SHARE
EDUCATE • NETWORK • INFLUENCE

MQ Internals: What Really Happens Under the Covers of IBM MQ for z/OS

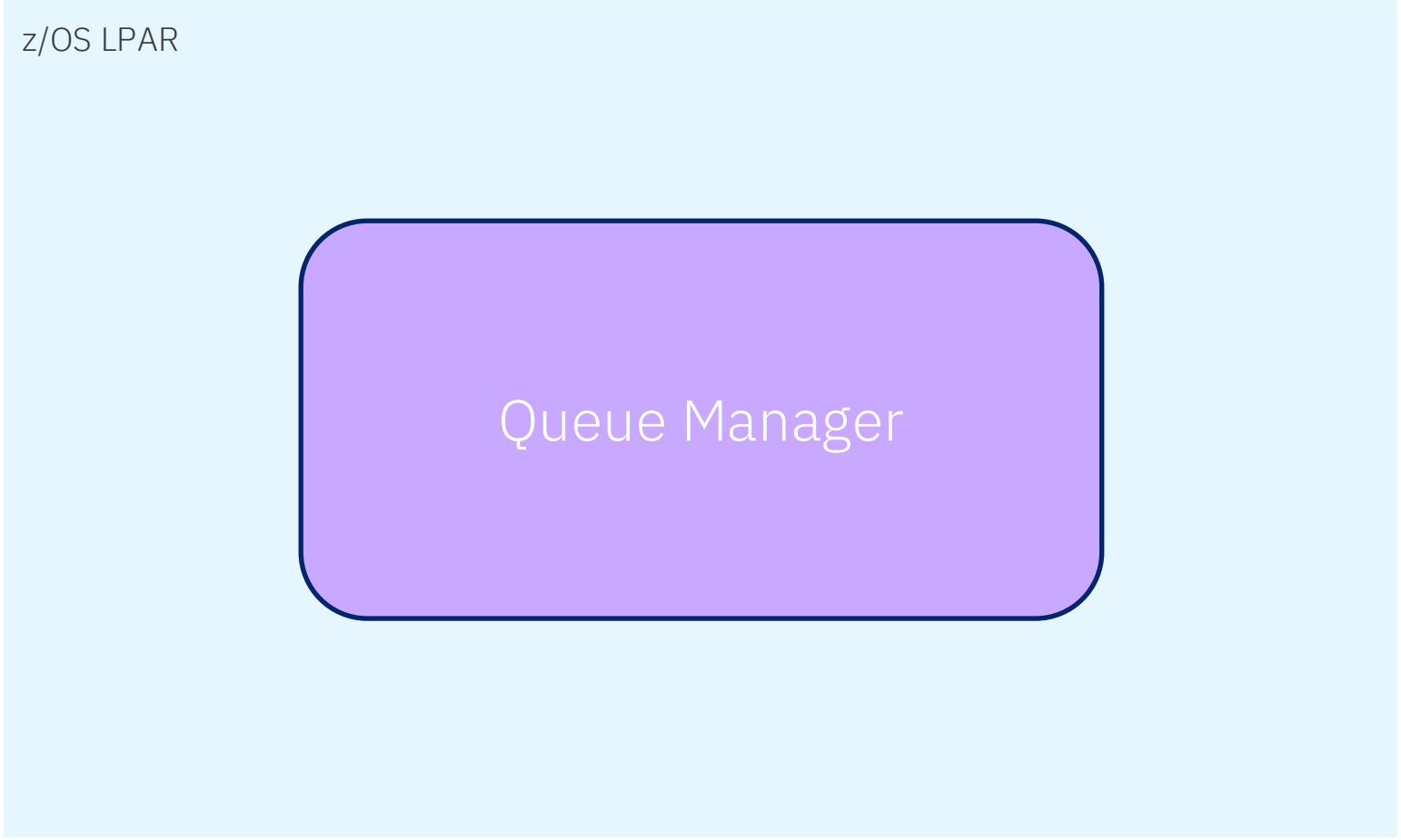
Toby Keegan
Software Engineer, IBM MQ for z/OS
toby.keegan@ibm.com

Agenda

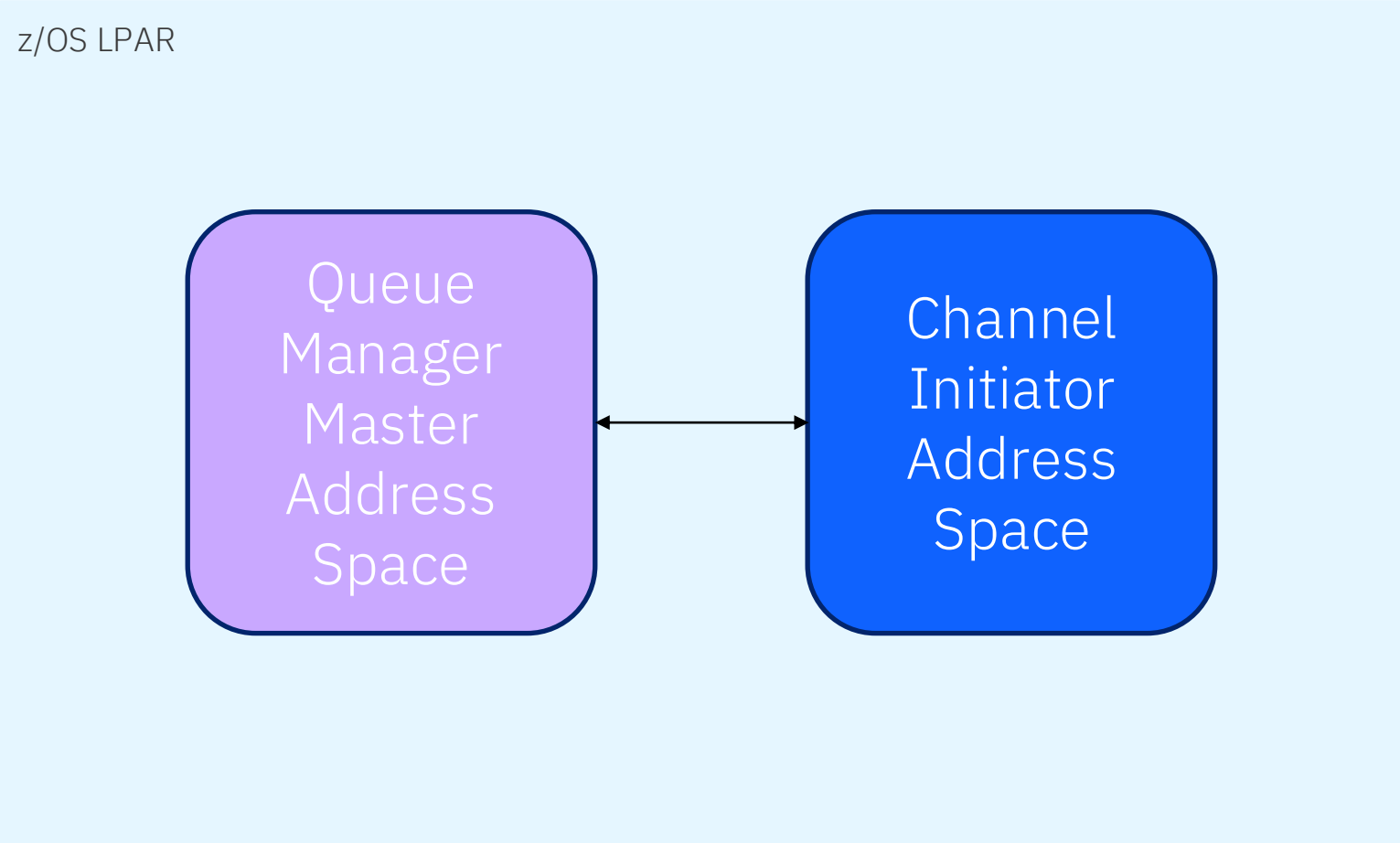
- What does a queue manager look like?
- Resource Managers
- How does MQ store messages?
- How does MQ ensure integrity?
- Performance considerations

What does a queue manager look like?

