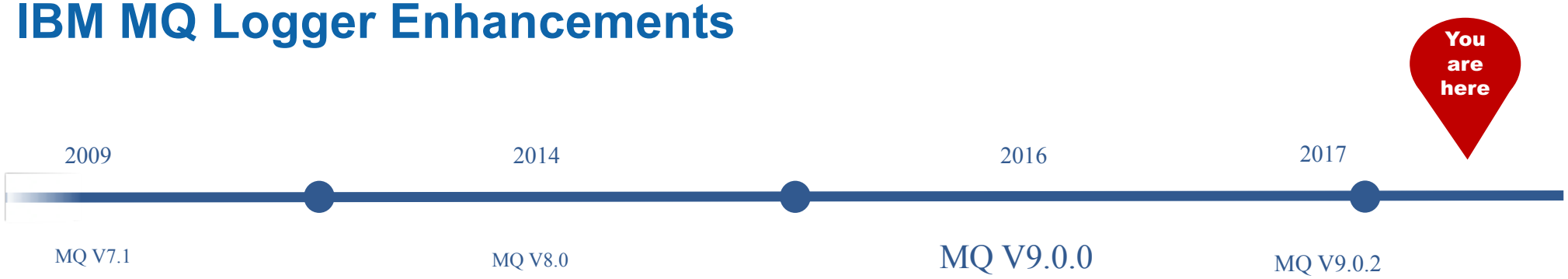
What's New in MQ Logging?

Christopher Frank

chrisfra@us.ibm.com

IBM MQ Logger Enhancements



- The IBM MQ Logger has seen a number of enhancements in recent years:
  - **v7.1** – Significant internal changes to improve throughput, reduce contention
  - **v8.0** – Further internal changes to reduce log I/O, improve log write performance
  - **v9.0.2** – Significant enhancements to Linear logging, management, reporting, monitoring

IBM MQ Logger Enhancements

N

O

T

E

S

The Recovery log has been a part of IBM MQ since the beginning, as it is the means by which MQ can assure persistent messages are not lost or duplicated following a failure.

IBM MQ has evolved over the years, and the Logger component has seen improvements over the past several releases of the product in a number of areas:

- MQ V7.1 (2009) saw a significant rewrite of the Logger, with changes intended to improve overall throughput, especially when under heavy load. These changes were internal to the Logger and so largely unseen, at least directly.
- MQ V8 (2014) introduced further changes, with new checkpointing default values intended to reduce I/O to the log, as well as improvements to how log writes were performed.
- MQ V9.0.2 (2017) again introduced significant enhancements to Linear log management in particular, with a goal of reducing the performance cost of using linear logging, as well as simplify the administrative effort required for managing log extents. Further changes in the areas or monitoring and reporting provide administrative benefit to users of both Circular as well as Linear logging.

The focus of this presentation will be on these latest new features and how customers can benefit from using them.

Logging Overview

- IBM MQ records all the information you need to recover from a queue manager failure in a recovery log
- This includes:
  - Creating and deleting MQ objects
  - Persistent message updates
  - Transaction states
  - Changes to object attributes
  - Channel activities
- The log contains the information you need to recover all persistent updates to message queues
- However, IBM MQ also relies on the disk system hosting its files
  - If the disk system is itself unreliable, information, including log information, can still be lost

Logging Overview

N

O

T

E

S

IBM MQ uses a write-ahead logging strategy to record all the information needed to recover from a queue manager failure. This includes information related to the creation, alteration or deletion of MQ objects, as well as persistent message updates, transaction status and channel activities.

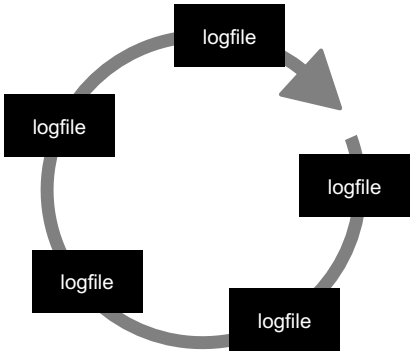
The log contains the information you need to recover all persistent updates to message queues, and optionally, to recover MQ objects themselves. However, to do this it relies on the underlying storage system hosting its files – meaning the recovery log is only as reliable as the storage on which it is recorded. If the storage system (Disk, SSD, NAS/SAN, etc) is itself unreliable, information, including log information, can still be lost.

Types of Logging

MQ always logs all the data you need to recover from a queue manager failure in a recovery log

Restart recovery (**circular** and **linear** logging)

Enough information held in the log files to rebuild MQ resources to the level that they were at prior to the queue manager stopping



Circular logging

Media recovery (**linear** logging only)

Enough information held in the log files to rebuild MQ resources in the event of losing or corrupting MQ data

Linear logging comes with an added cost in terms of performance and administrative effort

Linear logging

Types of Logging

NOTES

MQ supports two types of logging: Circular and Linear. Both types record enough information in the log files to rebuild MQ resources to the level that they were at prior to a queue manager failure.

With Circular logging, all the necessary log files are allocated up front. You never run out log files as they are used in a wrapping fashion. This incurs the least overhead in terms of performance and administrative effort. The caveat is that only restart recovery is supported.

With Linear logging log extents are created as required in a sequence. Because this has the potential to go back much further in time, the recovery capabilities can go beyond just restart recovery – Media recovery can be supported as well.

The caveat with Linear is that log space is not reused, so steps must be taken to prevent the available disk space from filling. Dealing with these inactive log files, as well as the recording of media images, is an administrative task.

Pros and Cons

- Circular logging:
  - Has a lower administrative overhead
  - Has better performance
- Linear logging:
  - Enables media recovery
  - Can provide a long-term message audit trail
    - For both customers and IBM use
  - Provides a level of redundancy upon disk failure
- Neither approach will help if:
  - The recovery log is corrupted (e.g. disk failure)
  - A queue or messages on it are accidentally deleted
  - The log itself is damaged

Pros and Cons

N

O

T

E

S

Circular logging is often preferred and what is usually recommended, as it has a lower administrative overhead as well as better performance.

