64-bit Support for IBM MQ Queues

Communications Subcommittee

Jamie Farmer

2024 TPF Users Group Conference
May 05-08, New Orleans, LA

IBM Z

Agenda

- Introduction
- Using 64-bit IBM MQ Memory
- Persistent Messages for IBM MQ Queues
  - Checkpointing IBM MQ Queues
  - Sweeping IBM MQ Queues
- 64-bit IBM MQ Diagnostics
- Summary
- What’s Next

Introduction

Pain Points

- Today, messages on memory queues are stored in system work blocks (SWBs), a critical system resource that resides below the 2 GB bar
  - With limited 31-bit memory, more frequent I/O operations might result as the IBM MQ sweeper needs to free up SWB memory
  - Expanding the number of SWBs might not be possible or might mean sacrificing other system resources
  - Limits growth of applications and IBM MQ

Introducing 64-bit for IBM MQ Queues

- 64-bit IBM MQ queues will store their messages in a **new memory type dedicated to 64-bit IBM MQ** that resides above the 2 GB bar

- Greatly increases the scalability of IBM MQ by taking full advantage of the memory above the 2G bar

- Improved performance and resiliency of IBM MQ

Using 64-bit IBM MQ Memory

64-Bit IBM MQ Memory

- A new CTKA variable created to define the amount of 64-bit IBM MQ memory that you want z/TPF to allocate
  - Maximum value up to 9 TB – over 4000 times that of 31-bit IBM MQ!
  - 64-bit IBM MQ memory will be allocated during an IPL

- Not possible for other activities on the system to deprive 64-bit queues of memory and vice versa

- Storage is dispensed as 4 KB 64-bit IBM MQ memory entries referred to as **MQM entries**

