IIB and Caching

WebSphere User Group UK
IBM South Bank - March 2015

James Hart – IBM Integration Bus L3 Service
Agenda

- Embedded “Global Cache”
- Connecting to external WebSphere eXtreme Scale grids.
- Best practices & and tips
Agenda

- Embedded “Global Cache”
- Connecting to external WebSphere eXtreme Scale grids.
- Best practices & and tips
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 infrequently changing data

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

- **Elastic “In-Memory” Data Grid**

- **Virtualizes free memory within a grid of JVMs into a single logical space**
  - Accessible as partitioned, key addressable map by applications and subsystems

- **Provides fault tolerance through replication**
  - e.g. Primary/secondary stores with failover, voting etc…

- **Easy to Scale**
  - Add more JVMs dynamically while it’s running without restart

- **Available as component or standalone software and hardware appliance**
  - Foundational technology used “under the covers” in Message Broker
IIB Global Cache in A Nutshell

- **IIB contains an embedded WebSphere eXtreme Scale grid**
  - WXs components are hosted within integration server (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 integration node (broker)**
  - Starts with multiple integration servers but easy to extend to multiple nodes.

- **Advanced configuration available**
  - Integration server properties and Policy Profiles for more sophisticated topologies

- **IIB developer has simple artefacts for working with the global cache**
  - Unaware of the underlying technology (WXS) or topology
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);
    }
}
IIB Programming model – Data Eviction

- Set the “time to live” for data in the embedded global cache.

```java
MbGlobalMap evictMap = MbGlobalMap.getGlobalMap("evicto", new MbGlobalMapSessionPolicy(30));
evictMap.put("key", "val");
```

- Time to live is wrapped inside the MbGlobalMapSessionPolicy object.
- Specify a value in seconds. The default value is 0, which means data never gets automatically removed.
- Data is evicted when the “time to live” period has elapsed, after the data is last updated.
- The time to live is an override, which applies to data put or updated in a given map, within a given instance of the message flow. The underlying map, and data already present in that map, are unaffected.
In 8.0.0.x, only Java primitive types (bytes, numbers) and strings could be stored in the Global Cache.

In IIBv9, the Global Cache classloading mechanism was updated to allow users to use any existing Java classes with the Global Cache, as long as they implement either the Serializable or Externalizable interfaces.

The Java classes must be placed in the broker or integration server level shared-classes directories, and made available to all brokers participating in the cache.

Message flows can now share data with external applications when accessing external grids or XC10 appliances using existing Java classes.

The Global Cache can now be combined with JAXB functionality. Users can convert message data into a Java object using JAXB, and then store that Java object in the Global Cache.
Topologies - Introduction

- Broker-wide topology specified using a cache policy
  - Default provides single broker embedded cache
  - 1 catalog server and up to 4 container servers

- The initial value is Disabled
  - No cache components run in any integration servers.
  - Switch to Default and restart in order to enable cache function

- Broker will generate sensible defaults for port range and listener host name
  - Can choose a convenient port range for use by cache components in given broker
  - Can specify listener host name: broker cache components host name for binding
The Default Cache Topology

- Shows the cache components hosted in a 6-EG broker, using the default policy.
Topologies Policy File

- Use a cache policy file to define a multi-broker grid
  - Sample policy files are included in the product install

- Policy file tells broker how to participate in Global cache
  - Specify the policy file as the 'policy' property value on all brokers that are to participate

- Example policy file: 2 broker global cache, single catalog server

```xml
<?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>
```
Topologies – policy “none”

- None Policy
  - Switches off the broker level policy
  - Configure each integration server individually

- Example screenshot shows the integration server-level properties
  - Useful for fixing specific cache roles with specific integration servers.
  - You may wish to have dedicated catalog server integration servers.
  - Tip – start with “Default” or policy file, then switch to “None” and tweak the settings.
Administrative Tools

- **Full Resource statistics and Activity log**
  - Understand the state of the Cache and Cache Interactions