Linear logging offers several additional capabilities, including:

- Enabling media recovery, meaning damaged MQ objects can be recovered as opposed to having to delete and redefine them.
- The log files can be archived as they become inactive, and can serve as a long-term message audit trail, both for both customers as well as IBM use.
- Media recovery means customers have an additional level of redundancy to protect against disk failure.

The protection the recovery log provides is not absolute - there are situations from which neither approach will rescue you:

- If the disk failure includes the recovery log – no recovery log, no recovery.
- If a queue or messages on it are accidentally deleted.
- The log itself becomes damaged or corrupted in some way – again, no recovery log, no recovery.

Linear Logging - Viewpoint

- Today, there aren't many good reasons for using Linear logging
  - RAID disk redundancy helps protect against media failure (the primary justification)
  - It can provide a (user-unfriendly) message audit trail
- MQ customers are often not aware of the performance penalty that Linear imposes
  - Which can be significant!
- But they are usually very aware of the administrative cost
  - Recording media images frequently enough to make recovery time satisfactory upon failure
  - Archiving and/or deleting old log extents

Linear Logging - Viewpoint

NOTES

One could make a very strong argument against the use of Linear logging today, for some very good reasons:

- Media technology and reliability today is very different from what it was when MQ was originally introduced. For example, RAID disk redundancy helps protect against media failure, and media failure is often the primary justification for using Linear.
- Another justification is that it provides a built-in form of message audit trail, albeit a very user-unfriendly one.

Other reasons for leaning towards Circular vs Linear logging are the costs associated with using the latter:

- MQ users are often not aware of the performance penalty that Linear imposes, which can be quite significant. This is due both to the overhead of continually formatting new log extents, as well as the cost of archiving inactive log files if this is being done. This both impose a processor as well as I/O cost which will in turn impact the performance of the queue manager that is sharing those same processors and file system.
- What MQ admins are usually very aware of are the administrative cost associated with Linear logging. While the formatting of new log extents is handled by the queue manager, other necessary tasks relate to linear log file management and cleanup must be dealt with by the MQ administrator. SupportPacs such as MS0L, M073, and MS62 are available to assist with this, but are provided as-is and are not a supported part of IBM MQ. Media recovery also requires the periodic recording of media images, which must be captured frequently enough to make recovery time satisfactory following a failure.

Linear Logging - Reality

- The flip side of the coin:
  - Even with RAID disk redundancy, there are still cases where media recovery can provide some benefit
    - Can recover from accidental deletion of a queue file
    - Damaged objects (as opposed to data on disk) can be recovered
      - Although maybe not administrative errors or a damaged log
  - Sometimes a user-unfriendly audit trail is better than no audit trail at all
  - If using Linear today, switching to Circular is not a simple matter
    - Queue managers must be deleted and recreated

- MQ customers have long requested enhancements to Linear that would enable them to keep the above benefits while relieving some of the performance and administrative burden

Linear Logging - Reality

N O T E S

But the reality is that a not-insignificant number of MQ users continue to use Linear logging. Reasons for this include:

- Even with the disk redundancy provided by RAID technology, there are still cases where media recovery can provide some benefit. These include the ability to recover from accidental deletion of a queue file (which RAID technology obviously cannot help with) as well as recover damaged MQ objects and their contents. Of course, media recovery may not address every possibility – data might be lost due to administrative errors or a damaged log.
- No one will argue that log files are a very user-unfriendly form of message audit trail. But sometimes a user-unfriendly audit trail is better than no audit trail at all.
- For many, the biggest challenge might be simply that they have queue managers that have been configured to use Linear logging, and switching these to use Circular is not a trivial task, give that such queue managers must be deleted and recreated

For these and no doubt other reasons, MQ users have long requested enhancements to Linear that would enable them to keep the above benefits while relieving some of the performance and administrative burden

MQ Logging – What's New?

- Changes to logging defaults to potentially improve the performance of logging
- New statistics provide greater insight into logging behavior
- New options to potentially improve the performance of Linear logging
- New options that extend the capabilities of media image recovery
- New options to automate much of the Linear logging administration burden
- Improved documentation and messages

MQ Logging – What's New?

N

O

T

E

S

In MQ V8 there were changes made to potentially improve the performance of logging. We'll review those quickly to make sure people are aware of what those were, how they may benefit from them, and in a small number of cases, see a negative impact and so need to make some adjustments to the logging parameters.

We will also review new statistics related to logging that were made available in V9, and how you can obtain those and gain some insight into the logger's behavior.

There was also adding some improved documentation and messages that should help administrators better understand how to size the log, and to see whether the sizes in place need periodic adjustment.

The above will benefit customers using both circular and linear logging. The most significant changes made to logging in V9 specifically benefit customers using linear logging. We will be reviewing these in detail in this presentation. The areas that were enhanced include the following:

1. New options were added that will potentially improve the performance of Linear logging
2. New options were also added that extend the capabilities of media image recovery
3. New options were added to automate much of the Linear logging administration burden

Logger Performance Changes in V8

- IBM MQ V8 included changes to reduce log writes
- When possible the logger will use writev() when writing to the log
  - Allows writing continuous data from the end and the start of the log buffer in a single I/O
    - Avoids the need to do an additional log force when the log buffer wraps
  - Windows does not support writev() and so behaves as before
- Default checkpoint frequency is now 50000
  - Prior versions of MQ defaulted to 10000
  - If you used *CheckPointLogRecdMax* you may want to revisit
  - In some cases could require you to increase the size of the log (or reduce the frequency)

http://www-01.ibm.com/support/docview.wss?uid=swg21201999

Logger Performance Changes in V8

N

O

T

E

S

IBM MQ V8 saw several changes relating to logging that were intended to reduce I/O to the recovery log and potentially improve queue manager performance, particularly when running heavy persistent message workloads.

One change was to how the logger handled writing data to the log. Previously, if the log buffer were to wrap, an additional log force would be needed to write all the buffered data to the recovery log. With MQ V8 the logger will when possible *writev()* when writing to the log. Using this form of write allows writing continuous data from the end and the start of the log buffer in a single I/O operation, reducing the number of writes needed. This change is on most Distributed platforms, but not Windows – the reason being that Windows does not support writev(), so on this platform the logger behaves as it did in previous releases.