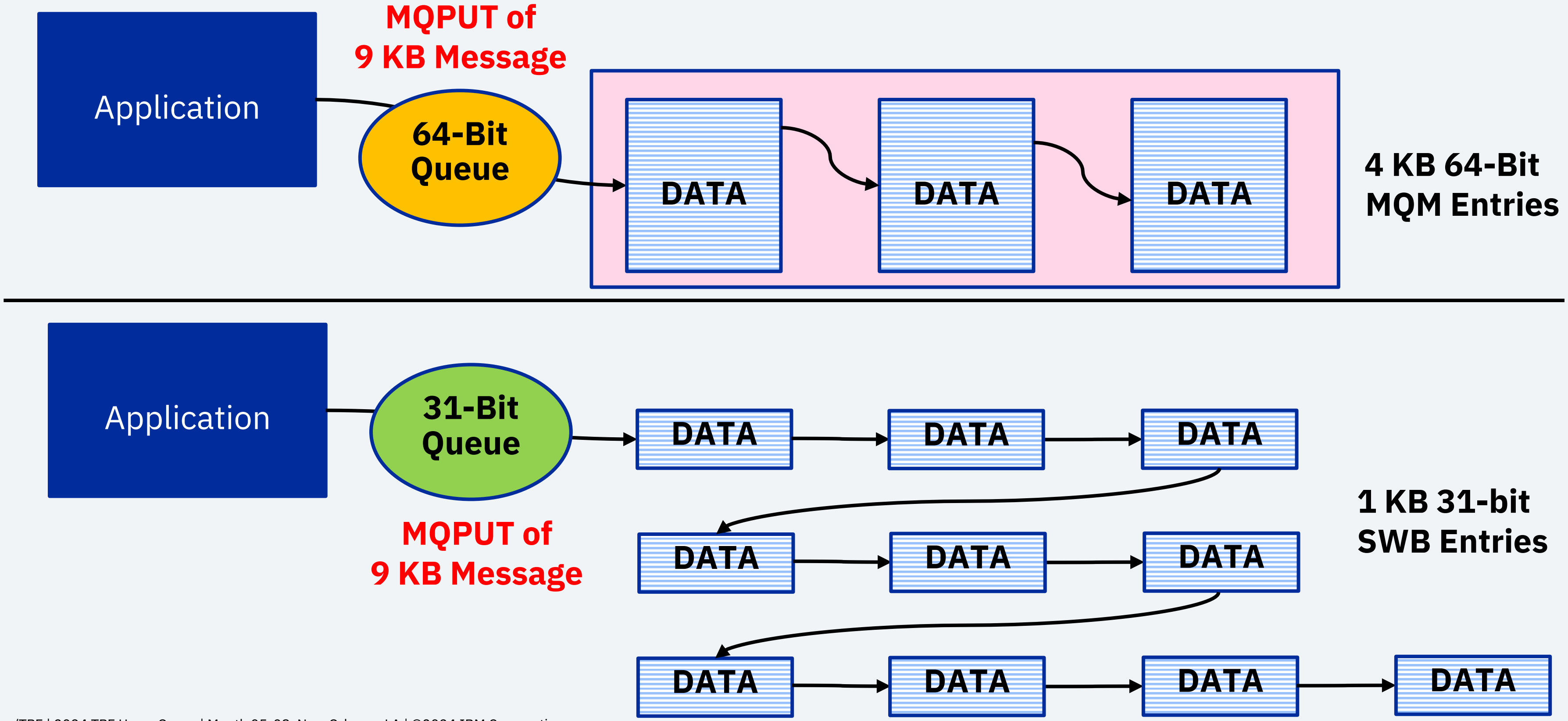
Create or Migrate to a 64-Bit Queue

- Define a new queue
  - **`ZMQSC DEFINE QL-’<Queue name>’ 64BIT-YES`**
- Migrate an existing 31-bit queue
  - **`ZMQSC ALTER QL-’<Queue name>’ 64BIT-YES`**

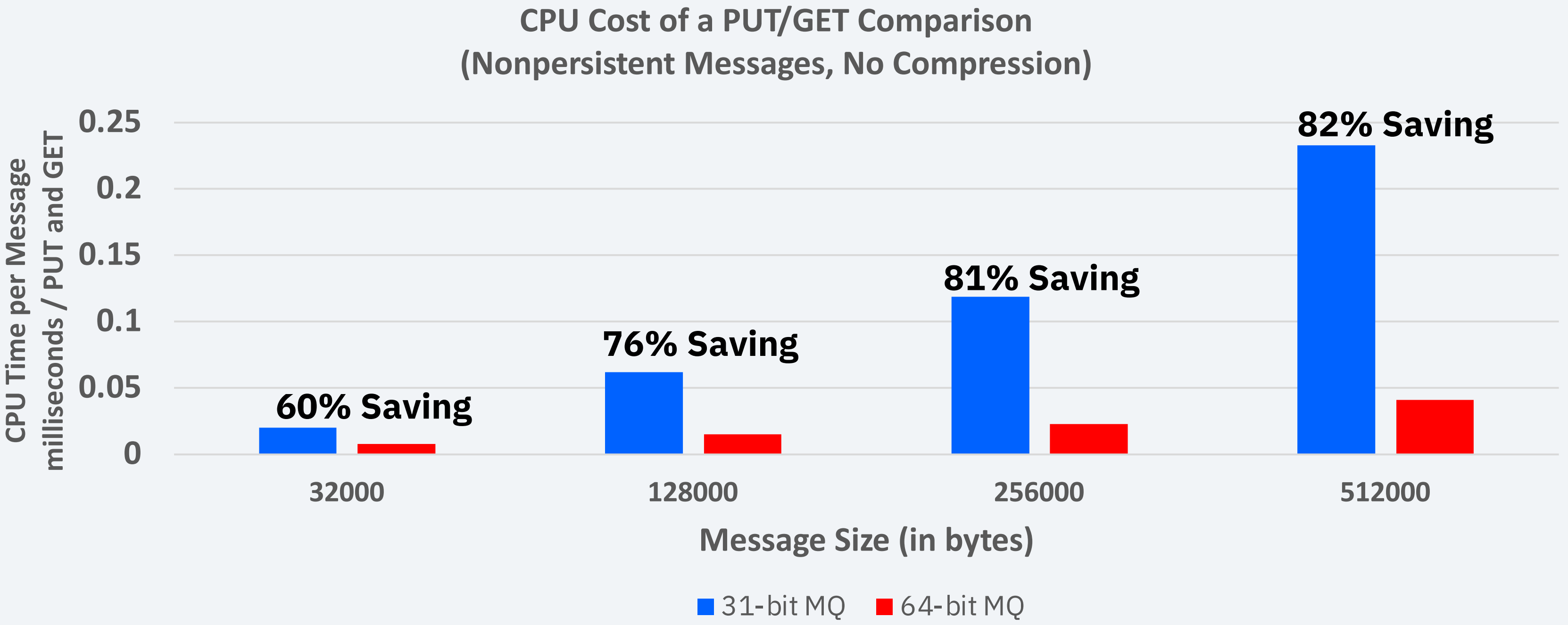
- Seamless migration to (and if necessary, fallback from) 64-bit without application changes
  - z/TPF IBM MQ applications do not have knowledge of how messages are being stored.