<table>
<thead>
<tr>
<th>Message</th>
<th>Timestamp</th>
<th>RM</th>
<th>MSGFLOW</th>
<th>Message Summary</th>
<th>Node</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>I 81F11504I</td>
<td>27-Jun-2012 13:25:5...</td>
<td>MF_StoreCache</td>
<td>Waiting for data from input node 'MQ Input'.</td>
<td>MQ Input</td>
<td>INPUT</td>
<td></td>
</tr>
<tr>
<td>I 81F11504I</td>
<td>27-Jun-2012 13:29:3...</td>
<td>MF_StoreCache</td>
<td>Received data from input node 'MQ Input'.</td>
<td>MQ Input</td>
<td>INPUT</td>
<td></td>
</tr>
<tr>
<td>I 81F11109I</td>
<td>27-Jun-2012 13:29:3...</td>
<td>GlobalCache</td>
<td>Connected to cache 'WMB'</td>
<td>Java Compute1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I 81F11107I</td>
<td>27-Jun-2012 13:29:3...</td>
<td>GlobalCache</td>
<td>Checked whether key exists in map 'SYSTEM.BROKER.DEFAULTMAP'</td>
<td>Java Compute1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I 81F11107I</td>
<td>27-Jun-2012 13:29:3...</td>
<td>GlobalCache</td>
<td>Put data into map 'SYSTEM.BROKER.DEFAULTMAP'</td>
<td>Java Compute1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I 81F11107I</td>
<td>27-Jun-2012 13:29:3...</td>
<td>GlobalCache</td>
<td>Checked whether key exists in map 'SYSTEM.BROKER.DEFAULTMAP'</td>
<td>Java Compute1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I 81F11107I</td>
<td>27-Jun-2012 13:29:3...</td>
<td>GlobalCache</td>
<td>Put data into map 'SYSTEM.BROKER.DEFAULTMAP'</td>
<td>Java Compute1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I 81F11107I</td>
<td>27-Jun-2012 13:29:3...</td>
<td>GlobalCache</td>
<td>Committed a local transaction.</td>
<td>MQ Input</td>
<td>INPUT</td>
<td></td>
</tr>
<tr>
<td>I 81F11107I</td>
<td>27-Jun-2012 13:29:4...</td>
<td>MF_StoreCache</td>
<td>Waiting for data from input node 'MQ Input'.</td>
<td>MQ Input</td>
<td>INPUT</td>
<td></td>
</tr>
</tbody>
</table>

**mqsicacheadmin command**

- Provide advanced information about the underlying WXS grid
- Validate that all specified brokers are participating in a multi-broker grid
- Check that the data is distributed evenly in underlying cache elements
- Use with the "-c showMapSizes" option to show the size of embedded cache
- Use with the "-c clearGrid -m <mapname>" option to clear data from cache
Multi-instance broker support

- In 8.0.0.x, Global Cache components – either catalog or container servers - cannot be hosted within a multi-instance broker.

- In 9.0.0.0, container servers can now be hosted within a multi-instance broker. Multiple listener hosts can be provided in the Global Cache configuration to allow container servers to correctly fail over as part of broker fail over.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<cachePolicy xmlns="http://www.ibm.com/xmlns/prod/websphere/messagebroker/globalcache/policy-1.0">
  <broker name="CATBRK1" listenerHost="host1.ibm.com">
    <catalogs>1</catalogs>
    <portRange>
      <startPort>3000</startPort>
      <endPort>3019</endPort>
    </portRange>
  </broker>
  <broker name="CATBRK2" listenerHost="host2.ibm.com">
    <catalogs>1</catalogs>
    <portRange>
      <startPort>3000</startPort>
      <endPort>3019</endPort>
    </portRange>
  </broker>
  <broker name="MIBRK">
    <listenerHost>host1.ibm.com</listenerHost>
    <listenerHost>host2.ibm.com</listenerHost>
    <catalogs>0</catalogs>
    <portRange>
      <startPort>3020</startPort>
      <endPort>3039</endPort>
    </portRange>
  </broker>
</cachePolicy>
```
Agenda

- Embedded “Global Cache”
- Connecting to external WebSphere eXtreme Scale grids.
- Best practices & and tips
Connectivity to external WebSphere eXtreme Scale grids

- In addition to the embedded global cache, introduced in v8.0.0.1, you can now also work with external WXS grids.
- This includes XC10 appliances.
- Connections are configured using the WXSServer configurable service.
- Connect to multiple external grids, and the embedded global cache at the same time.

- WXSv8.5 and 8.6 CORBA only (no XIO yet).
Configurable service for external grids

- Specify the catalog endpoints for your external grid, and the grid name.
- Optionally, if the grid requires user ID / pwd authorization, create a security identity and refer to it in the configurable service.
- For this example, the security identity is created by:
  `mqsisetdbparms <broker> -n wxs::xc10id -u <userId> -p <pwd>`
- Optionally point to a client override file.
Programming model for external grids

- New signature added for MbGlobalMap.getGlobalMap(String mapName, String configurableService).

- The resulting MbGlobalMap object then behaves exactly the same as if you were working with the embedded global cache. All connectivity and session management is handled by the broker.