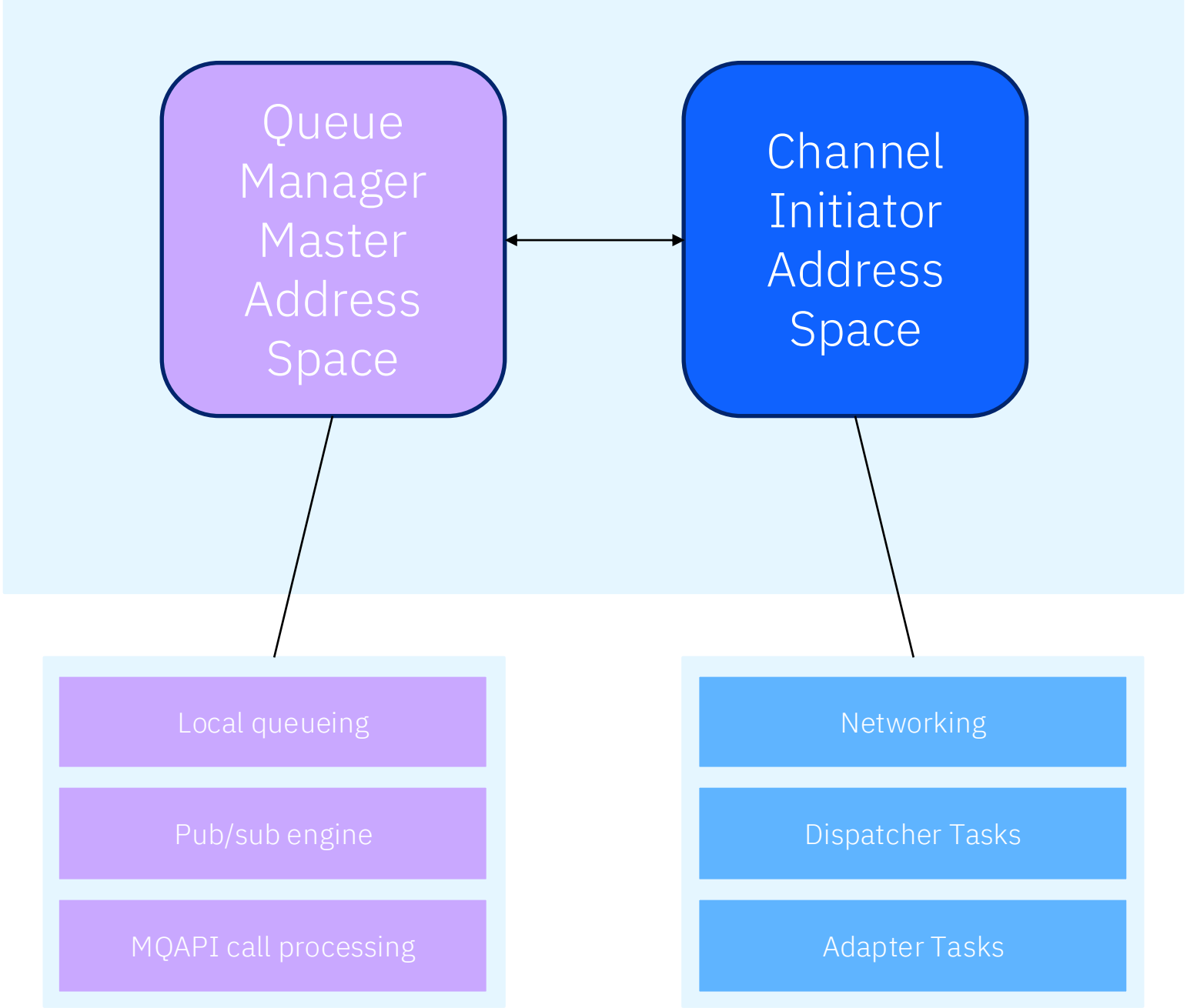
At a glance



At a glance

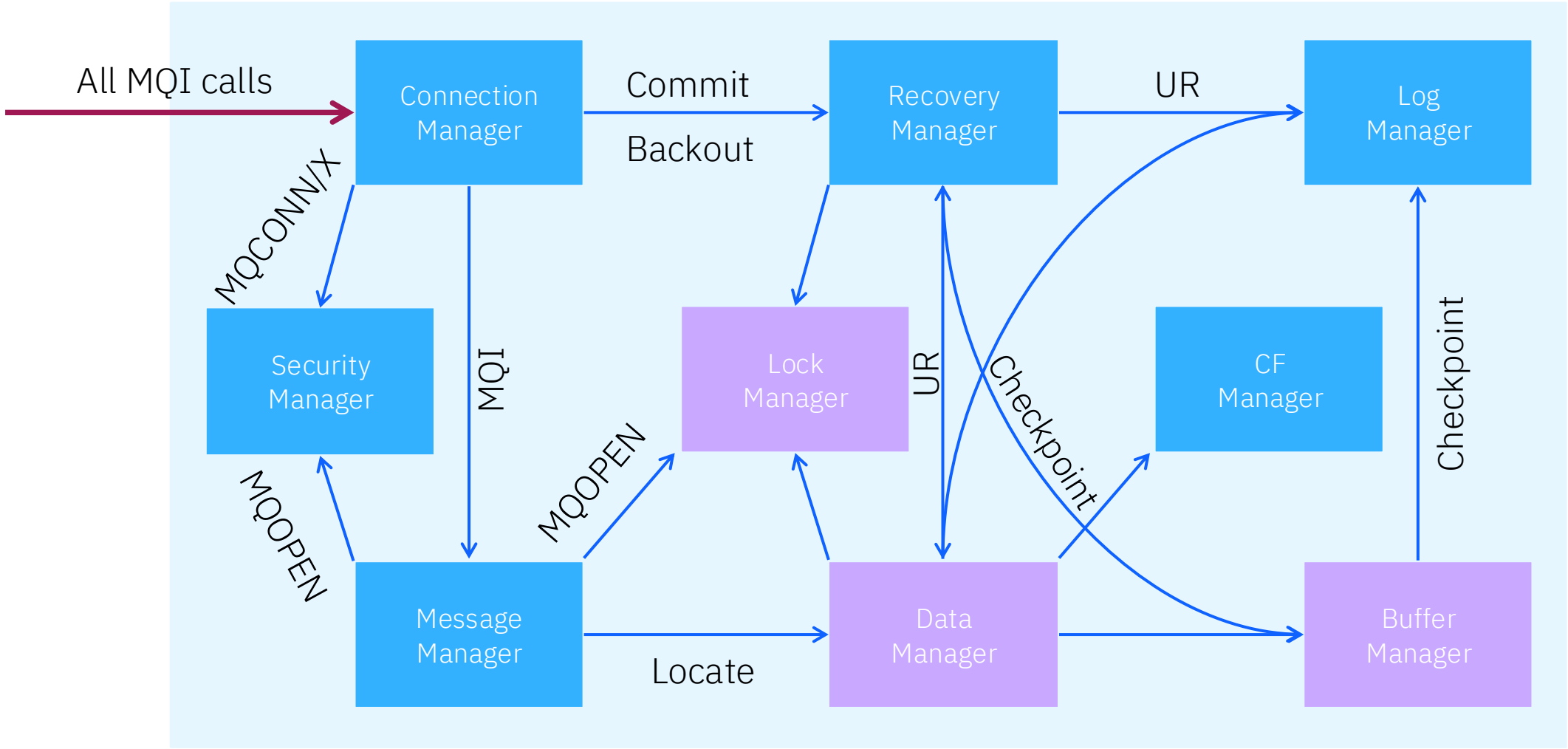


At a glance

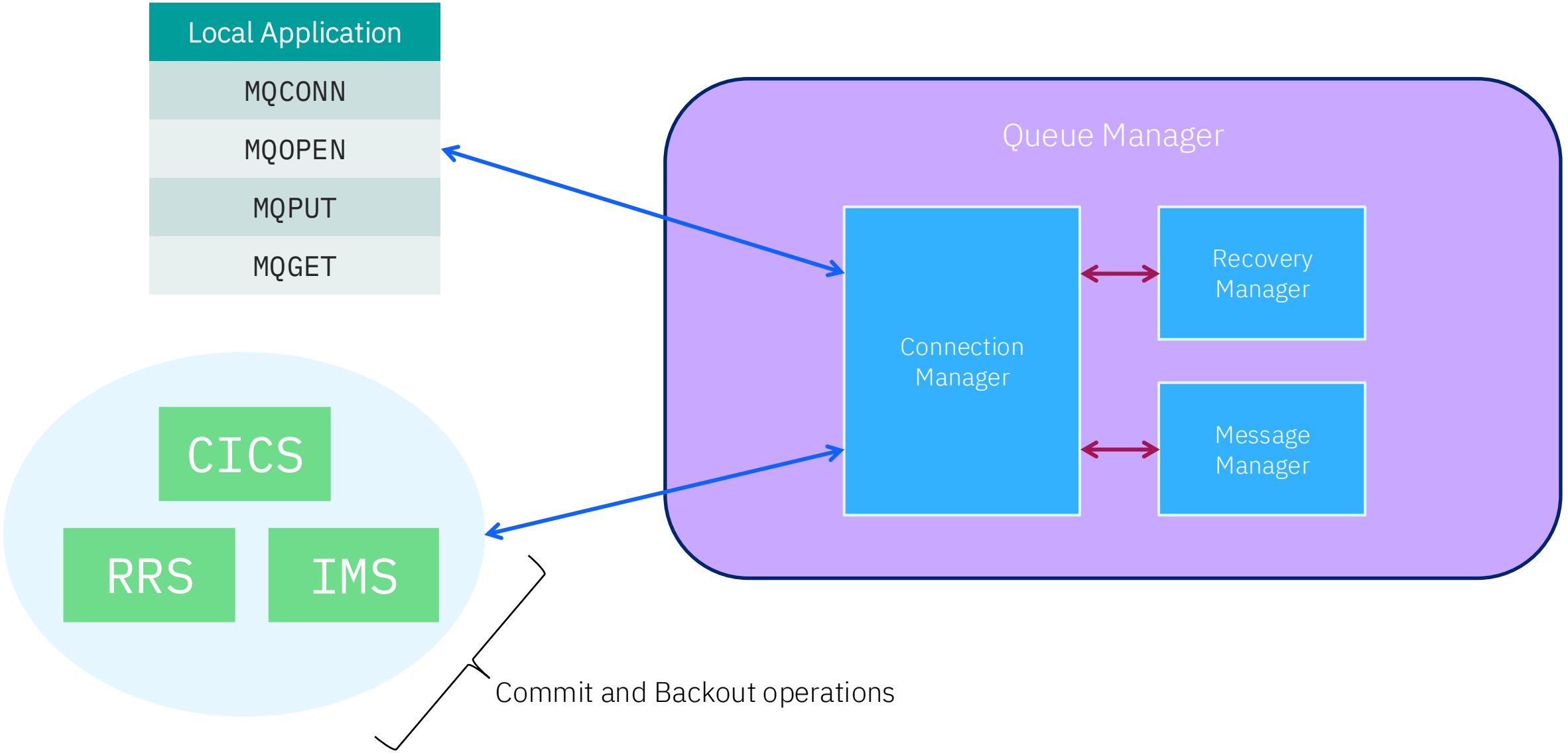


Resource Managers

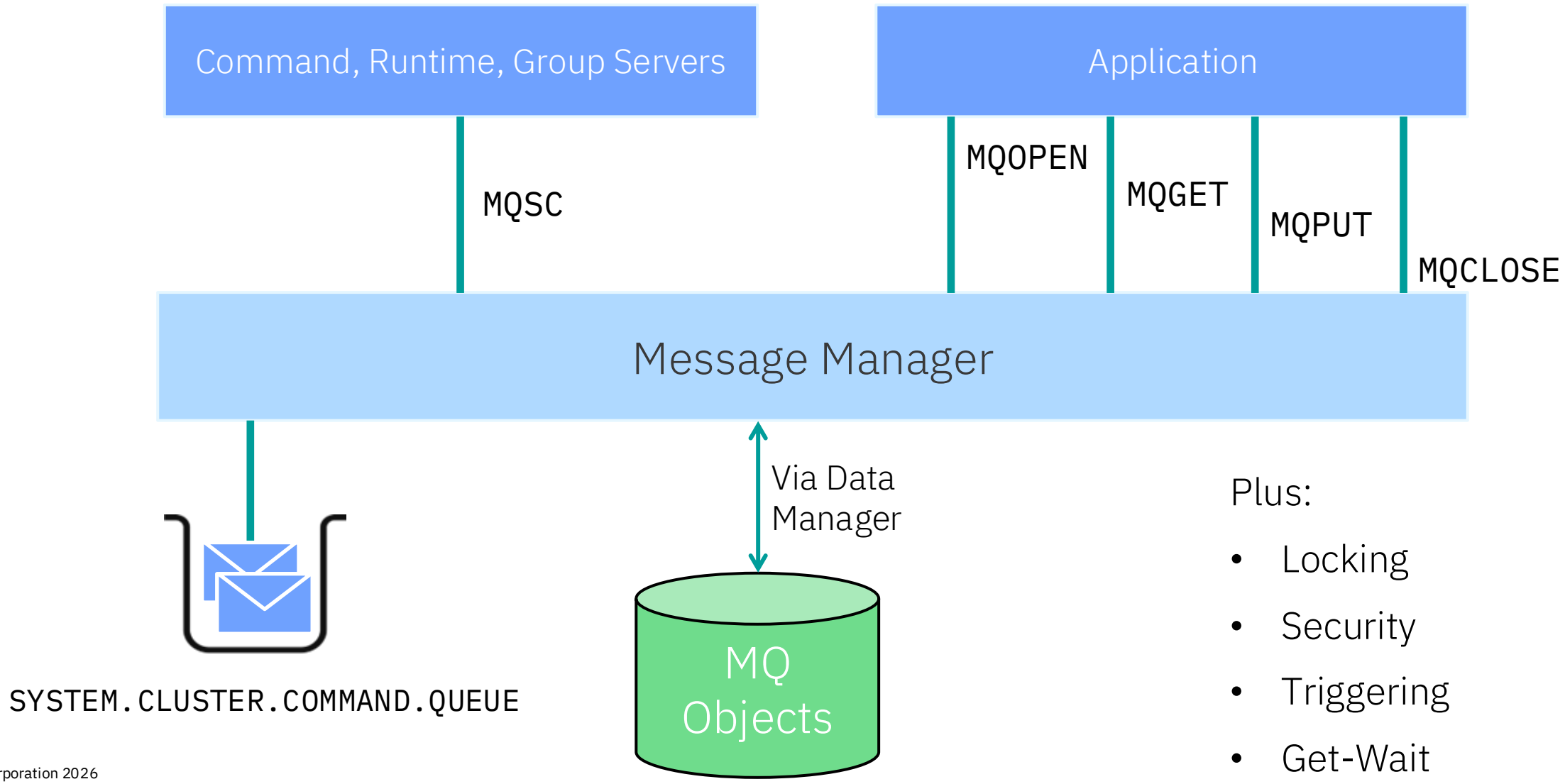
More than meets the eye!



Connection Manager



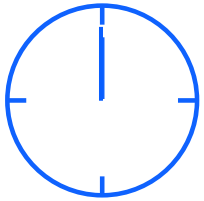
Message Manager



Plus:

- Locking
- Security
- Triggering
- Get-Wait

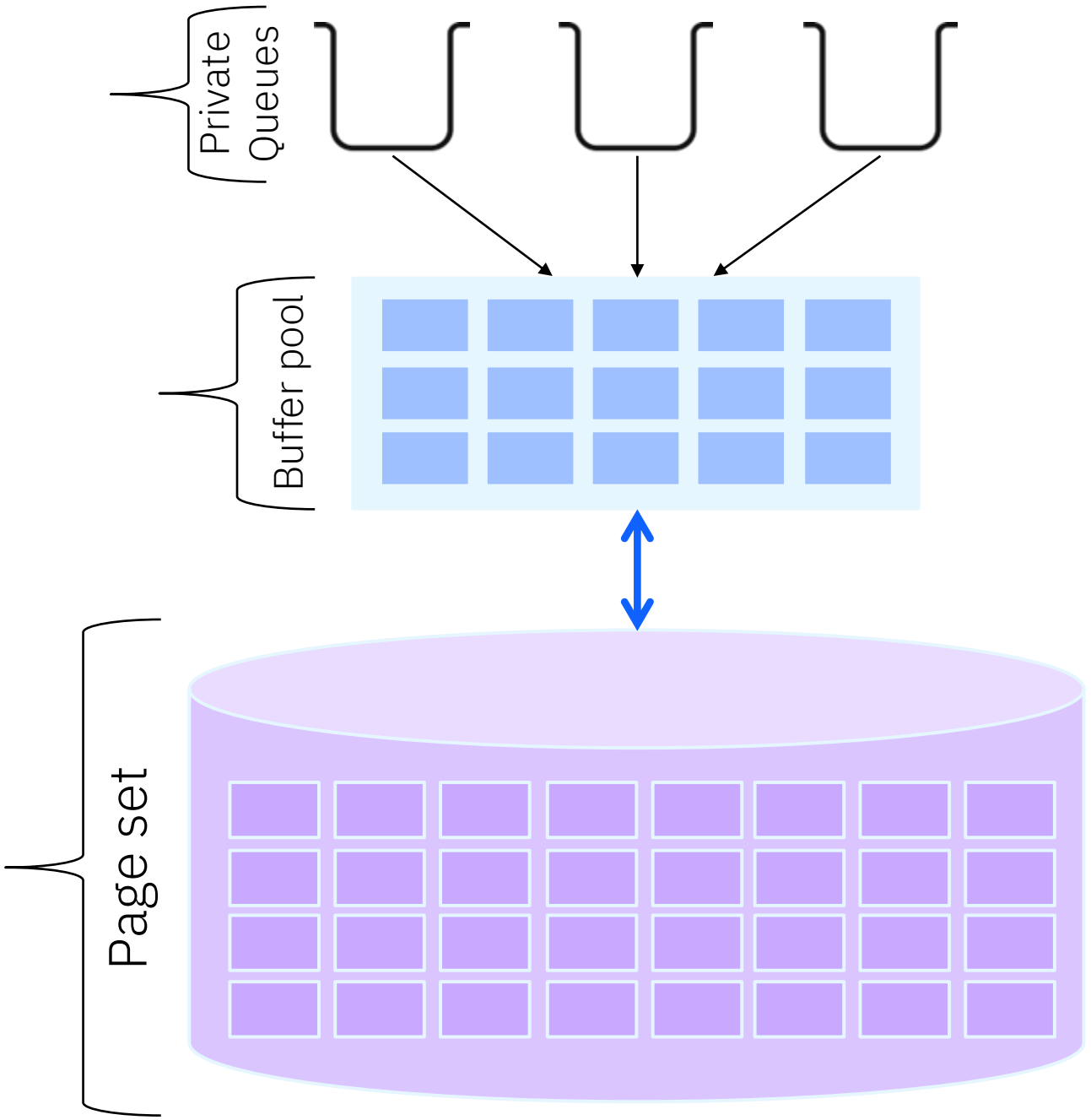
Recovery Manager

					
1-PHASE COMMIT	BEGIN	IN FLIGHT	IN COMMIT	COMMITTED	
1-PHASE BACKOUT	BEGIN	IN FLIGHT	IN BACKOUT	BACKED OUT	
2-PHASE COMMIT	BEGIN	IN FLIGHT	IN DOUBT	IN COMMIT	COMMITTED

The Buffer Manager and Data Manager

Private Queues - Storage

- Messages on private queues are stored in-memory in buffer pools, always in 4KB pages.
- If messages are not quickly consumed, they are moved to a page set on disk.
- Private queues are associated with a single buffer pool/page set pair
- MQ uses a **Deferred Write Processor** (DWP) to move in-memory messages to the page set.
- Under normal conditions, the DWP will asynchronously write messages to the page set when they have been in the buffer pool for 2 full log checkpoints



Private Queues - Storage

There are two thresholds which trigger additional movement of messages to disk:

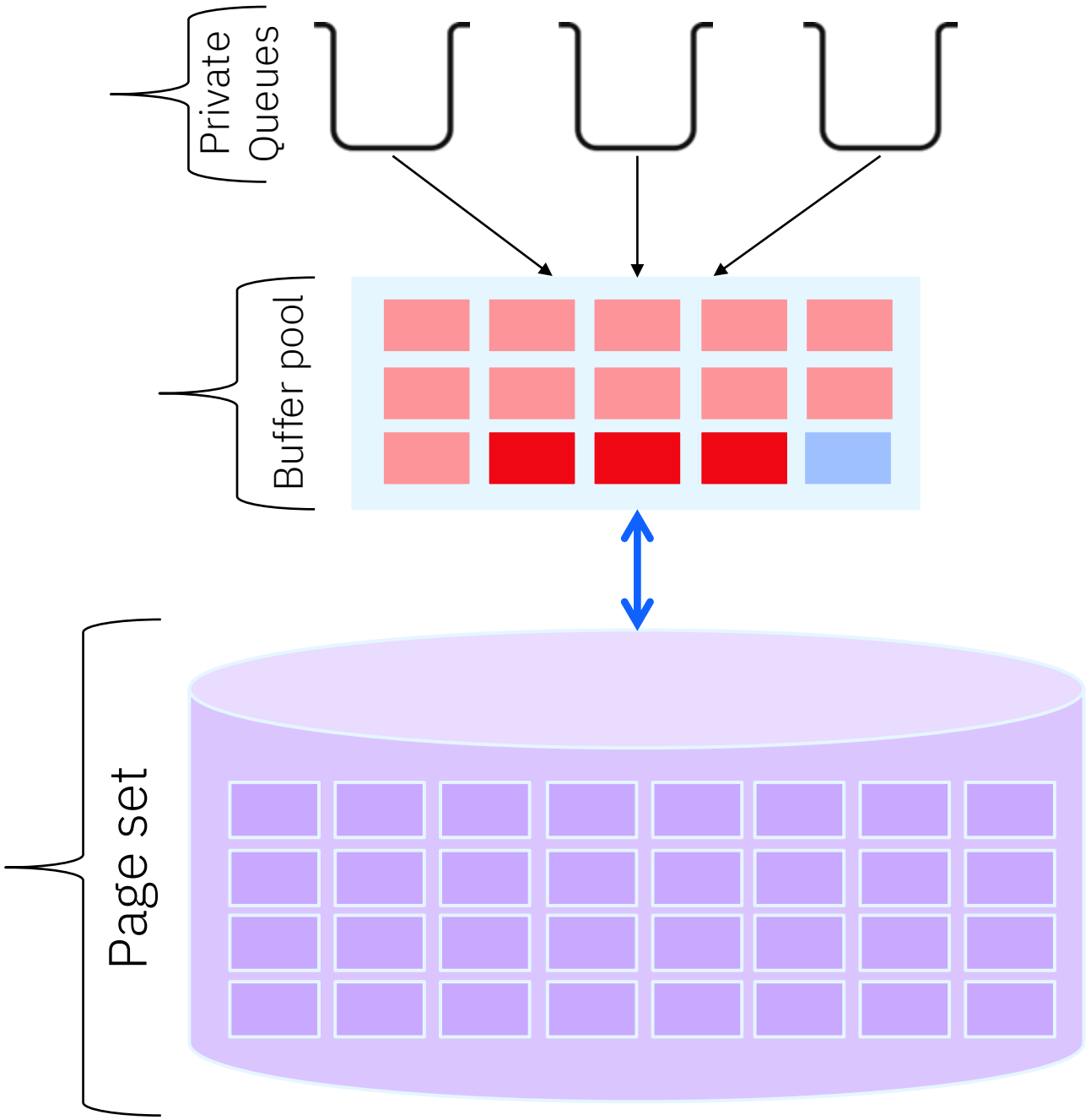
Deferred Write Threshold (typically 85%):

- DWP will asynchronously write messages to the page set

Buffer Pool Threshold (typically 95%):

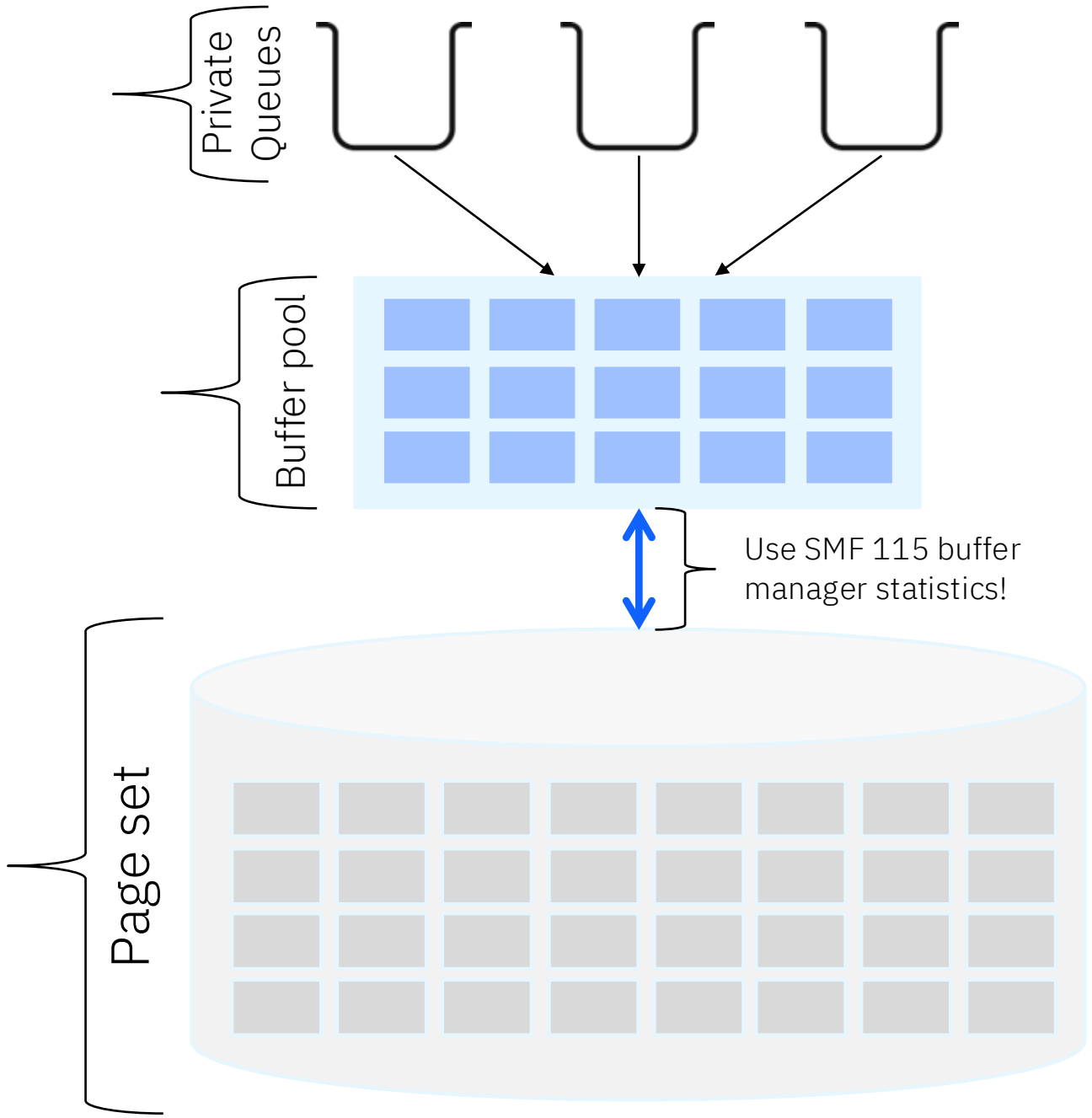
- DWP has not been able to keep pace with the rate of messages arriving in the buffers
- Switches to synchronous writes, impacting the performance of MQPUTs to the queue

Note: High Performance FICON (zHPF) can improve synchronous write times if you are hitting this limit often.



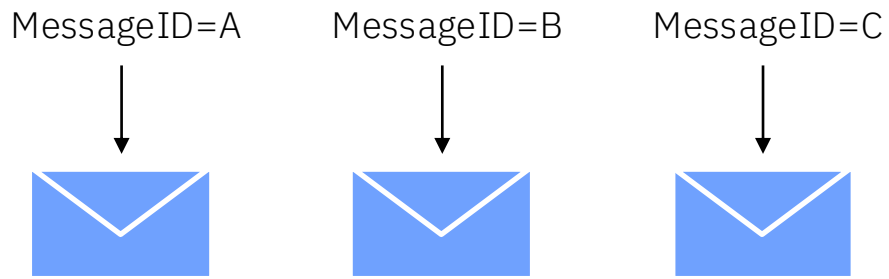
Private Queues – Performance

- Buffer pools are key to private queue messaging performance.
- They minimize the chance of the application having to wait for the queue manager to perform disk IO to retrieve a message.
- Buffer pools can be in 64-bit storage, meaning an entire 64GB page set can be stored in a single buffer pool if required.
- Normally, buffer pools are split into two types:
  - For long lived messages, buffer pools can be made quite small as messages will be written to the page set after two checkpoints.
  - For short lived messages, buffer pools can be relatively large to minimize page set IO for performance.