64-Bit IBM MQ Memory vs 31-Bit IBM MQ Memory (SWBs)

Reduced processing for larger messages



Performance of MQGET and MQPUT for 64-bit MQ



For persistent messages seeing 16% - 33% reduction in CPU cost per MQPUT/MQGET

64-Bit Queue Considerations

- The primary use case for 64-bit queues is for high volume, FIFO queues.
  - For example, high volumes of data from z/TPF to remote queues like business events.

MQGETs must be sequential (FIFO)

- Browsing is currently not supported
- Searching by message ID is not supported
- Searching by correlation ID is not supported

A 31-bit queue that uses any of the previous actions cannot be migrated to a 64-bit queue.

Identifying Candidate Queues for Migration to 64-Bit

APAR PJ46881 (Nov 2022) adds a new display option to help identify queues that use functionality restricted on 64-bit

C = Search by correlation ID
M = Search by message ID
B = Browsing

```
User:      ZMQSC DISPLAY QL-* GETDIAG

System:    CSMP0097I 14.39.34 CPU-B SS-BSS  SSU-HPN  IS-01
           MQSC0278I 14.39.34 LOCAL QUEUE MQGET DIAGNOSTICS DISPLAY
                                                              SEARCH/    XMITQ  LAST
           Queue Name                                         BROWSE     GET    PROG
           -------------------------------------------------- ---------- ------ -----
           CalculatorQueue                                    NO         N/A
           CalculatorSyncReplyQueue                           NO         N/A
           AsyncCalculatorQueue                               NO         N/A
           MY.MEMQ.1                                          NO         N/A
           MY.MEMQ.2                                          YES  (CMB) N/A    ABCD
           MY.XMITQ.1                                         NO         NO
END OF DISPLAY
```

Benefits of using 64-Bit IBM MQ Queues

31-Bit Queues	64-Bit Queues
• Message storage below the 2 GB memory bar	• Message storage above the 2 GB memory bar
• Messages are written to 1 KB memory blocks (general purpose SWBs)	• Messages are written to 4 KB memory blocks (MQM) dedicated to IBM MQ only

Persistent Messages for 64-bit IBM MQ Queues

Persistence Messages for IBM MQ (31-Bit and 64-Bit)

- **Recovery Log**
  - Each MQPUT (along with its message data) and information for each MQGET are written to the recovery log as a high performing way to have every persistent message operation written to disk.
- **Checkpoint**
  - Required for persistent messages to keep the resident set of IBM MQ messages on queues to disk
  - Written every 5 seconds to avoid recovery log from filling
- **Sweeping**
  - Persistent and nonpersistent messages written to disk to free resources in low memory situations.