Another change was to increase the default checkpoint frequency from every 10,000 to every 50,000 log operations. It was determined that this increase would not excessively impact typical recovery times, but would provide a substantial performance benefit by greatly reducing how often the logger would perform checkpoints. Two things to note on this change:

1. If you were using a non-default value for *CheckPointLogRecdMax,* perhaps to decrease the number of checkpoints taken, you may want to revisit whether you need to do that any longer.
2. Also, it was found that, in some cases, this resulted in increased LOG_FULL conditions – should you encounter this, you will want to either increase the size of the recovery log, or manually increase the checkpoint frequency by specifying a value lower than 50000 using the *CheckPointLogRecdMax* tuning parameter.

What's New in v9.0.2?

Automatic Media Imaging

Media Recovery

- One differentiator of Linear vs Circular logging is the ability to do Media recovery
  - Allows you to restore an MQ object from an image in the log if it has becomes damaged
  - Particularly useful when that object contains business data (e.g. persistent messages)

- Media recovery requires that images be captured periodically
  - This has always been an administrative responsibility
  - Must be done frequently enough to make the recovery time satisfactory

Media Imaging

N

O

T

E

S

Using linear logging makes it possible to perform media recovery – the ability to recover the state of a damaged MQ object, such as a queue or topic.

To do this, it is necessary to periodically capture images of the objects, so you have something to perform recovery from. Recording images has always been an administrative responsibility – one that is often automated by use of a script. To be recoverable, each object must have an image recorded frequently enough using *rcdmqimg* to make the recovery time satisfactory.

The Impact of taking Media Images

- Writing media images to the log can dramatically impact MQ performance
- Some times are good for taking a media image:
  - When a queue is empty
  - When the system is quiet
  - When the size of the logs required for media recovery is large
  - When a lot of time and/or activity has passed since the last one
- Some times are, therefore, not:
  - When the queue is large or growing
  - When a lot of activity is taking place in the system
  - When an image was just taken, even if the queue is empty
- Careful scheduling of images needed
  - But often not practical – it's very difficult to take images at only the good times

The Impact of taking Media Images

NOTES

Writing media images to the log can dramatically impact MQ performance – and not in a good way. There is guidance provided regarding when you should take media images, and when you should avoid doing so if possible. Some times are good for taking a media image:

- When a queue is empty
- When the system is quiet
- When the size of the logs required for media recovery is large
- When a lot of time and/or activity has passed since the last one

If some times are good for taking a media image, that means there are also times when taking media images is not so good. For example:

- When queue(s) are large or growing
- When the queue manager is busy and there is a lot of persistent message activity taking place in the system
- When an image was just taken, even if the queue is empty

So good judgement is needed to take media images in good situations, and avoid doing so when conditions are less than ideal. Unfortunately, this judgement is often not exercised – it's very difficult to take images at only the good times, since those times are constantly varying. In many cases MQ admins simply automate the taking of media images, so no judgement is used at all.

How the Queue Manager can help

- A queue manager is constantly aware of its state. It knows things like:
  - How many messages are on each queue
  - When and how an object is changed
  - When the queue manager is busy (and not busy)
  - When the last image was taken
- The queue manager can choose the best time to record an image
- But it need guidance:
  - How long is an acceptable time between images?

    or
  - What size to aim for the amount of media recovery data?
- With some target values, coupled with its knowledge of the state of the system, a queue manager could decide for itself the best time to take a media image

How the Queue Manager can help

NOTES

Even the best MQ admin does not have the same knowledge of a queue manager as it has of itself. A queue manager is constantly aware of the state. It knows things like how many messages are on each queue, when the queue manager is busy (and when it is not), when and how an object is changed, when the last image was taken, and so on. So the queue manager is best qualified to decide for itself when the best time would be to record an image.

What a queue manager does not know are what the MQ admins objectives are for recovery. So it needs to be provided with some guidance – some target values that it can combine with what it knows about the state of the system. For example, how long is an acceptable time between images? Or, what size to aim for the amount of media recovery data?

Simply taking images for all objects at once (for example, on the hour) can dramatically impact performance, especially when the system is under heavy persistent load, queues get very deep, etc.

But with some target values, coupled with its knowledge of the state of the system, a queue manager could decide for itself what is a good time to take a media image

New Media Imaging Target Values

- MQ V9.0.2 adds a new QMGR attribute: **IMGSCHED**
  - *MANUAL* means image recording remains an administrative task
  - *AUTO* means the queue manager will handle the recording of images
    - You can still take manual images if you want to

- If *AUTO* specified, two additional QMGR attributes specify the imaging target values:
  - **IMGINTVL**: This is the target frequency for recording media images
    - Default = 60 minutes
    - *OFF* indicates that automatic media images will not be recording on a time interval basis
  - **IMGLOGLN**: This is the target amount of log written after which media images will be taken
    - *OFF* (the default) indicates that automatic media images will not be recorded based on the number of bytes written

New Media Imaging Target Values

N

O

T

E

S

With MQ V9.0.2 a new queue manager attribute was added, to indicate whether automatic media images should be taken by the queue manager - **IMGSCHED**:

- When set to MANUAL, this tells MQ that automatic media images should not be taken - image recording remains an administrative task. But setting this to AUTO means the queue manager will schedule the recording of images for you, based on additional attributes to be discussed next. Note that with IMGSCHED(AUTO), you can still take manual images if you want to.

If IMGSCHED(AUTO) is specified, then two additional queue manager attributes are used to control automatic media imaging:

- **IMGINTVL**: This is the target frequency for recording media images, and is specified in the number of minutes since the last media image. A special value of OFF can be specified to indicate that automatic media images will not be recording on a time interval basis. The default is 60 minutes.
- **IMGLOGLN**: This is the target amount of recovery log written by which the queue manager will automatically record media images, specified in the number of megabytes since the last media image. Here, too, a special value of OFF (the default) can be used to indicate that automatic media images will not be recorded based on the number of bytes written.