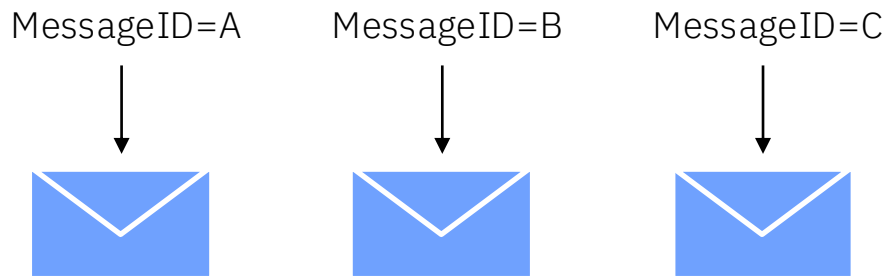
Index your queues!

- Specify the INDXTYPE attribute on local queues to tell the queue manager how to index the queue. Options are:
  - No index
  - By message ID
  - By correlation ID
  - By group ID
- Choose a value based on the most common approach of getting messages from the queue
- The queue manager will emit a message warning you if you repeatedly get by one of these indexable attributes, but the queue is not indexed.

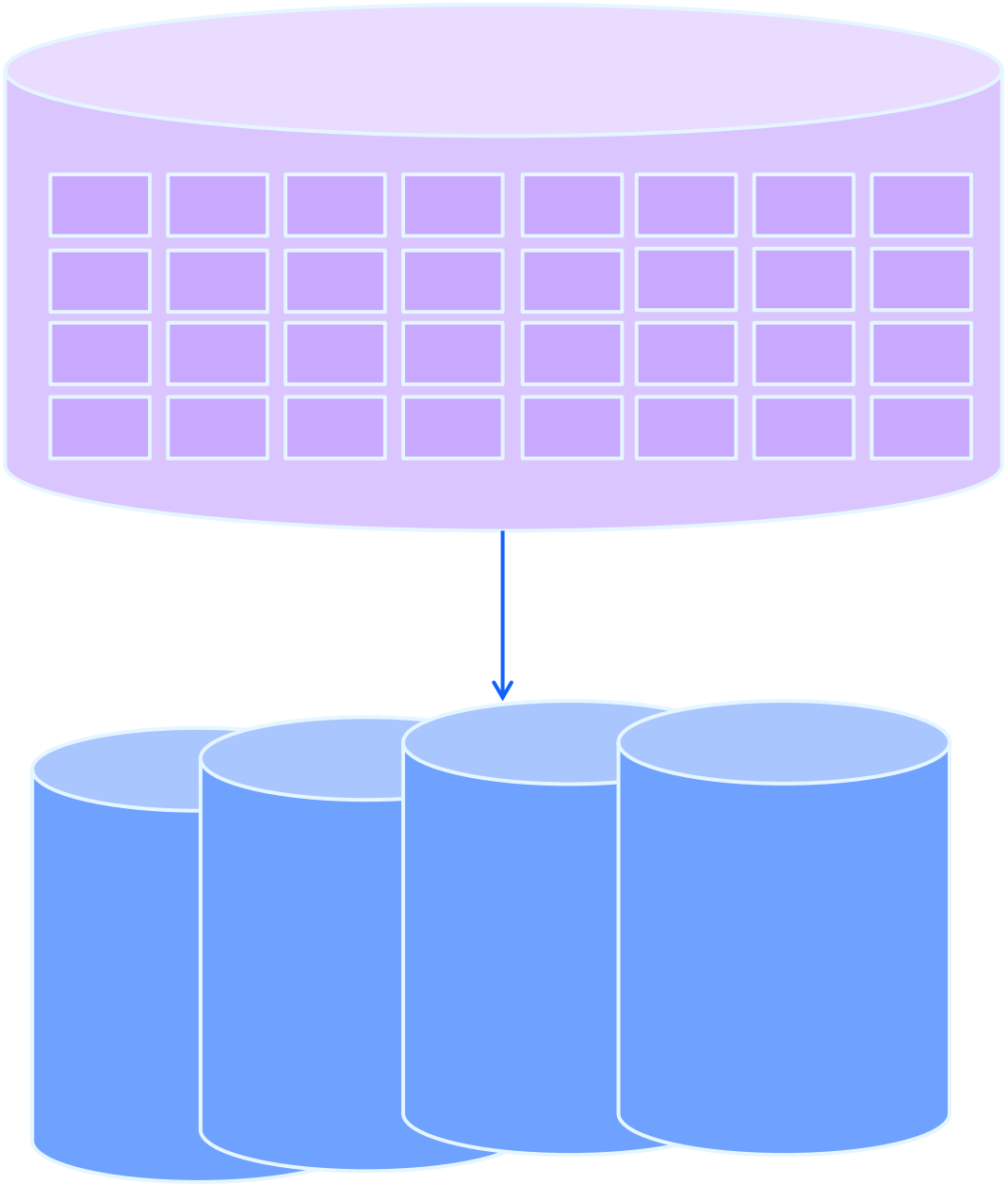


Index your queues!

- Private queue indexes are maintained in 64-bit storage.
- Each message on an indexed queue uses 136 bytes to maintain the index.
- Therefore, 10 million indexed messages would use 1.3GB of above-the-bar storage.
- If you’re going to index deep private queues, make sure you account for this in the MEMLIMIT attribute of your *MSTR JCL.



Resiliency for your private queues



Page set backups

- Many customers periodically back up their page sets in case of corruption.

- Full backup:
  - Shut down the queue manager
  - Back up the page sets
  - Disruptive, but simple and consistent
  - Doesn’t need earlier logs to recover

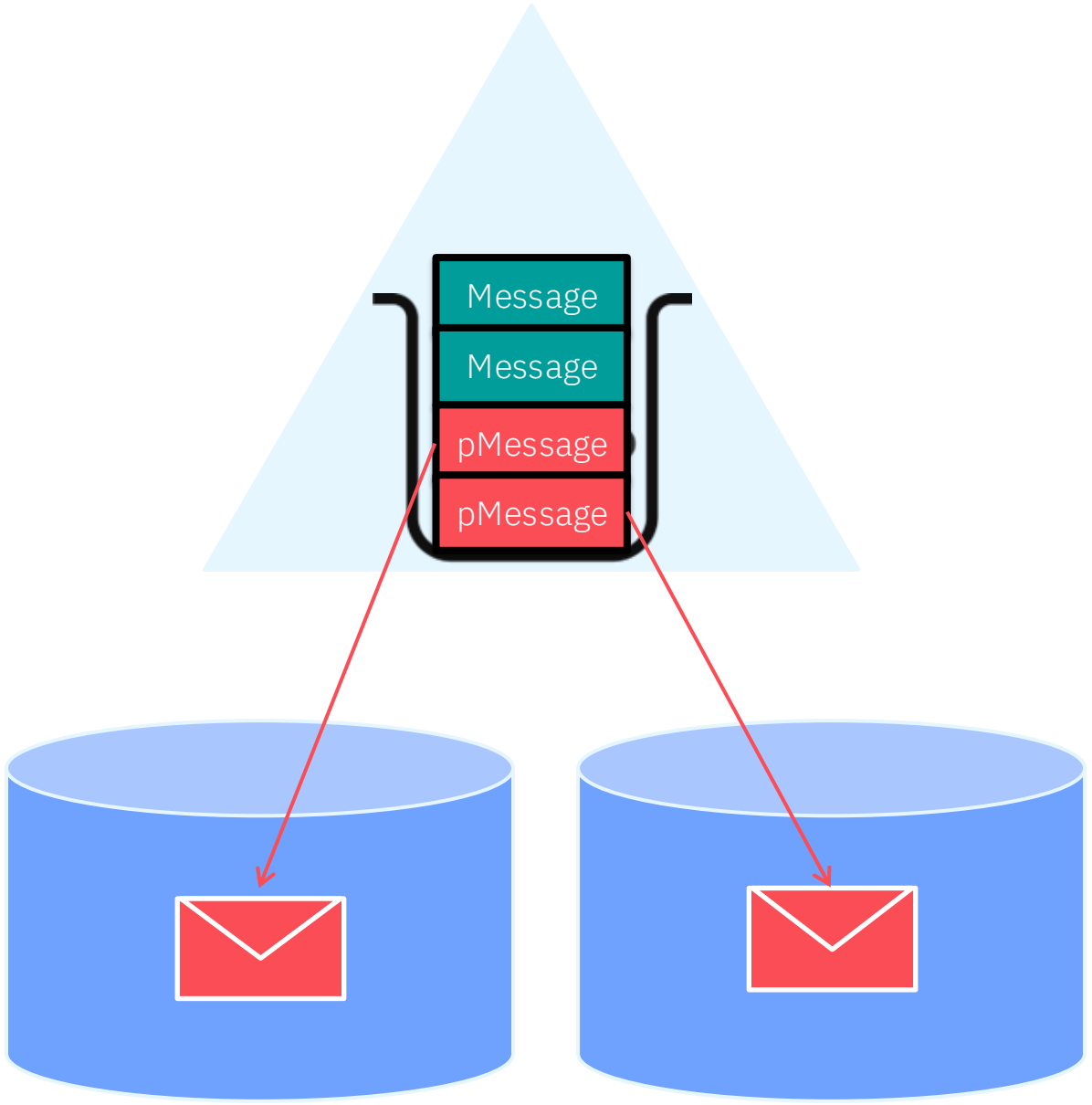
- Fuzzy backup:
  - Keep the queue manager running
  - Back up the page sets
  - Non-disruptive, but needs earlier logs and page sets to reconcile changes that are not in the page set

- There are no special considerations here, other than the DASD cost of backing up page sets that hold deep queues.

