Filtering	Disabled
Global Cache Components	Catalog and Container servers

Deployed Flows and Flow Resources QuickView:

Name	Resource Type	Run State	Deployment Ti...	Modification Ti...	BAR File Name	Version

Last Updated: 08:59:37

MQ04BRK (winmvsd1.hursley) Administration Log

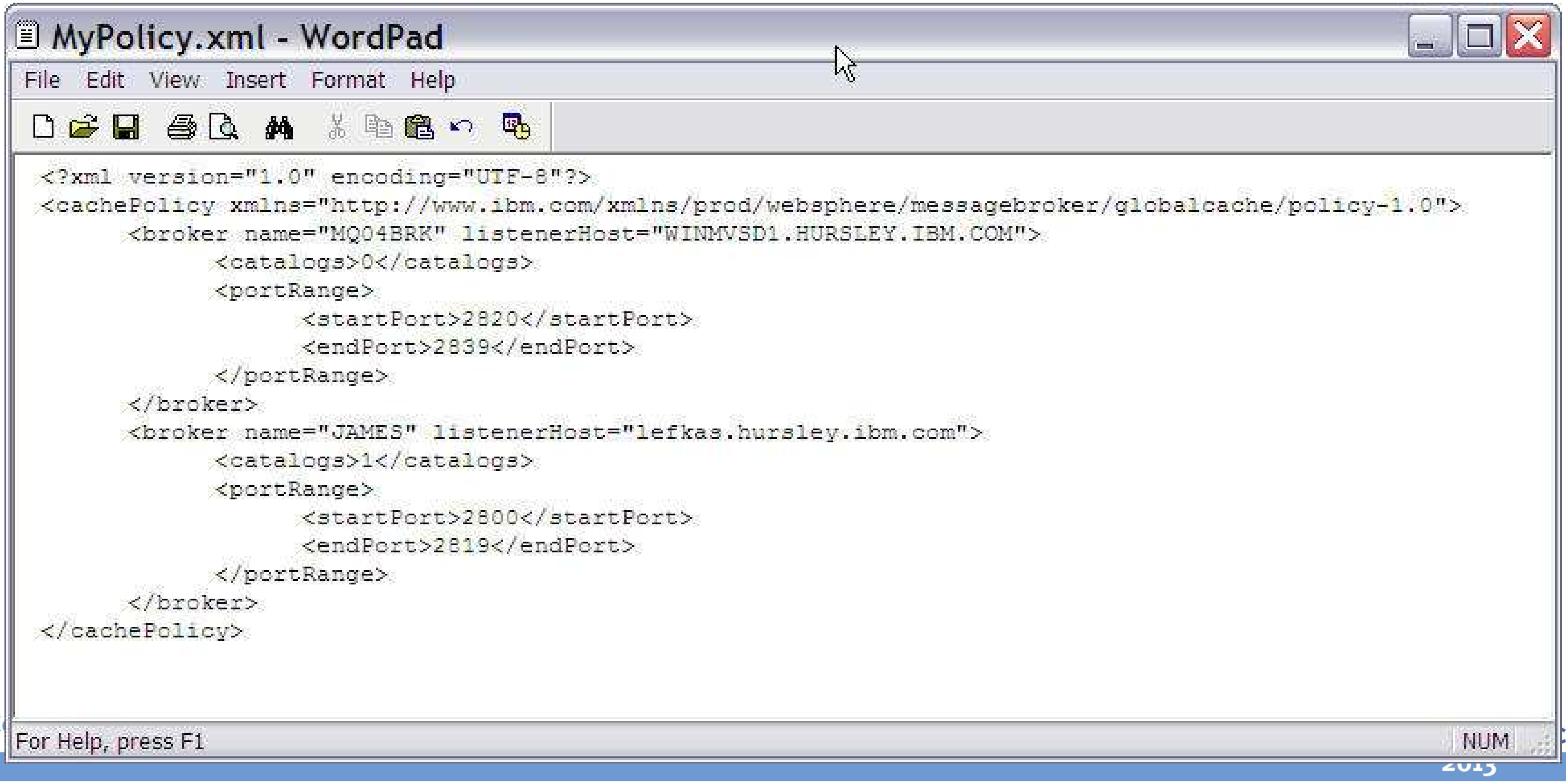
	Mess...	Source	Timestamp	Message Detail
i	BIP28...	Change Notification	01-May-2012 13:54:00 BST	The resource 'EG2' of type 'ExecutionGroup' was created on object 'MQ04BRK' of type 'Broker' with parent '' of type ''
i	BIP28...	Administration Result	01-May-2012 13:53:59 BST	The request made by user 'JHART' to 'create' the resource 'EG2' of type 'ExecutionGroup' on parent 'MQ04BRK' of type 'Broker' ha
i	BIP28...	Administration Request	01-May-2012 13:52:55 BST	The request made by user 'JHART' to 'create' the resource 'EG2' of type 'ExecutionGroup' on parent 'MQ04BRK' of type 'Broker' ha
i	BIP28...	Change Notification	01-May-2012 13:38:04 BST	The resource 'EG1' of type 'ExecutionGroup' was created on object 'MQ04BRK' of type 'Broker' with parent '' of type ''
i	BIP28...	Administration Result	01-May-2012 13:38:04 BST	The request made by user 'JHART' to 'create' the resource 'EG1' of type 'ExecutionGroup' on parent 'MQ04BRK' of type 'Broker' ha
i	BIP28...	Administration Request	01-May-2012 13:36:51 BST	The request made by user 'JHART' to 'create' the resource 'EG1' of type 'ExecutionGroup' on parent 'MQ04BRK' of type 'Broker' ha