Result is Smart Media Imaging

- MQ tried to be clever about taking media images
- When a media image is taken manually, it is of the object in its current state
- When done automatically, several strategies are used
  - When queues to be imaged are often very busy, MQ might take a *partial image* of it
    - This is an image of the queue as it was at a point slightly in the past.
    - Messages put or got to the queue since the recovery point do not have to be written to the image
      - Since recovering the queue would replay all log records and so would get reconciled then
    - Recording the image would be significantly faster using this strategy
      - The trade-off being that replaying would take a little more time.
    - This strategy works best when most messages sit on the queue briefly
      - Applications that use MQ as a database will subvert this
- Best Practice for media images is to take them when queues are empty or nearly so
  - The queue manager watches for such times and may decide to record an image early, at an opportune time
    - It is much easier for the queue manager to spot such times than for an MQ admin

Result is Smart Media Imaging

N

O

T

E

S

Another performance advantage that could be gained by having the queue manager decide when to record media images is for it to use strategies that result in less data being written to the log.

One strategy would be to establish a point of recovery for an object, but not record a full copy of the object at that time in the log. This is known as a *partial* media image. When the queue manager records a partial media image, it does so on behalf of a recovery point which is a little time in the past. If many (or all) of the messages on the queue were put since the recovery point, then these messages do not have to be recorded in the media image, since recovering the queue would replay all log records since the recovery point and so would include all these recently put messages. Similarly, if many or all of the messages that were on the queue at the recovery point have since been gotten off the queue, these messages wouldn't need to be recorded in the partial media image because they aren't on the queue now. So, if the cards are right, the partial media image may contain no data at all and so be very quick to record, even though the queue was never empty. This optimization is most effective when most messages only rest on the queue for a short period of time. If your applications do things like use MQ as a database, using queues as a place to store data long-term, then this strategy won't work, as the queue manager will have no alternative but to record full media images every time.

One of the best times to automatically record a media image is when the queue is completely empty, or almost empty, because then there is little user data to log so the media image written is very small. The queue manager watches for such times and if a media image hasn't been written for a while, but IMGINTVL or IMGLOGLN hasn't expired yet, the queue manager may decide to record a media image anyway because now is a really good time. It is much easier for the queue manager to spot such times than it is likely to be for an MQ admin, so this provides another performance boost to automatic media image recording.

Controlling what is Recoverable

- Media images allow you to recover damaged objects
- But many objects may never need to be recovered
  - It may be less costly to delete and recreate some objects from a saved definition
  - e.g. Topic objects have no associated data – only attributes

- You can now choose whether local queues and other objects are recoverable
  - If you choose for them to be not recoverable then you cannot record images or recover them
  - And, automatic media images will not be taken for such objects
  - But less data will need to be written to the log
  - And those objects can be saved and restored – just by other means (e.g. dmpmqcfg)

Controlling what is Recoverable

NOTES

By taking media images you can recover damaged objects. But many MQ objects may never actually need to be recovered. Topic objects, for example, have no data associated with them other than attributes; for such objects, it can be just as easy to simply recreate them and avoid the overhead of repeatedly capturing images of them.

MQ v9.0.2 gives you the ability to choose whether local queues and other objects are recoverable. If you choose for them to be not recoverable then you cannot use rcdmqimg to record an image of them, or rcrmqobj to recover the object. Neither will automatic media images be taken for unrecoverable objects. So you'll need to use another tool, such as dmpmqcfg, to save the definitions.

By making some MQ objects not recoverable, less data will need to be recorded taking images for the queue manager, saving log space and potentially improving performance.

Controlling Recoverability

- MQ V9.0.2 adds new QMGR attributes to control whether objects are recoverable or not:
- **IMGRCOVO**: Determines whether various object types are recoverable
  - *AUTHINFO*, *CHANNEL*, *CLNTCONN*, *LISTENER*, *NAMELIST*, *PROCESS*, *QALIAS*, *QREMOTE* and *SERVICE* objects
- **IMGRCOVQ**: Specifies the default *IMGRCOVQ* attribute for local and PERMDYN queue objects
  - Both can be set to *YES* or *NO* (Default is *YES*)
- A queue attribute is also defined for local and model queues:
  - **IMGRCOVQ**: Determines whether a local or PERMDYN queue object is recoverable
  - Can be set to *YES*, *NO* or *QMGR* (Default)

Controlling what is Recoverable

NOTES

With MQ V9.0.2, two new queue manager attributes were added, to control whether objects are recoverable or not.

The IMGRCOVO attributes determines whether AUTHINFO, CHANNEL, CLNTCONN, LISTENER, NAMELIST, PROCESS, QALIAS, QREMOTE and SERVICE objects are recoverable from a media image.

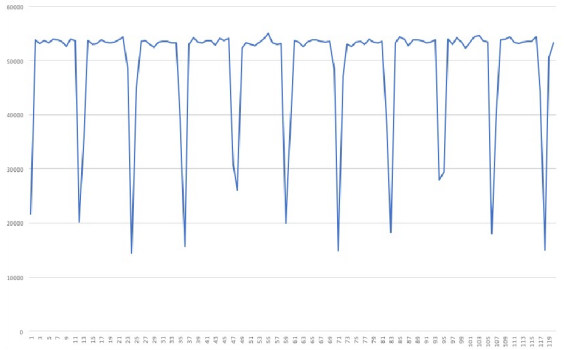
The IMGRCOVQ attribute specifies the default IMGRCOVQ attribute for local and permanent dynamic queue objects with IMGRCOVQ(QMGR)

Both can be set to YES or NO (Default is YES).

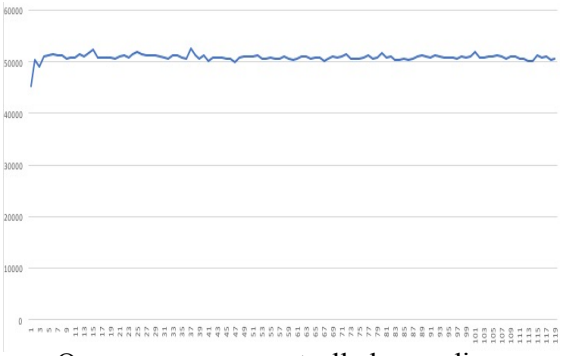
In addition, a queue attribute is also provided that can be specified for local and model queues: IMGRCOVQ determines whether a local or permanent dynamic queue object is recoverable from a media image. This can be set to YES, NO or QMGR (to inherit from the attribute above). The default is QMGR.