Shared queues

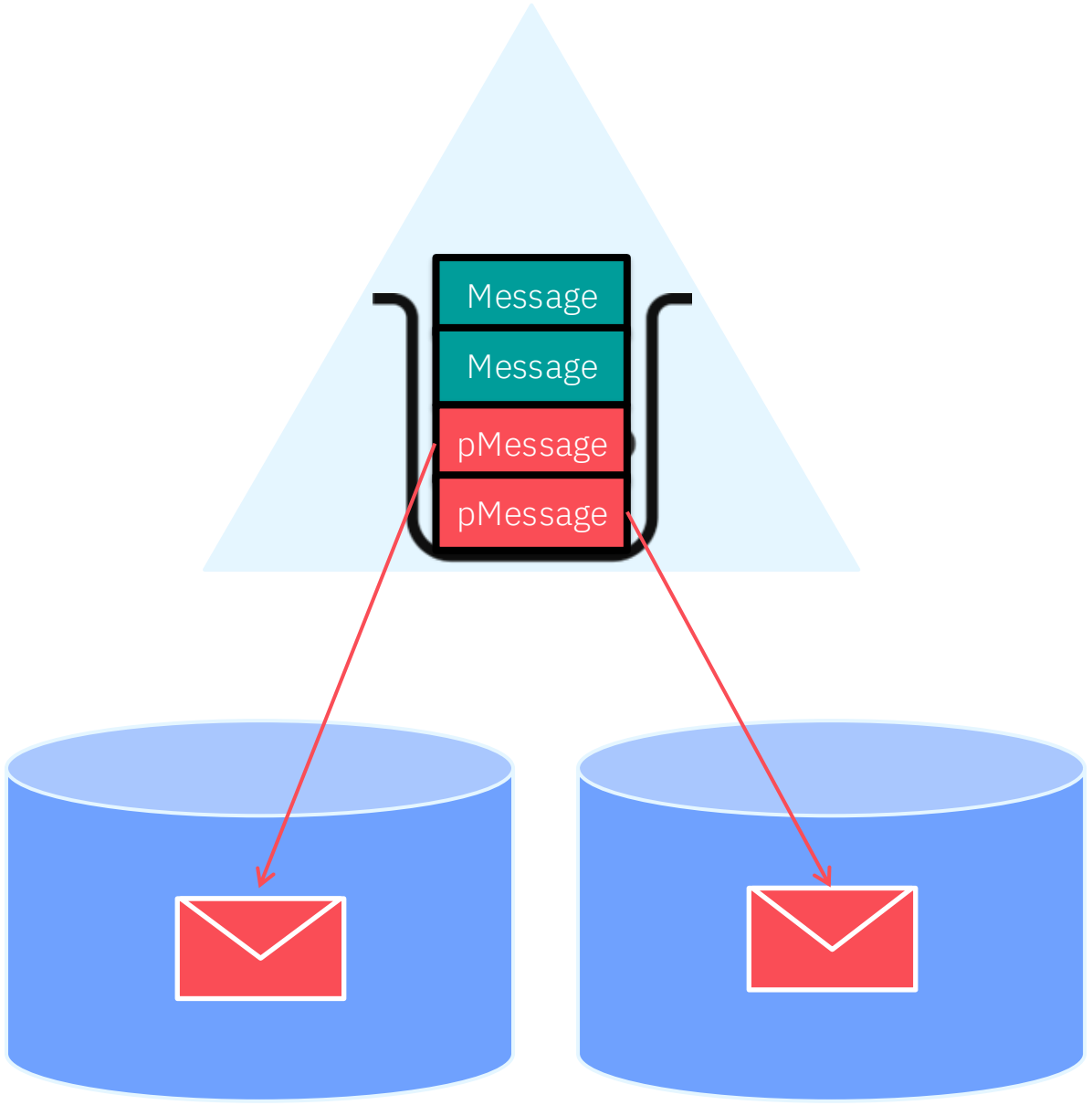
Shared Queues - Storage

- Shared queues are stored in the coupling facility (CF).
- Messages can be stored entirely in the CF, if they are < 63KB
- Larger messages are stored in your chosen offload system:
  - Shared message data sets (SMDS) is strongly recommended
  - Db2 is not recommended
- Large messages have pointers to their offloaded location stored in the CF instead.



Shared Queues - Storage

- The maximum supported CF structure size is 1TB.
- This is real storage, so relatively expensive.
- If SMDS is used, each queue manager gets its own SMDS data set for the structure.
- The maximum size of a single SMDS is 16TB.

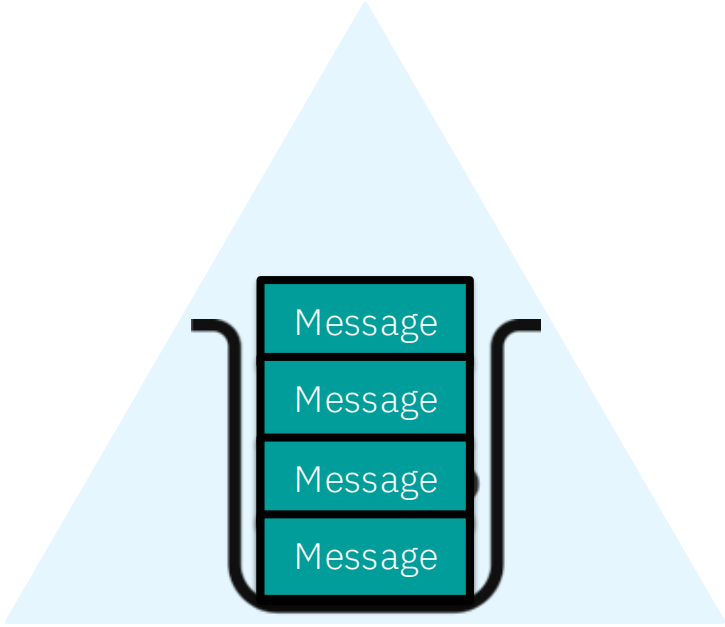


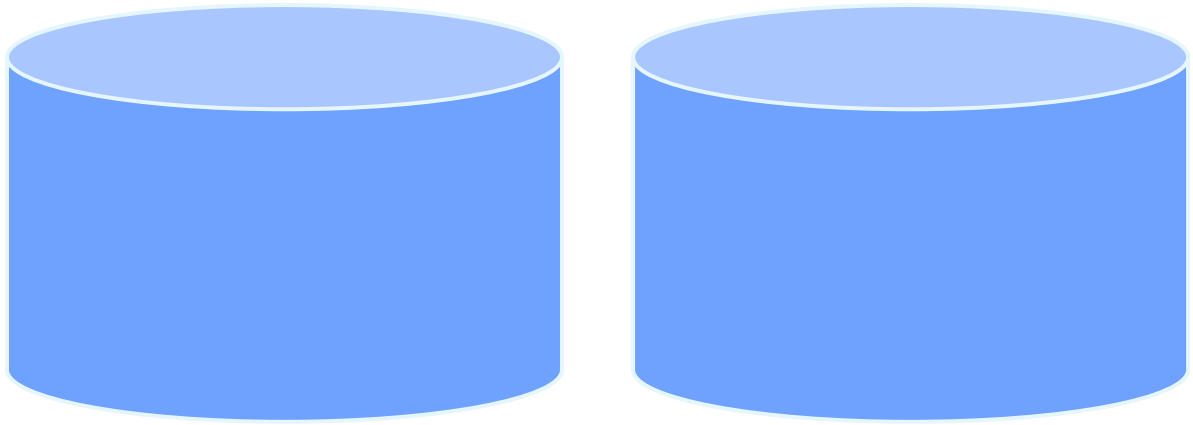
Shared Queues - Storage

- Shared queues perform best when the entire message is held in the CF, as the code path is minimized and there is no wait for slow disk IO.

- For messages which are <63KB, the best approach is to keep them in the CF and only offload them to SMDS when the queue starts getting deep (for example, during an outage).