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

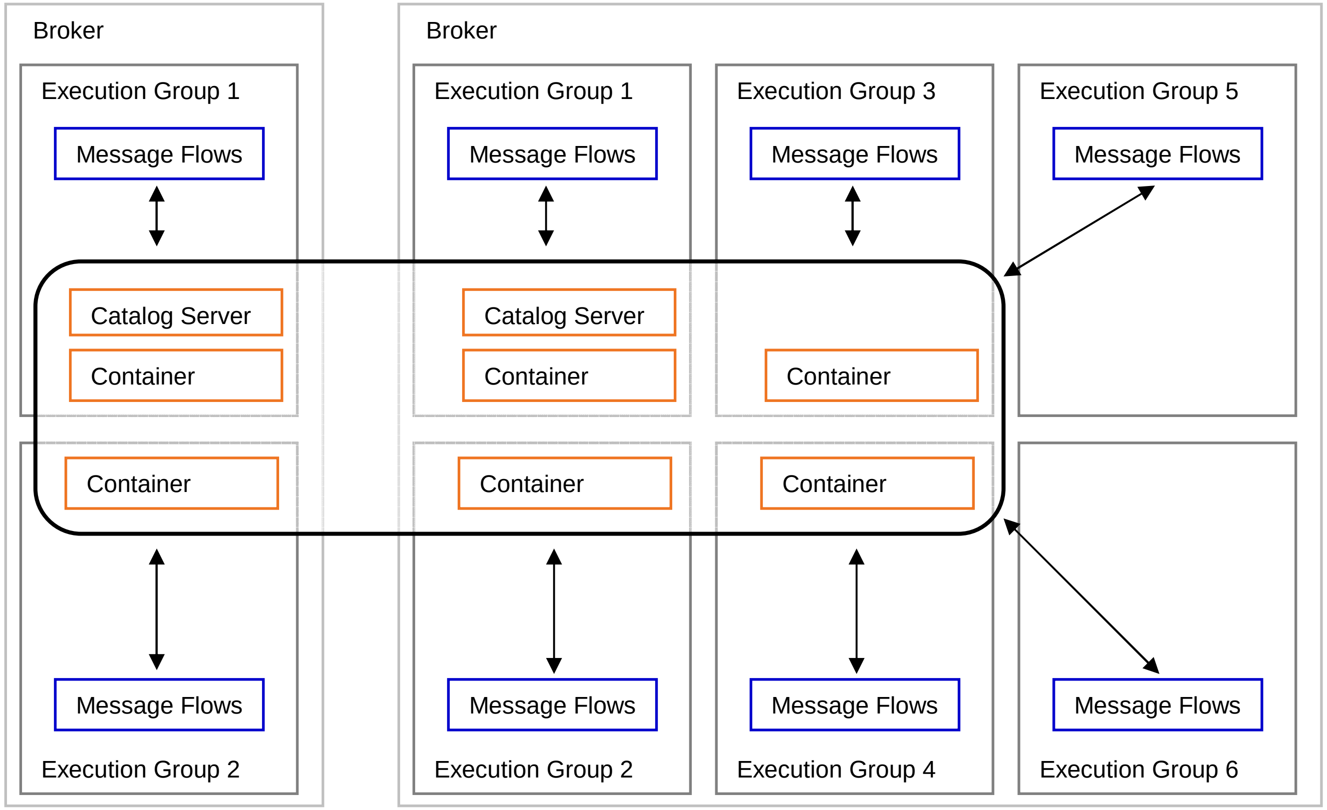
- The policy file below shows two brokers being joined, with only one catalog server.
  - MQ04BRK will host up to 4 container servers, using ports in the range 2820-2839.
  - JAMES will host a catalog server and up to 4 containers, using ports in the range 2800-2819.
  - All execution groups in both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



```
<?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>
```