```java
MbGlobalMap xc10Map = MbGlobalMap.getGlobalMap("Test.LUT","xc10Appliance");
xc10Map.put("key", "value");
```

- Interactions with external grids are logged in Activity Log and Resource Statistics in the same way as for the embedded global cache.
Override files for external grids

- For advanced use-cases, provide a WebSphere eXtreme Scale client configuration file, to override behaviours of the remote grid.

- The override file is typically a copy of the server’s objectgrid.xml configuration file, with some properties altered.
  - A common use would be to modify the following line in the file:
    ```xml
    <backingMap name=".*\LUT" template="true" readOnly="false"  lockStrategy="NONE"
    ttlEvictorType="LAST_UPDATE_TIME" timeToLive="3600" copyMode="COPY_TO_BYTES" />
    ```
  - And set the `numberOfBuckets` to 0.
    ```xml
    <backingMap name=".*\LUT" template="true" readOnly="false"  lockStrategy="NONE"
    ttlEvictorType="LAST_UPDATE_TIME" timeToLive="3600" copyMode="COPY_TO_BYTES" numberOfBuckets="0" />
    ```
  - This prevents the client inside the broker from keeping a "near cache" copy of any data, and removes the risk of having stale data.

- The `overrideObjectGridFile` property in the configurable service, if set, must be the fully qualified path to the file.
SSL for external grids

- Transport Layer Security for external grids is built on existing concepts within WMB. Follow the same steps for PKI setup as for SSL elsewhere in the broker.

- New options at the Execution Group level specify whether that EG should use SSL-Supported for client connections to external WXS grids.

- Optionally specify the protocol to use, and a key alias, if more than one trusted key exists in the keystore.

- If clientsDefaultToSSL is selected, all connections from this EG will attempt to use Transport Layer Security.

- With this box ticked, connections can only be made from EGs that do not host catalog or container servers.
Agenda

- Embedded “Global Cache”
- Connecting to external WebSphere eXtreme Scale grids.
- Best practices & and tips
Some popular advanced embedded topologies –
1: brokers can start in isolation
2: Protected catalogs in existing brokers – fixed roles

HOST 1
Broker 1 – using None policy

HOST 2
Broker 2 – using None policy

CLIENT ONLY

CATALOG + CONTAINER (+CLIENT)
NO FLOWS

CONTAINER (+CLIENT)

CLIENT ONLY

CATALOG + CONTAINER (+CLIENT)
NO FLOWS

CONTAINER (+CLIENT)
3: Protected catalogs in new brokers

HOST 1

Broker 1 – using Policy file

- CLIENT ONLY
- CONTAINER (+CLIENT)
- CLIENT ONLY
- CONTAINER (+CLIENT)

Broker 3 – using Policy file

- CATALOG + CONTAINER (+CLIENT)
- NO FLOWS
- CATALOG + CONTAINER (+CLIENT)
- NO FLOWS

HOST 2

Broker 2 – using Policy file

- CONTAINER (+CLIENT)
- CLIENT ONLY
- CONTAINER (+CLIENT)
- CLIENT ONLY

Broker 4 – using Policy file

- CATALOG + CONTAINER (+CLIENT)
- NO FLOWS
- CATALOG + CONTAINER (+CLIENT)
- NO FLOWS
Some operational best practices

- If possible, avoid having catalogs in execution groups where:
  - The JVM is frequently very busy with other processing
  - The EG is heavily used, and prone to “hang”
  - Network outages are expected between the catalogs
  - This is to avoid “network partition” conditions in the underlying WXS grid. See the next slide.

- Avoid stopping multiple EGs across multiple brokers at the same time. Otherwise, there’s a possibility you are destroying the primary and replica for some data at the same time.

- When performing maintenance activities on brokers where a multi-broker cache is used, try to stop or start only one broker at a time.
The network partition condition

- This scenario happens when the catalog servers are unable to communicate with each other, but remain running.