- This provides the best of both worlds:
  - Fast message access in normal operation
  - Ability to store lots of messages in an outage





Shared Queues – Offload Rules

- Each MQ CFSTRUCT definition has three offload rules associated with it.
- These rules specify the maximum message size that can be stored in the structure, when the structure is a given percentage full.
- The default rules assume that no messages under the 63KB limit get offloaded until the structure is very full!

```
DEFINE CFSTRUCT(SHALLOWSTRUCT)
                CFLEVEL(5) …
                OFFLOAD(SMDS)
                OFFLD1TH(70)    OFFLD1SZ(32K)
                OFFLD2TH(80)    OFFLD1SZ(4K)
                OFFLD2TH(90)    OFFLD1SZ(0K)

DEFINE CFSTRUCT(DEEPSTRUCT)
                CFLEVEL(5) …
                OFFLOAD(SMDS)
                OFFLD1TH(10)    OFFLD1SZ(0K)
                OFFLD2TH(10)    OFFLD1SZ(0K)
                OFFLD2TH(10)    OFFLD1SZ(0K)
```

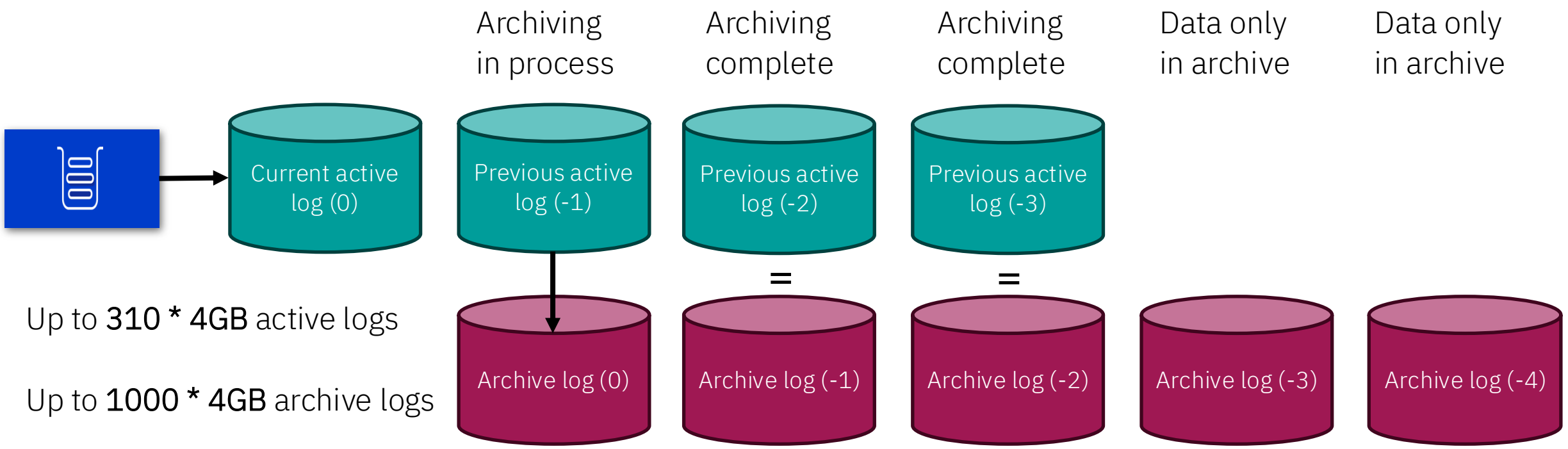
How deep can I go?

- We will see exactly where these numbers come from later
- * we haven't tested to the very ends of these limits. Care should be exercised if you want to approach these numbers!

	Approximate number of 1 KB messages
MQ on z/OS private queue	16.8 million
MQ on z/OS shared queue all in CF	613.6 million*
MQ on z/OS shared queue on SMDS	1.4 billion*
MQ on distributed	68.4 billion*

Let’s talk logging

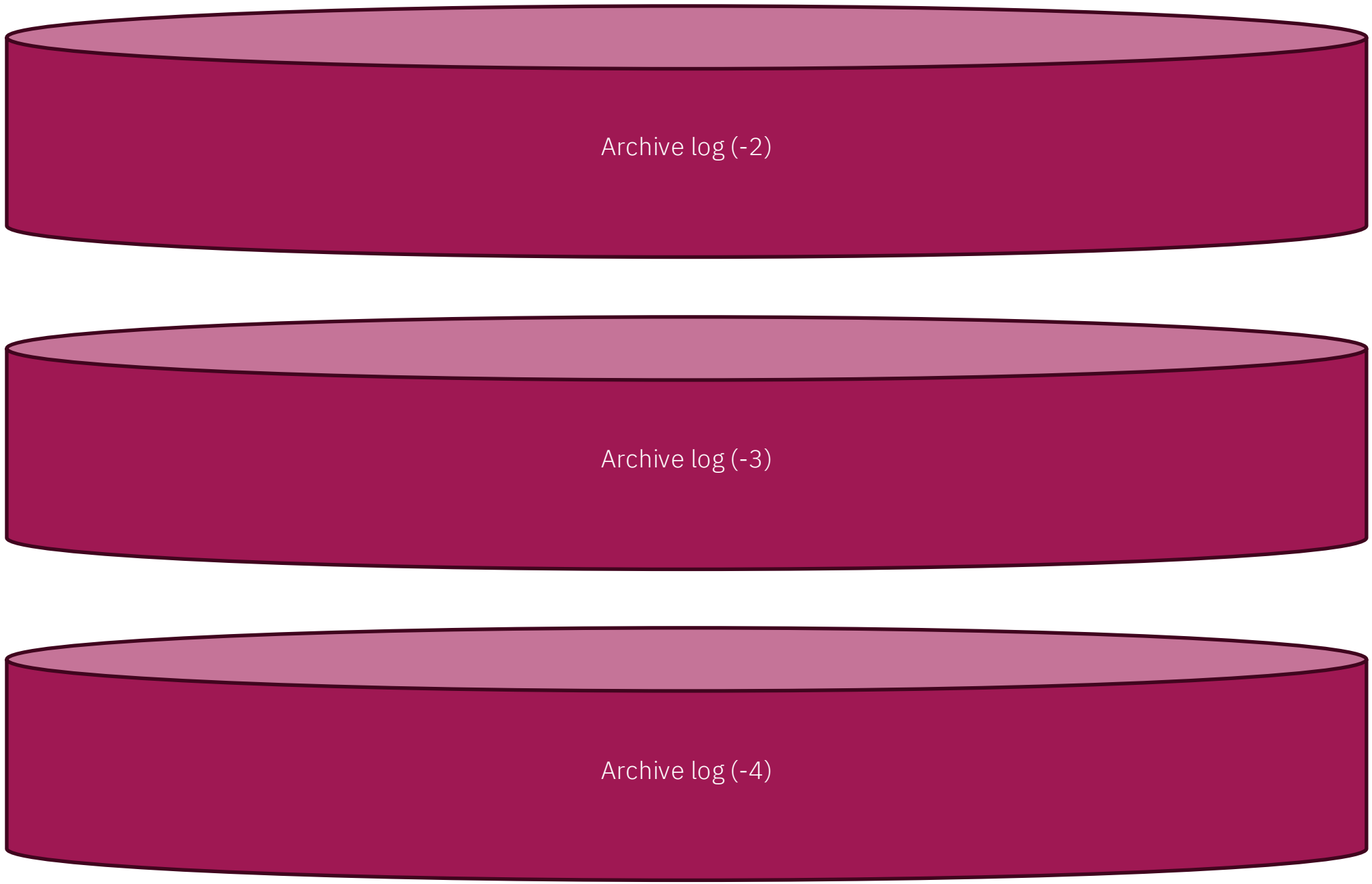
Logging



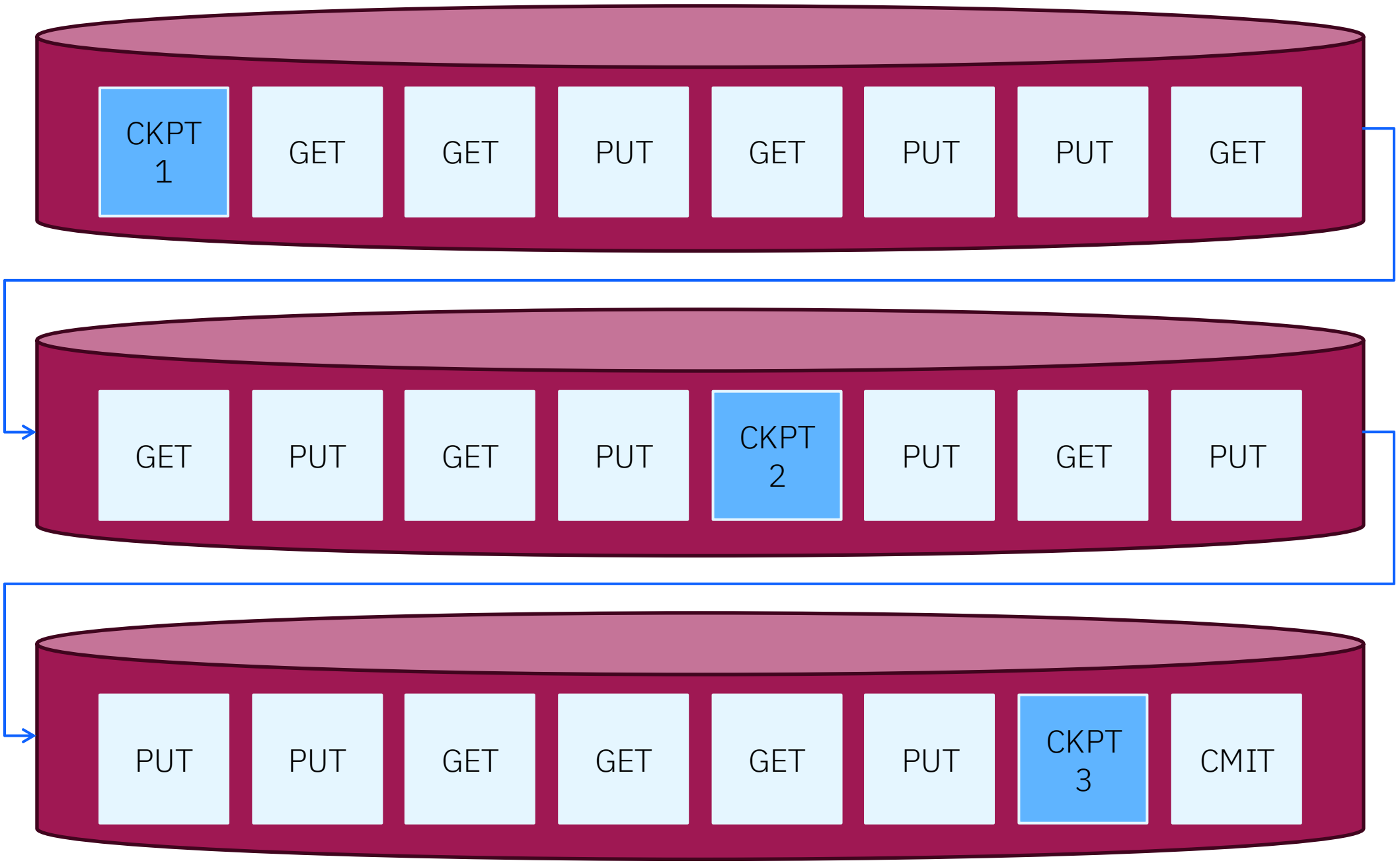
Up to **310 * 4GB** active logs

Up to **1000 * 4GB** archive logs

Logging



Logging


CKPT 1
GET
GET
PUT
GET
PUT
PUT
GET
GET
PUT
GET
PUT
CKPT 2
PUT
GET
PUT
PUT
PUT
GET
PUT
CKPT 3
CMIT

Recap

We have covered

Connection Manager

- The front door to the queue manager
- Passes requests to appropriate resource managers