2 Broker topology

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

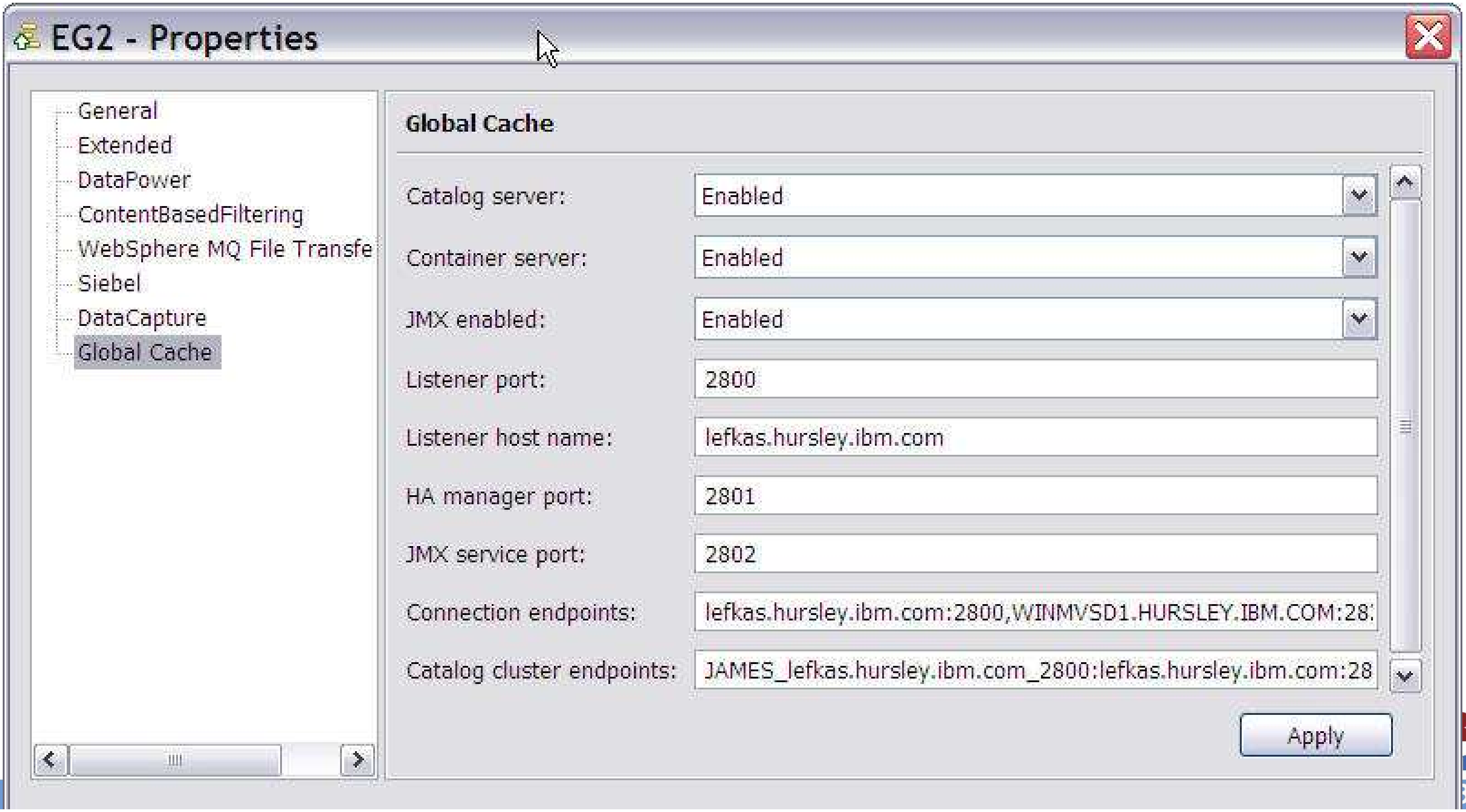


Notes

- 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.

