Should I mix persistent and non-persistent messages on a queue?

ColinPaice
Published on 11/10/2017

This question came up and caused a lot of discussion, as the answer is not a simple yes or no. If you are interested in making your shared queue queue manager recover a little faster – there is something you can do.

Most "simple" applications use persistent or non-persistent, but not both. (If they use both, it would be worth investigating, and documenting why mixed persistence is used). Some sophisticated applications may have persistent message as input, but use non persistent messages to control internal workflow, and reply with a persistent message.

I would expect a request queue to have persistent or non-persistent, but not both. A reply queue may have a mixture of persistence if it is used by multiple applications. One application uses non persistent replies, another application uses persistent replies.

A sophisticated application, such as the CICS bridge, has data from potentially many applications. The name of the transaction to be run is specified in the message data itself. In this case the queue can have a mixture of persistent and non-persistent messages.

What is the impact of having a queue with mixed persistence?

The elapsed and CPU time to put or get a persistent and non-persistent message are very similar. The big difference is the "commit". Non persistent messages are usually processed out of syncpoint. In this non persistent case, there is no logging to disk. Persistent messages are usually processed within syncpoint – and there is a log write to disk when the transaction is committed. If you have a single server then the messages may be processed – slow (for persistent) quick (non-persistent), slow, slow, quick and so on. Multiple servers may help, but you could still get all of your servers processing persistent messages at the same time, and so you get a 'blip' in throughput.

With shared queue on z/OS structures with persistent messages need to be backed up. The persistent messages can then be recovered from the logs in the event of a failure, but this can take time. You can specify a structure as not recoverable. It can only hold non persistent messages, and you do not need to backup the CF structure. In the event of a failure, a non-recoverable structure can be recreated instantly as there is no need to read the log to recover any messages (but all the non-persistent messages will be lost of course). You may want to consider structures just for non-persistent messages, and structures just for persistent messages.