My XMITQ Queues are filling up – what should I monitor?

ColinPaice
Published on 14/08/2017 / Updated on 24/08/2017

From a business perspective an application may see a slow down, and one cause is XMITQs are filling up. How do you detect this – and what can you pro-actively do?

You can monitor two types of data

MQ events

MQ can put a messages to an events queue when the message depth is over a certain value, or a message was on a queue for a long time.

1. Advantages
   1. Low overhead – not additional costs until the event is triggered
   2. You know when the incident happens
2. Disadvantage
   1. You do not see any trending information – like over the last month the average queue depth increases
   2. Some events you have to reset once it has fired. For example once you get queue high event, you need a queue low event to reset the trigger – this avoids getting events as the current depth of the queue hovers around the queue depth high value.

Regularly display information about the queue

1. Advantages
   1. You can see trends
2. Disadvantages
   1. This requires regular checks which uses CPU.
   2. The more queues you check the higher the costs
   3. The more frequently you check the higher the costs – but you get a more accurate picture
   4. You can miss problems. For example at 0100 the queue depth is 0. at 0101 the queue depth is 10000 at 0104 the queue depth is zero. At 0105 the monitoring sees the queue depth is zero and reports all is well!

What MQ events are useful?

1. Queue depth high tells you when a queue gets to a certain depth. If the normal depth of the queue is 5 then set the queue high depth to be at least 2 * batch size.
2. Service interval high. When a message is read and the time between the put and the get is longer than the service interval. Note: this is detection once the message has been got. If the message was stuck on a queue for 4 days because the channel was down, you get the event when the message is got. This is like the burglar alarm going off when the intruders leave the building.

**What attributes are interesting**

1. Display current depth
2. Display status and check the age of the oldest message (MSGAGE) for a cluster transmission queue messages for a destination which is down will be old – so this may not reliably tell you if there is a problem
3. DIS CHS and look at XQTIME
4. DIS CHS and make sure the channel is STATUS(RUNNING)

**What can you do?**

At a customer, when an MQ event occurred, the systems monitoring invoked a bash shell script passing in the queue manager name ($1) and queue name($2).

The script issued

```bash
# The logfile is /tmp/MQyymmdd.log
logFile="/tmp/MQ"$(date +%y%m%d)".log"

echo "DIS CHS(*) NETTIME BATCHSZ XBATCHSZ BYTSSENT MSGS STATUS XQTIME where(xmitq,eq,$2)"| runmqsc $1 >> $logFile
sleep 1

echo "DIS CHS(*) NETTIME BATCHSZ XBATCHSZ BYTSSENT MSGS STATUS XQTIME where(xmitq,eq,$2)"| runmqsc $1 >> $logFile
```

This issues the command with a 1 second gap between the display commands. This allows you to see

1. If the channel(s) was running
2. The rate of messages processed
3. if the queue depth is changing. This may show messages are arriving faster than the channel is processing them
4. How many bytes were sent – to allow you to calculate the bytes send per second, the channel data rate. Compare this with the “normal” value you collected earlier.
5. The nettime – to show if there is a problem with the network or at the remote end

**Tags** monitor, XMITQ

by ColinPaice