Value of Automatic Media Imaging

- Queue manager controlled imaging reduces the impact on other workload during recording
- Charts illustrate this:
  - At top: Manually imaging a deep queue
  - Bottom: Auto imaging of the same queue
- Impact on messaging workload is significantly reduced
  - Same data being recorded in the same span of time
  - But images records interleaved
  - Writing is cleverly scheduled
  - I/O avoidance, etc
- Application impact minimized
  - With no loss of recoverability



Manually controlled recording



Queue manager controlled recording

Value of Automatic Media Imaging

NOTES

These charts show the performance advantage to be gained by allowing the queue manager to record media images automatically based on its knowledge of the workload and the targets you set for *IMGINTVL* and *IMGLOGLN*. One advantage is that media images will gradually get staggered over time. When a media image is requested manually using *rcdmqimg*, the queue manager responds immediately to the request and records the image. When an automatic image is scheduled the queue manager does not need to respond immediately, and will use some discretion regarding when and how to write the image. By writing the image more slowly, recording the image should have less impact on the messaging workload. These charts illustrate this using the example of a very deep queue.

In the first chart, media images are triggered by running *rcdmqimg* roughly every 10 seconds. As the first chart shows, the test is severely impacted by the synchronous nature of manual image recording.

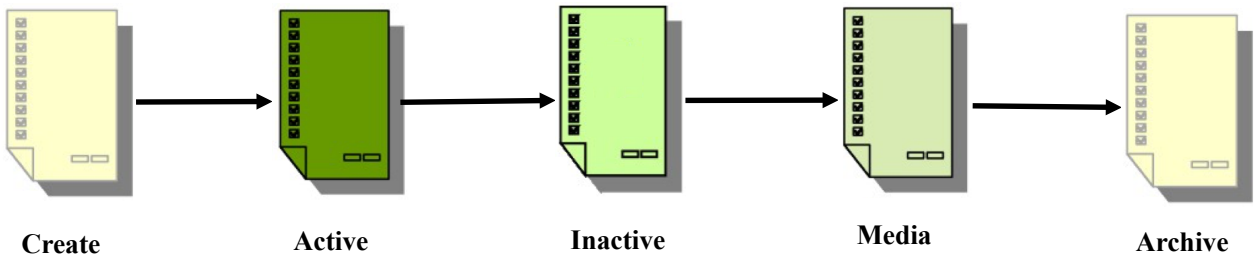
The second chart shows the same test (IMGLOGLN set to 400 to approximate the frequency of image recording in the manual test), with automatic media image recording minimizing the impact by interleaving the image records with the normal log writes. Remember that a Best Practice with taking a media image is to do so when queues are empty or nearly so, because then there is little user data to log, making the media image very small. With automatic media imaging, the queue manager watches for such times, and if a media image hasn't been written for awhile, but IMGINTVL or IMGLOGLN haven’t expired yet, the queue manager may decide to record a media image anyway because now is a really good time. It is much easier for the queue manager to spot such times than it is likely to be for you, so this provides another performance boost to automatic media image recording.

Questions?

Automatic Reuse of Extents

The Life Cycle of a Linear Log Extent

- Created and formatted by the queue manager
- Becomes part of the primary log extents (active)
- Once fully written it becomes (logically) read only and inactive
  - It may still be required for restart recovery
- Eventually it ceases to be required for restart recovery and required only for media recovery
- After this it is no longer needed by the queue manager and the customer can do whatever it likes with it



The Life Cycle of a Linear Log Extent

N

O

T

E

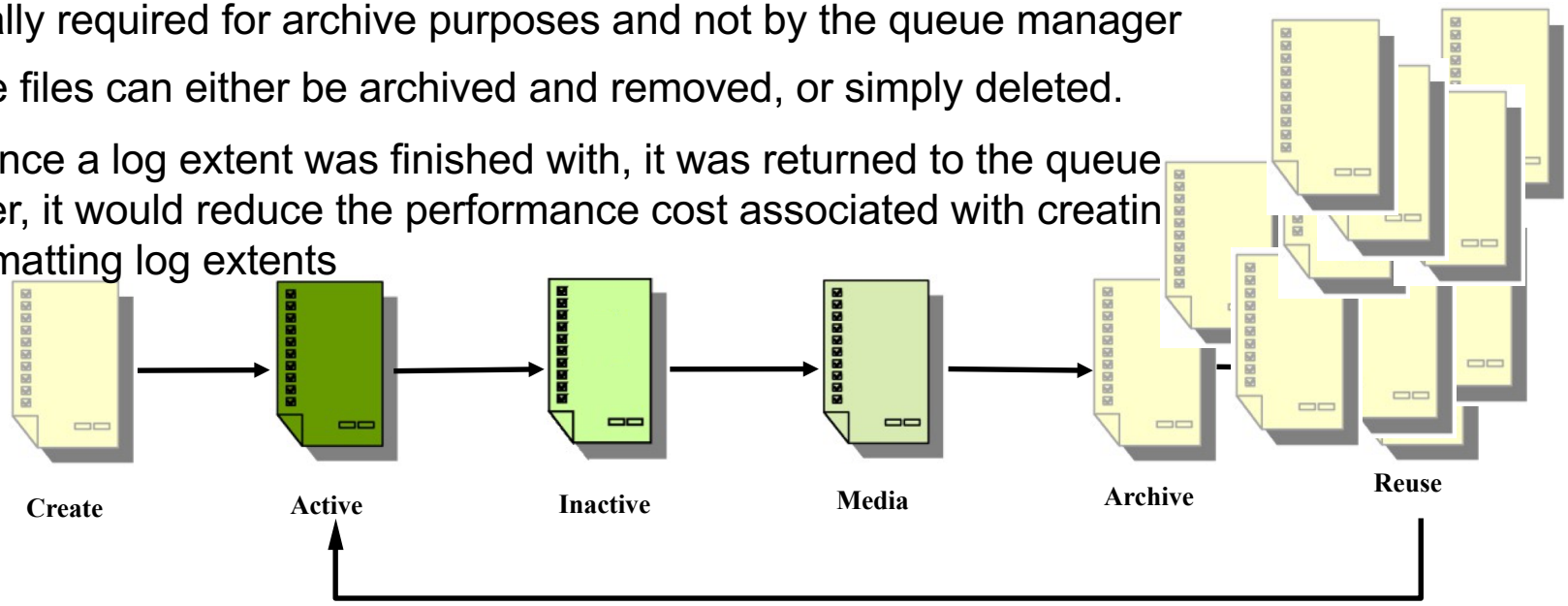
S

With Linear logging, extents are continually created and formatted by the queue manager, an activity that takes resources from the hosting platform and hence away from the queue manager. As new files become available they become part of the primary log extents, and are used by the queue manager to record data needed for recovery. Once a log file is fully written, it becomes inactive in the sense that the queue manager is no longer actively recording to it – however, it may still be required for restart recovery