Checkpointing IBM MQ Queues

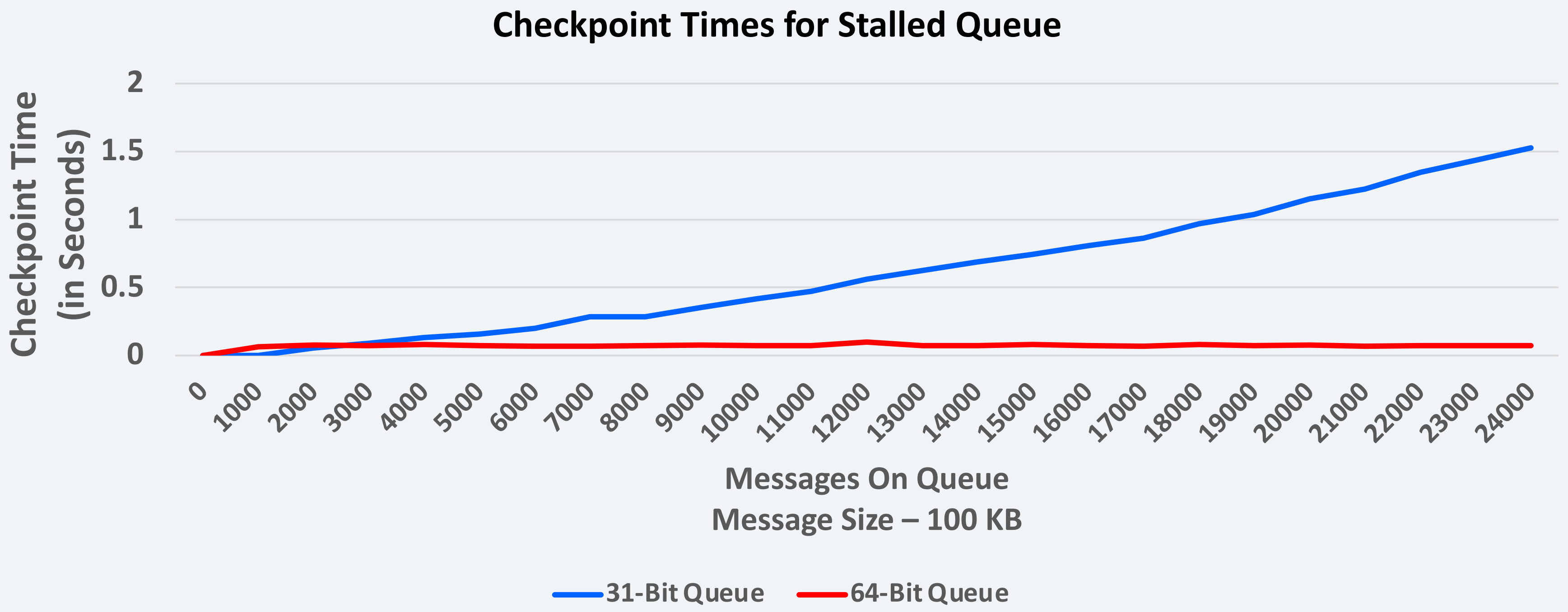
Checkpoint Summary for 31-Bit Queues

- For checkpoint, a complete copy or full replace of messages on 31-bit queues are written to fixed file records every 5 seconds.
  - This approach works because there is a limited number of messages that can be on 31-bit queues
    - Total number of SWBs defined, less than 2 gigabytes of memory.
- In stalled queue situations, I/O operations are significantly duplicated because the same messages are written to disk every 5 seconds.
- Each checkpoint ECB will do as many as 175 I/O operations in parallel

Checkpoint Summary for 64-Bit Queues

- 64-bit checkpoint **incrementally builds on the last checkpoint**, only additions and removals to persistent messages need to be recorded
  - Required as queues can have terabytes of messages on them
  - This means a **message will be written to pools one time at most**
    - FARF6 pools are supported and recommended
  - Messages written to long-term pools anchored off a fixed file record (QCCR) assigned to the queue
- Highly performant in a stalled queue case!
- Each checkpoint ECB will do as many as 250 I/O operations in parallel!
- Unlike 31-bit, queue lock is not held during I/O operations, which allows applications to put and get messages while checkpoint is running.

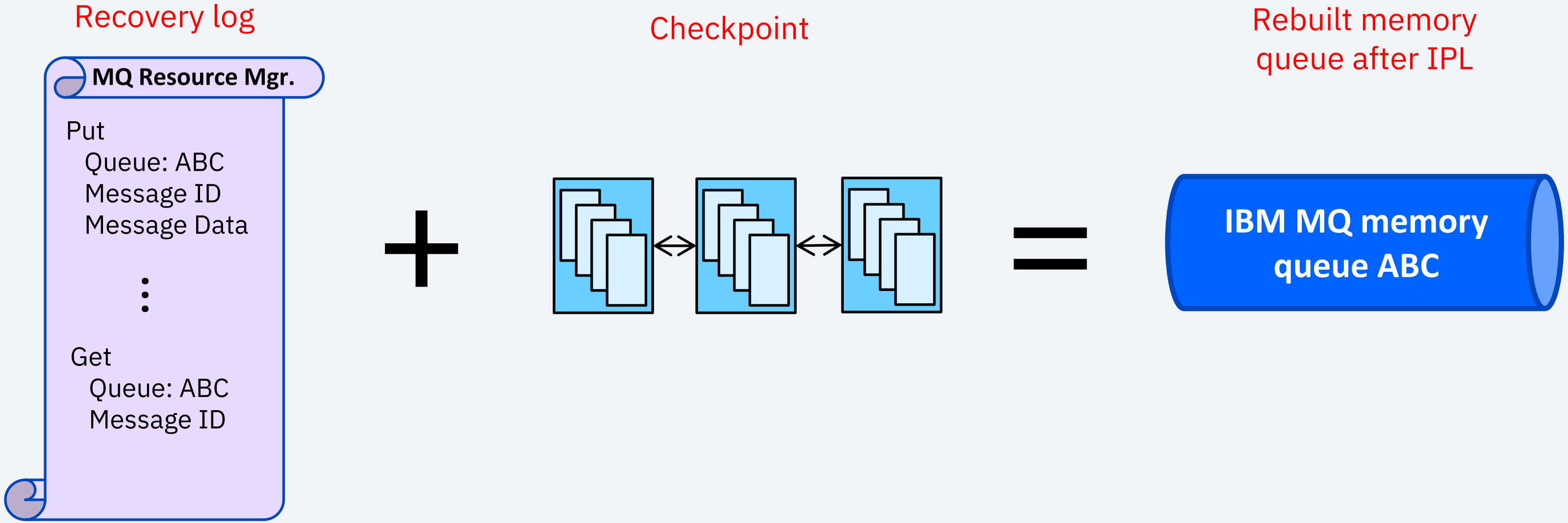
IBM MQ Checkpoint Performance (31-bit vs 64-bit)