- In this case, the catalog servers will attempt to rebalance the cache and keep running using their own view of the available catalogs and containers.

- When connectivity is re-established, the catalog servers have different states for the same grid, which they cannot reconcile.

- One or more catalog servers (and possibly container servers) are now placed into a "stopped" state, and the hosting execution groups need to be restarted in order for those cache server components to rejoin the cache.

- Message flows should continue to work, communicating with the remaining cache components, without any data loss.

- But the resilience of the cache to other failures is reduced until the relevant execution groups are restarted.

- Diagnostics added in APAR IT05000, see later slide.
Troubleshooting tips – 1

- Syslog is your friend!
  - Always look for BIP7160/BIP7161 in the syslog for details of catalog/containers failing to start.
  - BIP7190 when components are retrying to start.
  - BIP7162/BIP7163 when they successfully start.
  - BIP7155 when an EG can access the cache, BIP7156 when it can't.

- If your cache has not started, or some of it has not started, or EGs cannot connect to the cache, then look at the chronology of these events in the syslog.

- You should be able to see the sequence of EGs starting, their cache components starting (or not), and the EGs connecting to the cache (or not!).
Troubleshooting tips – 2

- Use mqsicacheadmin to interrogate how the cache is formed, and what’s in it.
  - mqsicacheadmin <brokername> -c showPlacement and mqsicacheadmin <brokername> -c listHosts commands are particularly useful. A healthy grid will list all the containers that you think should be there – with primary shards and synchronous replica shards spread across them.
  - mqsicacheadmin <brokername> -c showMapSizes –m mapName to verify what’s in the grid.

- Is this a multi-broker grid? If so – is the policy correctly defined, and have you started at least 2 catalogs up front?

- If you’ve decided to switch policy to “none” and manually configure EGs – then double-check the EG-level properties.
  mqsireportproperties <brokername> e <EG> -o ComIbmCacheManager –r (for all EGs.)
Troubleshooting tips – 3

- If your problem is more around data in the cache (or message flow interactions with the cache), rather than an operational issue:
  - Use activity log to see the history of cache interactions from a particular message flow. For example, checking that certain puts or updates actually happened.
  - Use resource statistics to see the number of cache interactions, including any with external grids.

- Does your Java code introduce any “race conditions”, where different flows (or instances) can attempt to put or update the same key at the same time?

- Are you running mqsicacheadmin <brokername> -c clearGrid at the same time? (either from the command line or custom Java code invoking the command).

- Java OOM errors appearing?
  - Check that you are not continually putting data into the cache without removing it, or without specifying a Time To Live.
  - If not – then it is possible the max JVM heap size is not big enough.
Important recent APARs

- **IT05000** - MISSING DIAGNOSTICS IN GLOBAL CACHE NETWORK PARTITION CASE
  - Currently targeted for 8.0.0.6 / 9.0.0.4

- **IT01705** - GLOBAL CACHE CONTAINER STARTUP SOMETIMES FAILS
  - Currently targeted for 8.0.0.6 / 9.0.0.4

- **IT02381** - GLOBAL CACHE ENTRIES LOST ON BROKER RESTART
  - Included in 8.0.0.5 / 9.0.0.3

- **IC98156** - GLOBAL CACHE FLOW CONNECTIONS TIME OUT TOO SOON
  - Included in 8.0.0.4 / 9.0.0.2 and above

Thank you! – Any questions?

- Feedback: [ibm.biz/wug15feedback](ibm.biz/wug15feedback)