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.



EG2 - Properties

Global Cache

Catalog server: Enabled

Container server: Enabled

JMX enabled: Enabled

Listener port: 2800

Listener host name: lefkas.hursley.ibm.com

HA manager port: 2801

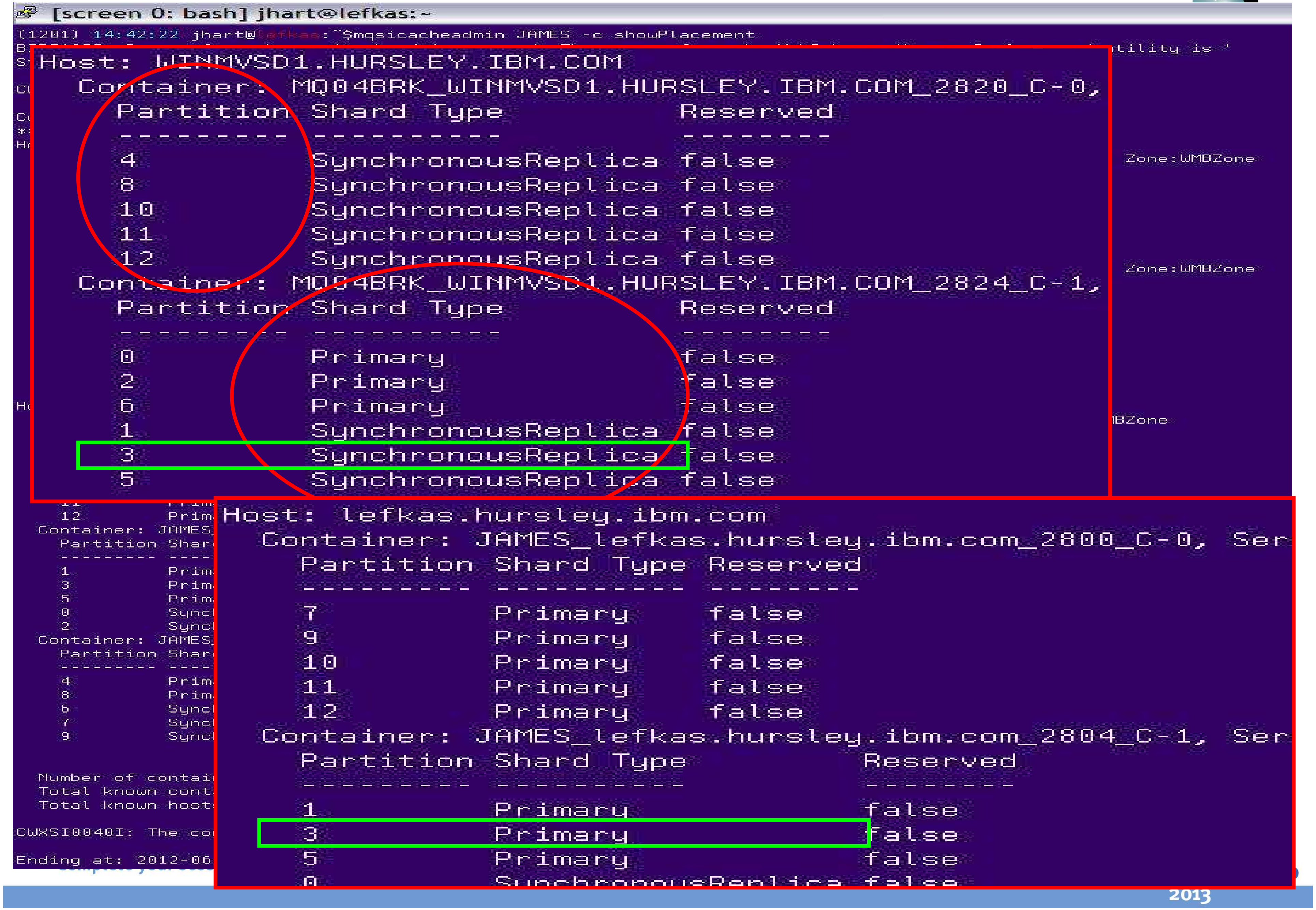
JMX service port: 2802

Connection endpoints: lefkas.hursley.ibm.com:2800,WINMVSD1.HURSLEY.IBM.COM:28

Catalog cluster endpoints: JAMES_lefkas.hursley.ibm.com_2800:lefkas.hursley.ibm.com:28

Administrative information and tools

- 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.
- Example mqsicacheadmin output on the next slide.



```
[screen 0: bash] jhart@lefkas:~
(1201) 14:42:22 jhart@lefkas:~$mqsicacheadmin JAMES -c showPlacement
```

```
Host: WINMVSD1.HURSLEY.IBM.COM
  Container: MQ04BRK_WINMVSD1.HURSLEY.IBM.COM_2820_C-0,
    Partition Shard Type              Reserved
    --------- ----------              ---------
    4         SynchronousReplica false
    8         SynchronousReplica false
    10        SynchronousReplica false
    11        SynchronousReplica false
    12        SynchronousReplica false
  Container: MQ04BRK_WINMVSD1.HURSLEY.IBM.COM_2824_C-1,
    Partition Shard Type              Reserved
    --------- ----------              ---------
    0         Primary                 false
    2         Primary                 false
    6         Primary                 false
    1         SynchronousReplica false
    3         SynchronousReplica false
    5         SynchronousReplica false
```

```
tility is '
Zone:WMBZone
Zone:WMBZone
1BZone
```

```
    12        Prim
  Container: JAMES_
    Partition Shar
    --------- ----
    1         Prim
    3         Prim
    5         Prim
    0         Sync
    2         Sync
  Container: JAMES_
    Partition Shar
    --------- ----
    4         Prim
    8         Prim
    6         Sync
    7         Sync
    9         Sync

  Number of contai
  Total known cont
  Total known host

CWXSI0040I: The co

Ending at: 2012-06
```

```
Host: lefkas.hursley.ibm.com