1000 messages added each invocation of the checkpoint

Recovering Persistent Messages on a 64-Bit Queue

- The recovery log will be used by 64-bit queues similarly to 31-bit queues
- When the system restarts, the recovery log is merged with the checkpoint to make sure no persistent messages are lost during an IPL.
  - For 64-bit, entire queue does not need to be brought into memory



Benefits of using 64-Bit IBM MQ Queues

31-Bit Queues

- Message storage below the 2 GB memory bar
- Messages are written to 1 KB memory blocks (general purpose SWBs)
- **Application ECBs cannot MQGET or MQPUT messages while checkpoint is running**
- **Full replace checkpoint**

64-Bit Queues

- Message storage above the 2 GB memory bar
- Messages are written to 4 KB memory blocks (MQM) dedicated to IBM MQ only
- **Application ECBs can MQGET and MQPUT messages while checkpoint is running**
- **Incremental checkpoint**
  - **Less I/O in critical situations**
  - **Faster checkpoint completion with increased I/O parallelization**

Sweeping IBM MQ Queues

Goal of the IBM MQ Sweeper

- The IBM MQ sweeper is intended to free memory used by IBM MQ when the system is running low on that type of memory.
  - For 31-bit IBM MQ, the goal is to free SWBs from 31-bit queues by writing messages to disk.
  - For 64-bit IBM MQ, the goal is to free 64-bit IBM MQ memory (MQM) from 64-bit queues by writing messages to disk.
    - **Only nonpersistent messages need to be written to disk for 64-bit IBM MQ because persistent messages were already written to disk by checkpointing!**

31-Bit Sweeper Architecture

- When running low on SWBs, a **single** ECB is created to sweep messages for all queues.
  - Queues with the most messages are not always swept first

- Swept messages from queues are written to z/TPF collection support (z/TPFCS).
  - Even though those message might have already been written to the 31-bit checkpoint.

31-Bit Unsweep Architecture

- 31-bit unsweep processing is reactive
  - Unsweep processing to move messages back into memory does not occur until an application issues MQGET and the next message on queue is not in memory.
  - Unsweep processing and the I/O associated from z/TPF collection support (z/TPFCS) reads are done in the application ECB
  - Application ECBs must wait until the unsweep completes before satisfying the MQGET
    - TPFCS BLOBs of messages are read in

- Because the swept messages are detached from the checkpoint, persistent messages that are being unswept must be written to the recovery log (again!).

64-Bit Sweeper Architecture

- The 64-bit sweeper will create multiple ECBs to sweep the queues (as well as unsweeping of queues).
  - Number of ECBs limited to control IOB usage.
- The 64-bit sweeper targets the queues with the highest surplus of messages.
  - Doesn't necessarily mean queues with the most messages

    - Q1 : 10,000 msgs with a GET rate of 1000 msgs/sec

        - 10 seconds worth of messages

    - Q2 : 1,000 msgs with a GET rate of 50 msgs/sec

        - 20 seconds worth of messages

  - If targeting 10 seconds worth of messages in memory, then Q2 has more messages to sweep.

64-Bit Sweeper Architecture (Cont.)