Eventually it ceases to be required for restart recovery, although it may still be required only for media recovery. But in time, it will no longer be needed even for this purpose – it no serves any purpose as far as the queue manager is concerned, and so can be deleted, or possibly archived if the customer saves these for audit or other purposes. These activities (deleting and possibly archiving) have always been an administrative responsibility – the queue manager does not handle these activities for you.

Reusing Old Log Extents

- It is the responsibility of the customer to ensure that old logs are dealt with in a timely fashion
  - If they are not, the file system will eventually become exhausted
- Once a log file is no longer needed for restart or media recovery it is only potentially required for archive purposes and not by the queue manager
  - These files can either be archived and removed, or simply deleted.
- But if, once a log extent was finished with, it was returned to the queue manager, it would reduce the performance cost associated with creatin and formatting log extents



Reusing Old Log Extents

N

O

T

E

S

With Linear logging, deleting and archiving (if desired) log extents that are no longer of use to the queue manager is the responsibility of the customer. Old logs cannot simply be ignored – they must be dealt with in a timely fashion. If they are not, the file system will eventually become exhausted. Many customers today simply delete these files.

But what if, once a log extent was finished with, the queue manager were to reuse it? It is after all a properly formatted log file – there's data in it on course, but that data is old now and can be overwritten. By feeding these log files back into the front of the log stream, the log creation step can be avoided, reducing the performance cost associated with creating and formatting log extents while at the same time making life a bit easier for the administrator.

New Log Management Modes for Linear Logging

- MQ V9.0.2 introduces three modes for Linear Log Management:
  - **Manual** – Log extents must be manually managed, as they are today
  - **Automatic** – The queue manager will automatically reuse log extents when they are no longer required for restart or media recovery
  - **Archive** – Same as Automatic, except that the queue manager will wait for notification that an extent has been archived before reusing an extent

New Log Management Modes for Linear Logging

NOTES

With MQ V9.0.2, there are now three modes in which Linear logging can be run

- When set to MANUAL, Linear logging works as it has always worked – log files are continually created and used, and whatever happens to them after that is an administrative responsibility.
- If set to AUTOMATIC, the queue manager will automatically reuse log extents when they are no longer required for restart or media recovery
- If ARCHIVE mode is selected, the behavior is the same as Automatic, except that instead of reusing extents immediately once they are no longer needed for media recovery, the queue manager will wait for notification that an extent has been archived (archival being an administrative responsibility as in the past), before reusing an extent.

Setting Log Management

- When you create your queue manager you can specify new options:
  - **-lla** Indicates that Automatic log management will be used

```
# crtmqm -lla AUTO_LOG
IBM MQ queue manager created.
…
```

  - **-lln** Indicates that Archive log management will be used

```
# crtmqm -lln ARCHIVE_LOG
IBM MQ queue manager created.
…
```

  - **-ll** Indicates that Manual log management will be used, as before

- If you have an existing queue manager you can edit qm.ini and can simply add one of the following lines to the Log stanza:
  - LogManagement=Automatic or
  - LogManagement=Archive

qm.ini (for example)

```
Log:
    LogManagement=Automatic
```

Setting Log Management

NOTES

With MQ V9.0.2, there are now three options to choose from when creating a queue manager that will use Linear logging:

- **-lla** – Indicates that Automatic log management is to be used.
- **-lln** – Indicates that Archive log management is to be used.
- **-ll** – Indicates that Manual log management is to be used.

For existing queue managers, you can alter the log management mode using a new attribute that can be specified in the Log stanza of the qm.ini file – *LogManagement=Automatic|Archive*

Archive Log Management

- When using Archive log management you will need to notify the queue manager when a log extent has been archived
- This is done using the new SET LOG command, or its PCF equivalent. For example:

```
SET LOG ARCHIVED(S0000011.LOG)
```

- This indicates that log extent S0000011.LOG is no longer required for archive purposes and once it's no longer required for restart or media recovery it is eligible to be reused
- Logger events are extended to include the oldest extent, ARCHLOG, still needing archiving
- If ARCHLOG gets out of step RESET QMGR TYPE(ARCHLOG) can be used to reset it

```
------------------------------------------------------------
Event Message Received
Command    :Logger Event Command  CompCode :0
Reason     :Logger Status
------------------------------------------------------------
Queue Manager Name     QM1
Current Log Extent     S0000011.LOG
Restart Log Extent     S0000011.LOG
Media Log Extent       S0000009.LOG
Log Path   /var/mqm/log/QM1/active/
Archive Log Extent     S0000003.LOG
------------------------------------------------------------
```

Archive Log Management

NOTES

If you choose to use Archive log management mode, the actual archiving of the old log files continues to be an administrative responsibility. But if you are doing archiving today already, you should be able to use the same process, with a small modification – once the file is archived, you will need to notify the queue manager so that it will know to it is safe to reuse the log file. This notification can be done using the new SET LOG command (or its PCF equivalent). For example:

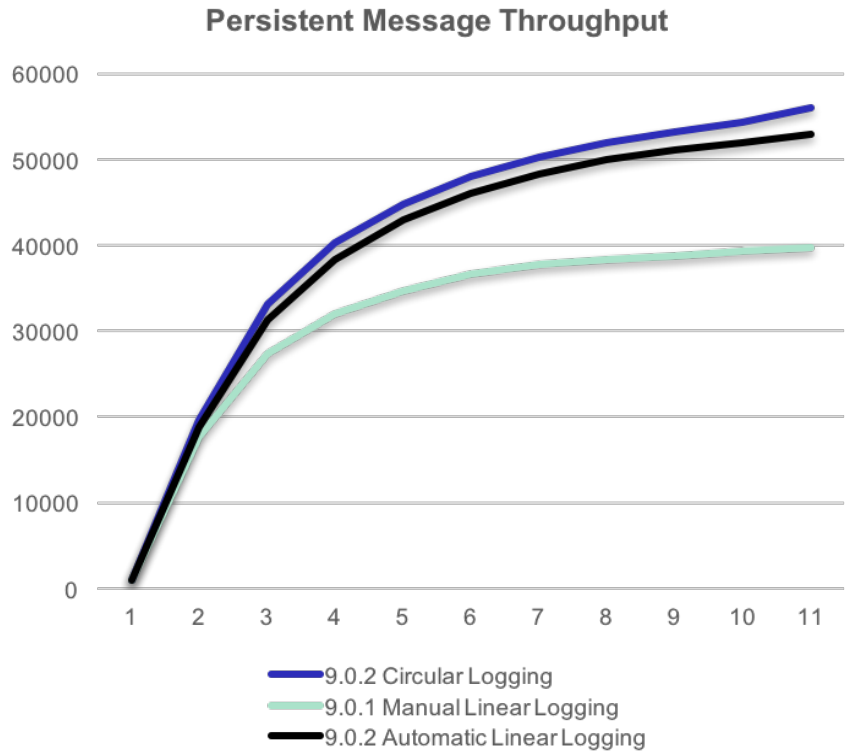
```
SET LOG ARCHIVED(S0000011.LOG)
```

This would indicate that log extent S0000011.LOG is no longer required for archive purposes and once it's no longer required for restart or media recovery it is eligible to be reused.

Logger events are extended to include the oldest extent, ARCHLOG, still needing archiving. Should ARCHLOG get out of step, RESET QMGR TYPE(ARCHLOG) can be used to reset it.

Value of Automatic Reuse of Extents

- Value of reduced administrative burden is obvious
- Value of log reuse shown in this chart:
  - Circular at the top
  - Manual Linear at the bottom
  - Auto Reuse impact on throughput very close to that of Circular



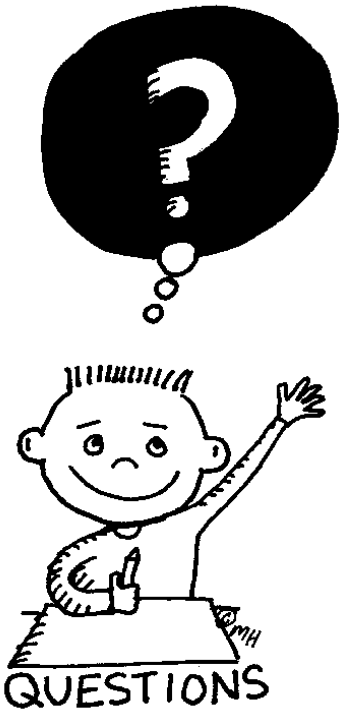
Value of Automatic Reuse of Extents

NOTES

This chart show the performance advantage to be gained by allowing the queue manager to reuse log extents.

Without automatic log management, the queue manager will continue to create and format new linear log files continuously, which impacts throughput (bottom graph). With V9.0.2, and making use of log file reuse, the queue manager can avoid having to continually format new extents, instead reusing old log files as the media recovery point is moved forward in the log, resulting in significantly improved performance (center graph). In fact, the V9.0.2 test with automatic log management shows throughput very close to that of circular logging (top graph).

Questions?



Other Changes…

Reducing Log Extents

- The queue manager will attempt to keep as many reusable log extents as needed
  - Goal is to avoid having to create new log extents
  - How many is related to workload as well as how often media images are taken
- If you feel that too much data is being kept, a new command is provided:
  - RESET QMGR TYPE(REDUCELOG)
  - Can be used to remove some or all reusable extents
- This command can also be used for circular logging
  - To release secondary log extents following a large spike in activity

Reducing Log Extents

N

O

T

E

S

While with automatic log management the queue manager will attempt to keep as many reusable log extents as necessary, to avoid having to create new log extents. How many are retained is related both to workload as well as to how often media images are taken.

If over time you feel that too much data is being kept, a new command is provided - RESET QMGR TYPE(REDUCELOG) – which can be used to remove some or all reusable extents.

This command can also be used for circular logging to reduce the number of secondary log extents following a large spike in activity. Note that this will not delete any secondary extents that are still needed for restart recovery – but it can result in log extents being freed more quickly then they otherwise would be.

DISPLAY QMSTATUS LOG (1)

- Both runmqsc and PCF now include extra information on DISPLAY QMSTATUS about the state of the log
  - These have been grouped together using DISPLAY QMSTATUS LOG

```
DISPLAY QMSTATUS LOG
     2 : DISPLAY QMSTATUS LOG
AMQ8705: Display Queue Manager Status Details.
   QMNAME(QM1)                                   STATUS(RUNNING)
   ARCHLOG(S0000011.LOG)                         ARCHSZ(16)
   CURRLOG(S0000018.LOG)                         LOGINUSE(29)
   LOGPATH(/var/mqm/log/QM1/active/)             MEDIALOG(S0000009.LOG)
   LOGUTIL(25)                                   RECLOG(S0000018.LOG)
   MEDIASZ(143)                                  REUSESZ(49)
   RECSZ(1)
```

Oldest extent needing archiving

MBs of data just waiting to be archived

MBs of data needed for media recovery

MBs of data needed for restart recovery

MBs of data available to be reused

DISPLAY QMSTATUS LOG (1)

N

O

T

E

S

A new runmqsc command, DISPLAY QMSTATUS LOG (as well as a PCF equivalent) is provided to aid in monitoring as well as assisting MQ admins in creating better log configurations. Information returned includes:

- The name of the oldest log file that requires archiving
- The log space needed for restart recovery (in MB)
- The log space needed for media recovery (in MB)
- The amount of log space waiting to be reused (in MB)
- The amount of log data waiting to be archived (in MB)

DISPLAY QMSTATUS LOG (2)

- Two new statistics are included related to the utilization of the log
- These are useful for both Circular and Linear logging

```
DISPLAY QMSTATUS LOG
     2 : DISPLAY QMSTATUS LOG
AMQ8705: Display Queue Manager Status Details.
   QMNAME(QM1)                              STATUS(RUNNING)
   ARCHLOG(S0000011.LOG)                    ARCHSZ(16)
   CURRLOG(S0000018.LOG)                    LOGINUSE(29)
   LOGPATH(/var/mqm/log/QM1/active/)        MEDIALOG(S0000009.LOG)
   LOGUTIL(25)                              RECLOG(S0000018.LOG)
   MEDIASZ(143)                             REUSESZ(49)
   RECSZ(1)
```

LOGUTIL is a percentage estimate of how well the queue manager workload is contained within the primary log space

LOGINUSE is the percentage of the primary log space in use for restart recovery at this point in time

DISPLAY QMSTATUS LOG (2)

NOTES

Two new statistics are included related to the utilization of the log.