  Container: JAMES_lefkas.hursley.ibm.com_2800_C-0, Ser
    Partition Shard Type Reserved
    --------- ---------- ---------
    7         Primary    false
    9         Primary    false
    10        Primary    false
    11        Primary    false
    12        Primary    false
  Container: JAMES_lefkas.hursley.ibm.com_2804_C-1, Ser
    Partition Shard Type              Reserved
    --------- ----------              ---------
    1         Primary                 false
    3         Primary                 false
    5         Primary                 false
    0         SynchronousReplica false
```

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

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.

Agenda

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

?

This was session 12627 - The rest of the week ……

	Monday	Tuesday	Wednesday	Thursday	Friday
08:00					Are you running too many queue managers or brokers?
09:30		What's New in WebSphere Message Broker			Diagnosing Problems for MQ CICS and WMQ - The Resurrection of Useful
11:00		Extending IBM WebSphere MQ and WebSphere Message Broker to the Cloud	WMQ - Introduction to Dump Reading and SMF Analysis - Hands-on Lab	BIG Data Sharing with the cloud - WebSphere eXtreme Scale and WebSphere Message Broker integration	Getting the best availability from MQ on z/OS by using Shared Queues
12:15					
01:30	Introduction to MQ	MQ on z/OS – Vivisection	Migration and maintenance, the necessary evil	The Dark Side of Monitoring MQ - SMF 115 and 116 Record Reading and Interpretation	
03:00	First Steps With WebSphere Message Broker: Application Integration for the Messy	BIG Connectivity with WebSphere MQ and WebSphere Message Broker	WebSphere MQ CHINIT Internals	Using IBM WebSphere Application Server and IBM WebSphere MQ Together	
04:30	WebSphere MQ application design, the good, the bad and the ugly	What's New in the WebSphere MQ Product Family	MQ & DB2 – MQ Verbs in DB2 & Q-Replication	WebSphere MQ Channel Authentication Records	
06:00			Clustering - The Easier Way to Connect Your Queue Managers		