- The 64-bit sweeper shares the same persistent mechanism as the 64-bit checkpoint
    - For persistent messages, the sweeping is simply pulling from memory
        - I/O is not required to sweep persistent messages!
    - For non-persistent messages, the sweeping is filing the messages
    - Entire queue does not need to be rebuilt in memory after an IPL
        - Can be done because checkpoint copy is also the swept copy of persistent messages

64-Bit Unsweep Architecture

- Unsweep processing for messages is integrated into the sweeper processing.
- System is analyzed to determine which queues can be swept vs unswept
  - Proactive unsweep: invoked as part of the sweeper to bring in messages before they are needed by the application
    - Multiple ECBs can be dedicated to unsweep
  - Reactive unsweep: invoked by application if proactive cannot keep up with application demand.
    - Should only be needed for poorly configured 64-bit IBM MQ memory

IBM MQ Unsweep Performance (31-Bit vs 64-Bit)

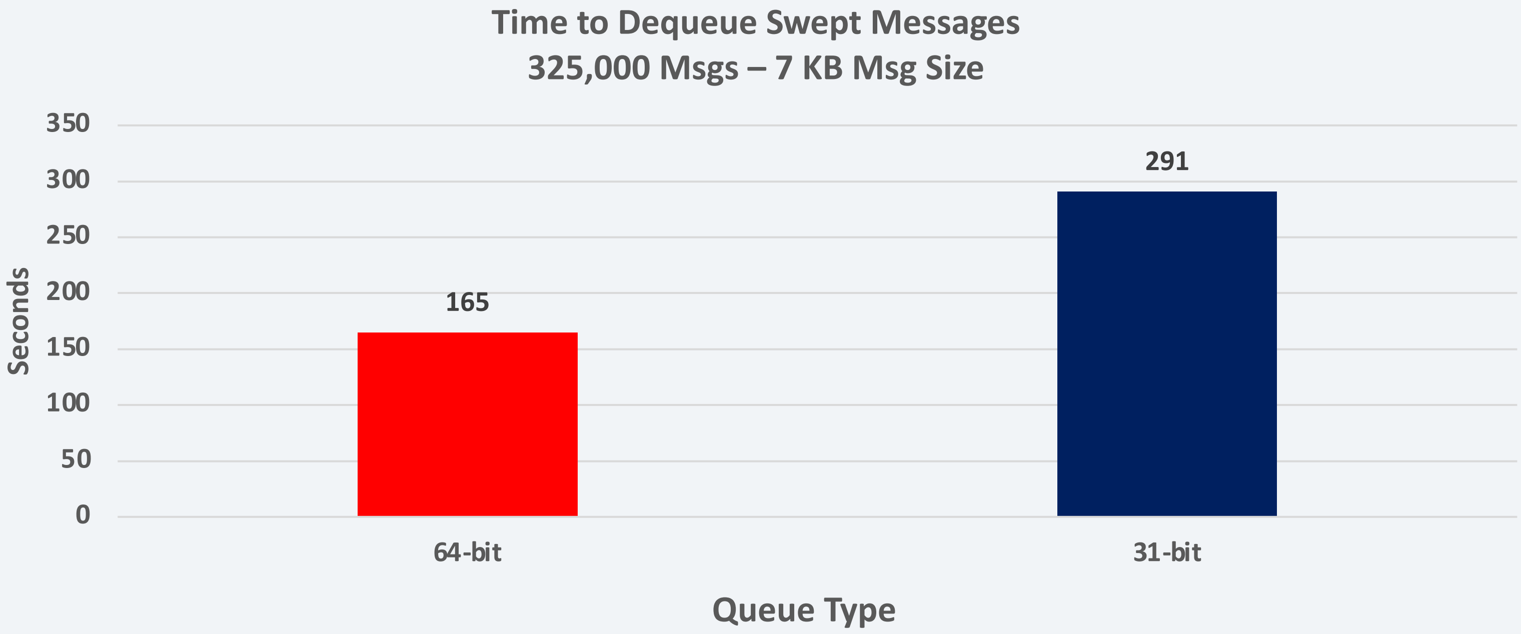
- 325,000 messages of 7 KB in size were added to both a 31-bit and 64-bit queue.
    - Approximately 300,000 of those messages were swept out to disk.

- 8 application ECBs each in a loop issuing an MQGET followed by a sleep of 4000 mics to simulate application logic processing that message.

- If the time to do an MQGET were 0, then each ECB could process 250 messages per second (2000 msgs / second total by all 8 ECBs) and the queue could be drained in 163 seconds.

IBM MQ Unsweep Performance (31-Bit vs 64-Bit)

- With 31-bit MQ, there were delays during many MQGET APIs reading messages from disk back into memory, so each MQGET took, on average, 394 microseconds to process
- With 64-bit MQ, proactive unsweeping the next message was always in memory when MQGET is issued so each MQGET took, on average, 6 microseconds to process



43% Reduction in Dequeue Time

Remote Recovery Performance (31-Bit vs 64-Bit)

- 325,000 messages of 7 KB in size were added to both a 31-bit and 64-bit queue.
    - Approximately 300,000 of those messages were swept out to disk.