- LOGINUSE is the percentage of the primary log space in use for restart recovery at this point in time. A value of 100 or greater indicates the queue manager may have allocated and be using secondary log files, probably due to long-running transactions at this point in time
- LOGUTIL is a percentage estimate of how well the queue manager workload is contained within the primary log space. If the value is consistently above 100 you may wish to investigate whether there are long-lived transactions or if the number of primary files is not sufficient for the workload. If the utilization continues to rise eventually requests for most further operations requiring log activity will be refused with an MQRC_RESOURCE_PROBLEM return code being returned to the application and transactions may be backed out

Log Record Timestamps

- The *dmpmqlog* tool can be used to print out the contents of the recovery log
- With v9.0.2 a timestamp has been added to each log record, and will be printed out by the tool
- The timestamp will be in sequence per object, but not necessarily across objects
- Here is an example of the output:

```
LOG RECORD - LSN <0:0:4619:20>
**********
HLG Header: lrecsize 240, version 1, rmid 0, eyecatcher HLRH
Creation Time: 2017-01-21 12:06:45.852 GMT Standard Time (UTC +0)
```

System topics on distributed queue managers

Distributed queue manager information is published to a range of system topic strings

$SYS/MQ/INFO/QMGR/….

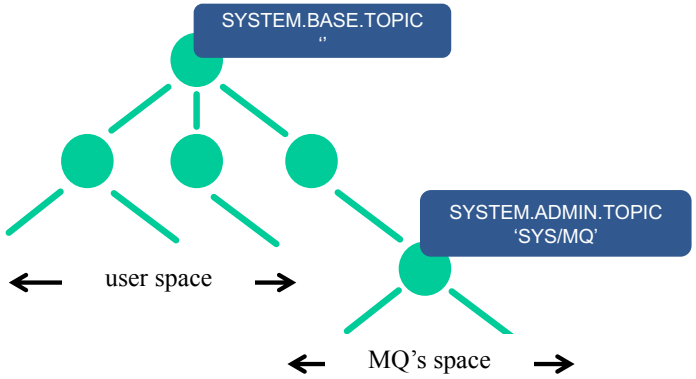
Authorised subscriptions receive their own stream of publications based on the topic string

Administrative subscriptions

e.g. For information to be continually sent to defined queues

Application subscriptions

e.g. To dynamically listen to information as required



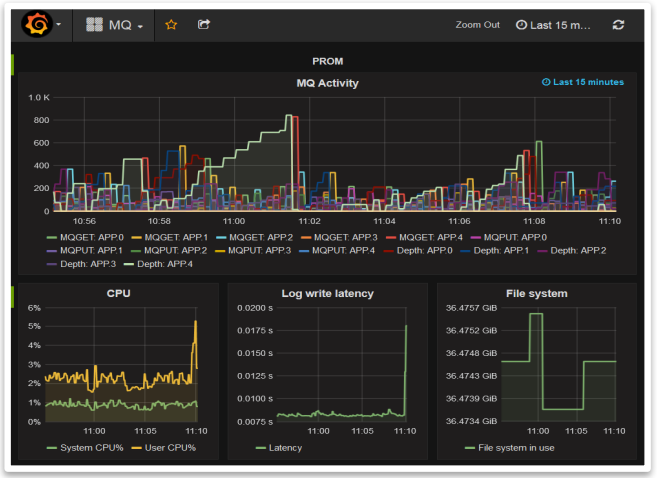
Unlocks system level information for MQ administrators and DevOps teams

Administrators can grant access to subsets of the data, pertinent to different application teams

Log Monitoring Information (1)

- System topics are new in V9 and allow you to subscribe to monitoring information periodically generated by the queue manager

- This includes information pertaining to the Recovery Log
  - How efficiently space in the log is being used
  - A measure of the latency the logger is seeing when writing to the log



Or use the data to feed your own dashboard
See github.com/ibm-messaging/mq-golang

```
Log - Physical bytes written
Log - Logical bytes written
Log - Write Latency
```

Monitoring Information (2)

- With v9.0.2 the Log topic includes information related to the new function introduced in v9.0.2:

```
Log - current primary space in use 29.63%
Log - workload primary space utilization 25.64%
Log - bytes required for media recovery 143MB
Log - bytes occupied by reusable extents 49MB
Log - bytes occupied by extents waiting to be archived 16MB
```

Monitoring Information

NOTES

System topics were introduced in IBM MQ V9. These enable administrators to subscribe to monitoring information that is periodically generated by the queue manager. The *Log* topic includes information related to the Recovery Log and the Logger component, and includes data related to the new function introduced in v9.0.2:

- Current primary space in use
- Workload primary space utilization
- Bytes required for media recovery
- Bytes occupied by reusable extents
- Bytes occupied by extents waiting to be archived

For a further explanation of this new data and how you can make use of it, see the developerWorks article here:

https://www.ibm.com/developerworks/community/blogs/messaging/entry/Statistics_published_to_the_system_topic_in_MQ_v9

Summary

- The Recovery Log a critical component of the queue manager
  - Has been enhanced over time to keep up with modern workloads

- Significant new function added in v9.0.2 related to log management
  - Statistics provide greater insight into logging behavior
  - Automatic Media Imaging can improve performance
  - Automatic Log Management can improve performance and reduce administrative burden

Questions & Answers