Message Manager

- Application-centric
- Handles MQI calls
- Handles MQSC requests

Recovery Manager

- Transaction state
- Queue manager recovery

Buffer and Data Managers

- Internal interfaces to locate and work with the storage underpinning MQ objects
- We learnt how private and shared queues store your messages

Log Manager

- Interfaces with the logs that are critical to queue manager recovery
- How logging ensures integrity of the queue manager

Performance Considerations

A queue manager is not a database!

IBM MQ	Database
Most operations are disruptive	Most operations are non-disruptive
Optimized for high volumes of concurrent read/write	Optimized for write once, read many
Messages represent information flow	Tables associate with endpoints of information processing

A queue manager is not a database!

Let's think about an example.

1500 messages are put to a queue with the following properties:

- **500** with `colour=red`
- **500** with `colour=green`
- **500** with `colour=blue`

In a database, getting all the green entries would be trivial with an SQL query, and very fast, too.

But in IBM MQ, the impact of getting these green messages looks like this:

CPU time for the **MQOPEN** on the queue:
40 -> 130 microseconds

Average CPU time per **MQGET**:
60 -> 350 microseconds

To match a selector string, the queue manager must read every message on the queue, from the start, every time, for every MQGET, until it finds a match.

Large Messages

- Large messages **may** have performance impacts
- The biggest message you can send to a queue manager is 100MB
- Large messages have no impact on queue traversal by the queue manager (when searching for a specific message on a queue, for example)
- Setting the max message size on a queue this high has no inherent impact on performance

Performance can be degraded in two areas:

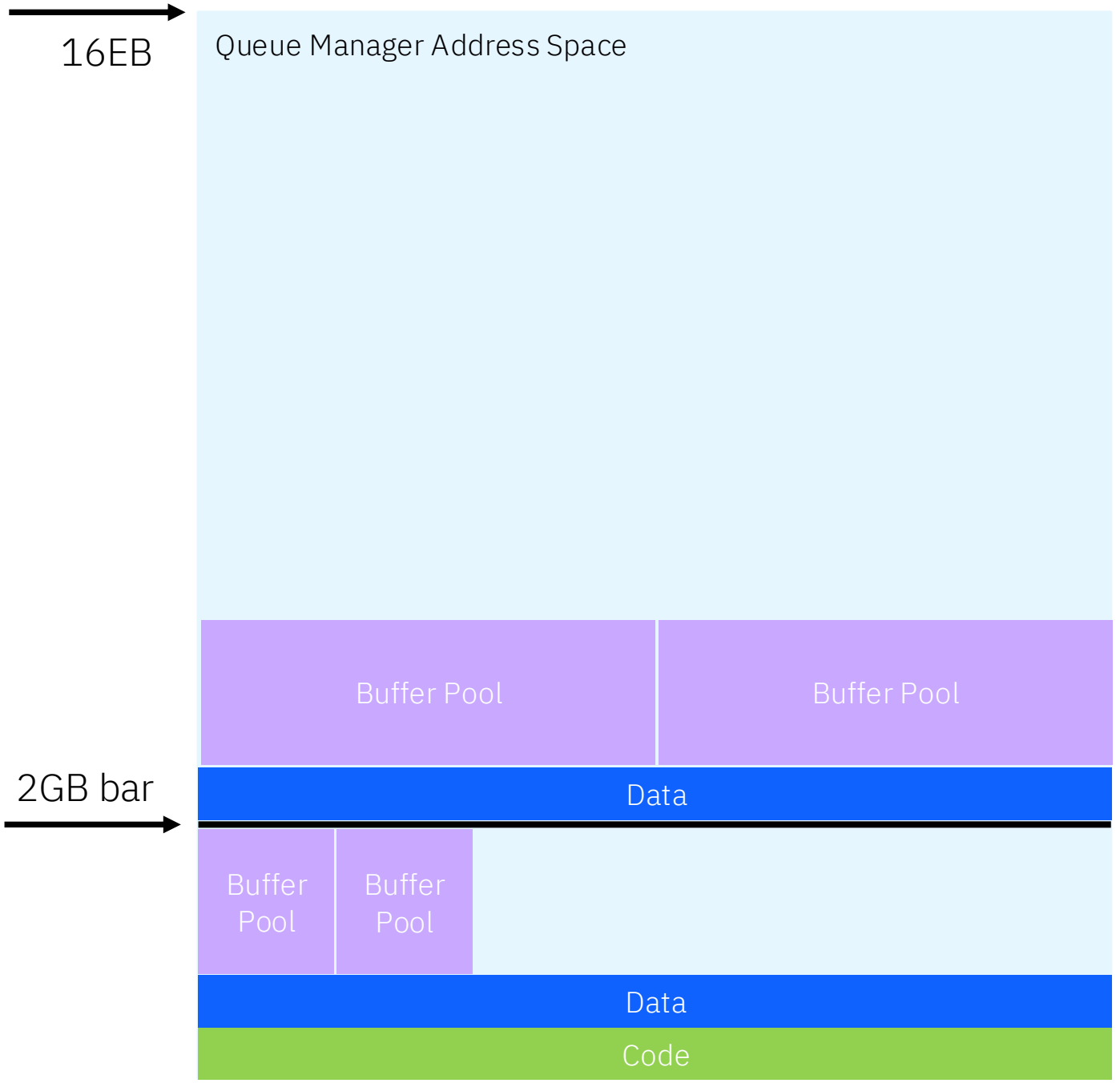
If you MQPUT many large messages, you might observe:

- Increased disk I/O (from page sets + logs)
- Increased CPU cost associated with the above

When you MQGET large messages, a significant number of pages must be traversed to retrieve the contents. This is made worse if the pages reside in the page set rather than a buffer pool.

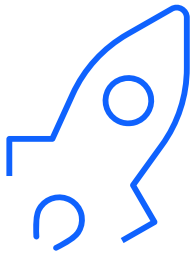
Use 64-bit buffer pools

- Often a quick win for performance
- More messages stay in fast in-memory buffers
- Larger buffer pools reduce synchronous I/O from the DWP
- Frees up 31-bit storage for other queue manager needs
  - For example, every application that connects to MQ reserves an amount of 31-bit storage in the queue manager address space

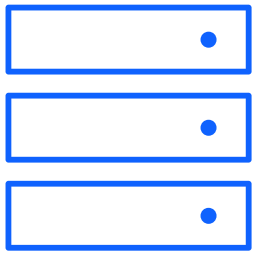


A bit of showing off

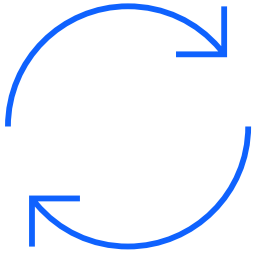
https://ibm-messaging.github.io/mqperf



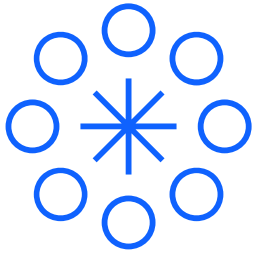
2 million messages per second on a single z/OS queue manager



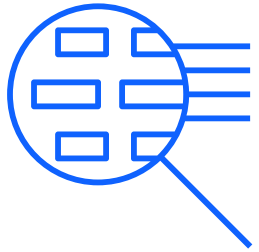
6x faster sustained logging with zHyperLink



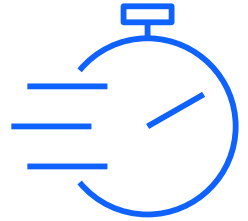
300,000 request-reply transactions per second



Over 97,000 persistent shared-queue messages per second



Rapid recovery with 1.2GB/s active log reads during recovery



Buffer pool writes at up to 600MB/s to avoid synchronous I/O slowdowns

Experience more with IBM

Visit us at the IBM Booth #113

After a full day of technical sessions, take a break with us!

Connect with our experts, snap a photo with the z17 Plexi or the latest Telum II, and get an up-close look at our Spyre Accelerator.

Come back each day for fresh topics and demos at our expert stations.

Think 2026

Join 5000+ senior business and technology leaders who are seizing the AI revolution to unlock unprecedented growth and productivity at Think 2026.

Find out more information using the QR code below.

IBM Digital Asset Haven

IBM Digital Asset Haven is the operational backbone for financial institutions and regulated enterprises entering the digital asset economy.

Find out more information using the QR code below.