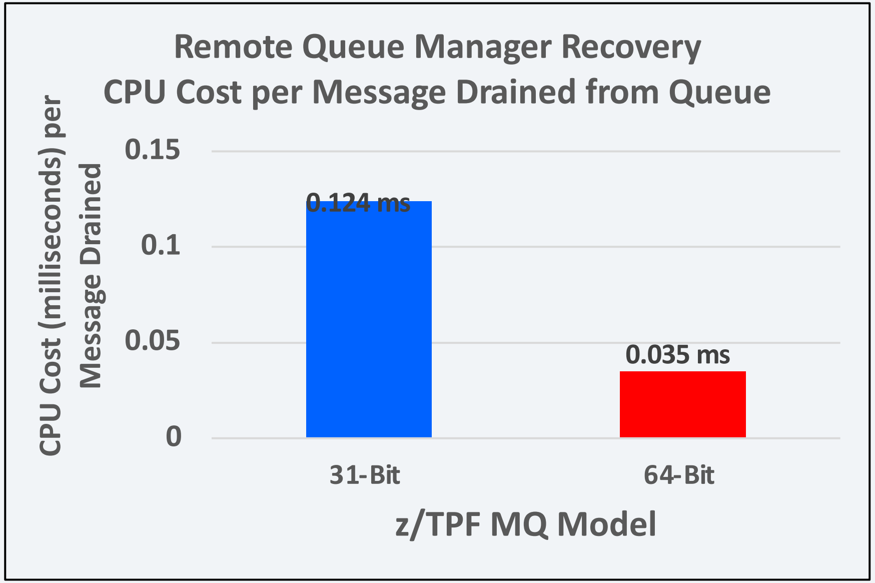
- 8 application ECBs each in a loop issuing an MQGET followed by a sleep of 4000 mics to simulate application logic processing that message. If time to do an MQGET was 0, messages would be removed from the queue at a rate of 2000/sec.

- Additional 8 application ECBs each in a loop issuing an MQPUT followed by a sleep of 8000 mics to simulate new work being added to the queue. If time to do an MQPUT was 0, messages would be added to the queue at a rate of 1000/sec.

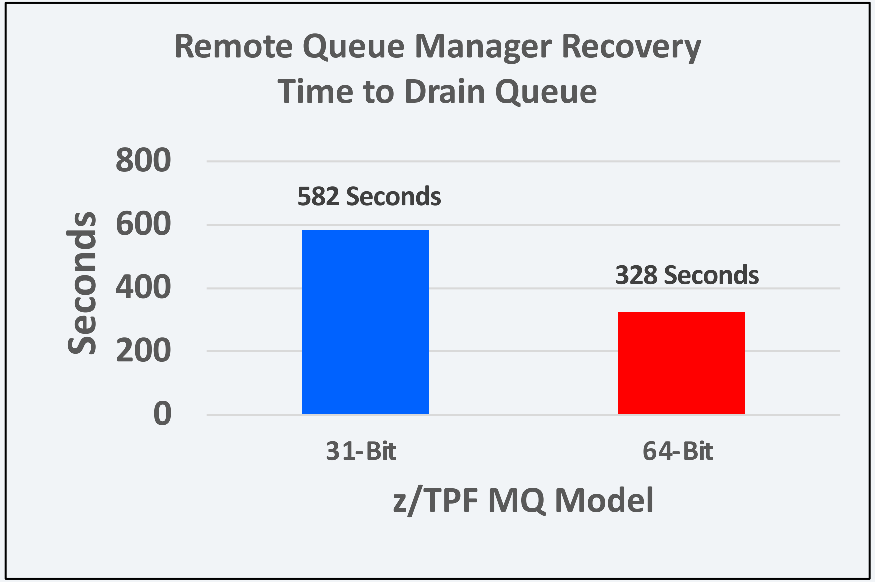
- If the time to do an MQGET / MQPUT were 0, then the queue could be drained in 325 seconds.

Remote Recovery Performance (31-Bit vs 64-Bit)

- Simulating a remote queue manager failure and transmission queue grows on the z/TPF system.
  - Queues grows to 325,000 7 KB messages and 300,000 are swept
  - Once the remote queue manager becomes active, measured the CPU cost per message dequeued and the time to drain the entire queue.



