Understanding re-allocation of messages in IBM MQ Clusters

AdrianDick
Published on 27/09/2018

Many users of IBM MQ utilise the clustering functionality to distribute messages across multiple instances of given queues, either to share the workload or as a failover mechanism. Generally, the selection of destination is made as an application puts a message into the system. However, if a message is in-flight and unable to be delivered to the selected cluster queue manager the clustering capability can automatically re-allocate messages to an alternate queue manager.

When does re-allocation occur?

Re-allocation of messages only occurs when a cluster-sender channel stops, e.g. unable to communicate with remote queue manager. And again every time a channel retries its connection and fails (so based on the channel's retry setting).
When reallocating, the clustering function examines all messages on the cluster transmission queue targeted for the stopping channel and re-drives the internal workload balancing algorithms to route each message to an alternative available destination.

When won't messages be re-allocated?

If carefully monitoring the cluster transmission queues, there are times where messages remain targeted for the original queue manager selection without any apparent re-allocation. This failure to select an alternate queue manager could be for one of two reasons:

- Not eligible for re-allocation
- Still most favourable destination

Messages not eligible for re-allocation

There are some messages which are flagged as not eligible for re-allocation. If the cluster transmission queue were to be browsed these can be identified as messages with the Message Descriptor (MD) accounting token field set to all zeroes.

Why is it not eligible?

If you see any message not eligible for re-allocation, it will be because of one of the following:

- Bind behaviour of putting application
  - Using bind-on-open – all message must go to the same queue instance
  - Using bind-on-group and actively using managed groups (MQPMO_LOGICAL_ORDER) – indicates all message in current group must go to the same queue instance

- Putting application has explicitly named the target queue manager, therefore bypassing the cluster workload management algorithms
  - This includes system messages to system queues on the target queue manager.

Still most favourable destination

Having confirmed the message is available for re-allocation, it may be that the workload balancing algorithm considers the original destination the most favourable target. You can detect this by seeing that the PutDate/PutTime gets updated as the stopped channel triggers further attempts at re-allocation.

Why is it still most favoured?

Now we're starting to delve into the mechanics of how workload balancing makes its decisions…

There is only one choice!

If issuing DISPLAY QCLUSTER shows just one instance of the queue, then clearly MQ can only send it to that queue manager, so reallocation can not select a different queue manager.

Need to understand precedence of cluster workload attributes

Having got this far the chances are that your system has non-default values for one or more of the cluster workload attributes on queues and channels. And it isn't always obvious to the uninitiated how these interact with each other.
So, let's talk our way through them. As we work down this list, eliminate all but the equal highest before considering the next attribute. To do this you'll need to inspect the cluster queue manager and cluster queue definitions cached on the sending queue manager. After elimination, if only a single choice remains, that will be why the message hasn't been directed elsewhere.

1. Eliminate all but the highest ranked channels, based on each of the channel CLWLRANK values
2. Eliminate all but the highest ranked queue, based on each of the queue CLWLRANK values
3. Eliminate all PUT(DISABLED) queues – If that's all the remaining instances the operation will fail, and the original message is not re-allocated
4. Eliminate queue managers suspended from the cluster – Unless they're the only option remaining!
5. Consider channel state and eliminate all but the most favoured:
   - Running channels most favoured
   - But stopping channels are favoured over those that are retrying which in turn are preferred over a stopped channel
6. NETPRTY on channels, if more than one channel remains for a given queue manager
7. Eliminate all but the highest priority channels, based on each of the channel CLWLPRTY values
8. Eliminate all but the highest priority queues, based on each of the queue CLWLPRTY values

A common issue here is misunderstanding the difference between CLWLRANK and CLWLPRTY, as both work in a very similar manner. The significant difference is whether you want them to be considered before or after the state of the channel. Before (using CLWLRANK) means messages will not be redirected in the event of a channel simply stopping.

Further reading

IBM Knowledge Center – MQOPEN and clusters
IBM Knowledge Center – Workload balancing in clusters