72% Reduction



44% Reduction

Benefits of Using 64-Bit IBM MQ Queues

31-Bit Queues

- Message storage below the 2 GB memory bar
- Messages are written to 1 KB memory blocks (general purpose SWBs)
- Application ECBs cannot MQGET or MQPUT messages while checkpoint is running
- Full replace checkpoint
- **Sweeper writes messages to z/TPFCS (separate from checkpoint)**

64-Bit Queues

- Message storage above the 2 GB memory bar
- Messages are written to 4 KB memory blocks (MQM) dedicated to IBM MQ only
- Applications ECBs can MQGET and MQPUT while checkpoint is running
- Incremental checkpoint
  - Less I/O in critical situations
  - Faster checkpoint completion with increased I/O parallelization
- **Sweeper leverages checkpoint copy of queues**
  - **No duplicated I/O**
  - **More efficient and resilient sweep / unsweep processing**
    - **No I/O required for sweep of persistent messages**

Sweeper Settings for 64-bit MQ

- New parameters have been created in keypoint A (CTKA) to control the rate of sweeping.
  - **MQMSWPL**: Percentage of in use 64-bit IBM MQ memory that must be in use before the sweeper will begin to sweep 64-bit queues.
    - Default value of 80%
  - **MQMSWPT**: Target percentage of in use 64-bit IBM MQ memory the sweeper will attempt to reach.
    - Default value of 60%

64-bit IBM MQ Diagnostics

New Diagnostic Fields in Local Queue Display

```
  MQSC0282I 21.51.24 LOCAL QUEUE DISPLAY:
  Queue Name          - Q64_TST
  Descr               - 64BIT QUEUE FOR TESTING
  QManagerName        - TPFQM

     ...

  64BIT               - YES
  MQM                 - YES

END OF DISPLAY+
```

New Description Parameter

YES when 64BIT set to YES

Set to YES once the queue has transitioned to using 64-bit MQM memory

New 64-Bit Statistics on Local Queue Stats Display

```
MQSC0285I 21.56.14 LOCAL QUEUE STATISTICS DISPLAY:  rcvry_driver_64bit_1
  Current Depth - 10
  Persistent Msgs - 45266580 (4041 megabytes)

    . . .

  Num of SWBs in use   - NONE
  Num of MQMs in use   - 10
  Num of pools in use - 441
  Aborted Sweep Count - 0 _
  Checkpoint High      - 0.065 seconds
  Checkpoint Last      - 0.063 seconds
  Chkpt in Progress    - NO
  Time Last 64Bit Swp                  - 0
  Time Last 64Bit Proactive UnSwp - 0
  Time Last 64Bit Reactive UnSwp  - 0
```

← Number of 64-Bit IBM MQ in use by queue

← Number of 4K Pool records in use by queue

} 64-bit IBM MQ Sweep Statistics

Queue Manager Display

```
MQSC0283I 15.56.30 QUEUE MANAGER DISPLAY
  QMNAME – TPFQM
  DESCR – TPF System Queue Manager
  DEADQ – DEAD.LETTER.QUEUE
  DLQ64 – DEAD.LETTER.QUEUE.64
     .  .  .

  Total MQM - 125437           In Use MQM – 1032
  HW In Use Count MQM – 2023 HW In Use Time MQM – 2024-02-15 17:05:01
  MQMSWPL(%) - 80          MQMSWPT (%) - 60
```

New description parameter for queue manager

New 64-bit dead letter queue

64-Bit IBM MQ Memory Statistics

64-bit Sweeper Parameters

Summary

64-bit Support for IBM Queues

- Delivered in December 2023 - PJ46819

- Apply the following additional APARs:
    - **PJ47238** (Mar 2024): Build failure due to incorrect migration considerations.
    - **PJ47256** (Mar 2024): Possible OPR-4 system error during restart

What’s Next?

Disclaimer

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

What's Next??

Effort is underway for follow-on 64-bit IBM MQ work

- Support for messages greater than 4 MB in size (up to 100 MB)
- Limited browse capabilities to browse messages in memory (that have not been swept)

Be a sponsor user

Sponsor users assist in design and implementation, and your feedback drives our development cycle.

Target personas

- Application Developers
- System Administrators
- Enterprise Architects

Interested? Contact

Jamie Farmer - jvfarmer@us.ibm.com
Cameron Doggett – camd.99@us.ibm.com
John Muller - jfmuller@us.ibm.com

Thank you