Mysteries of the IBM MQ Distributed Logger

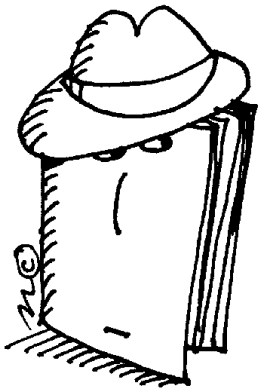
Christopher Frank

IBM Systems Middleware

Agenda

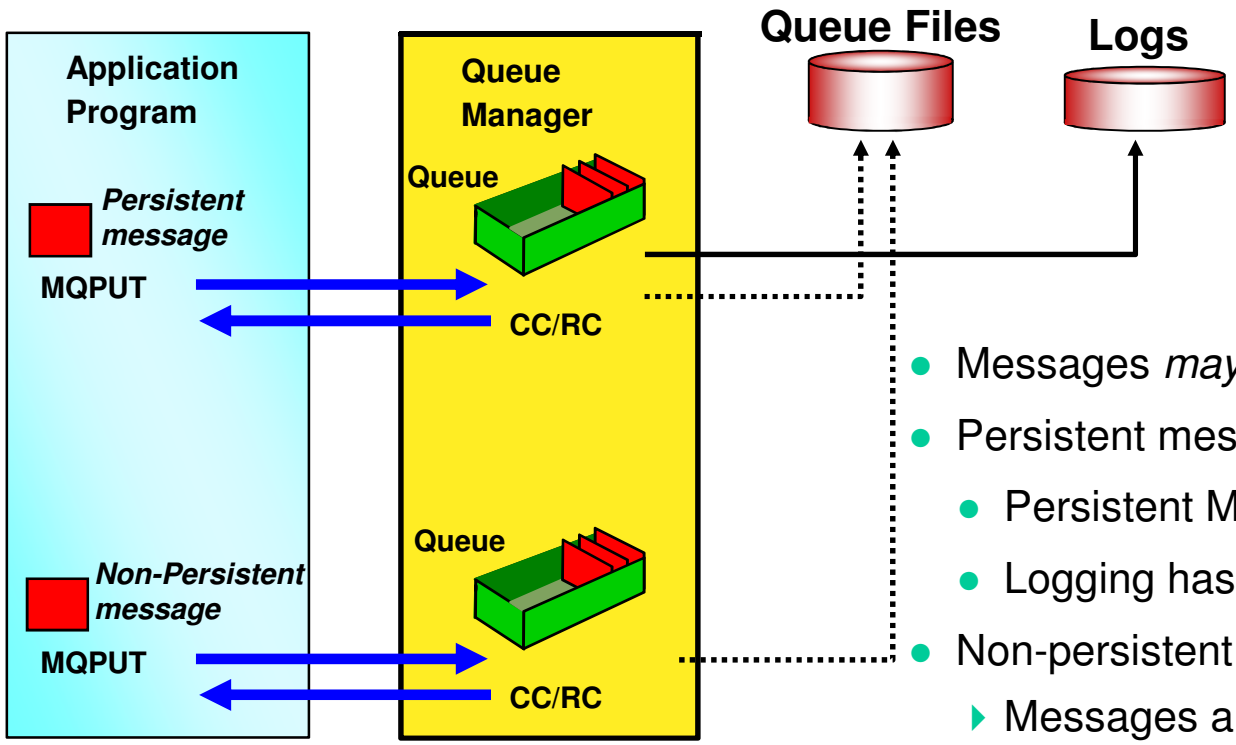
- Mysteries and Decisions
- Circular and Linear
- Understanding Logging Parameters
- Log Write Integrity!
- Log Operation
- AMQ7469!
- Behind the Curtain
- What's New
- Wrap-Up

Why "Mysteries"?



- **Most people understand the role of the MQ logger**
  - But not necessarily how it fulfills that role
  - Or how best to help it fulfill that role
- **It can be hard to tell when the Logger is "happy"**
  - And when it's unhappy, for that matter!
  - But it's very important to understand how best to configure the logger
    - …and how to tell if the configuration is not meeting your objectives
- **MQ Logger essential for persisting messages**
  - But often taken for granted
  - It just "works"…
  - …except when it doesn't…
    - …or at least appears not to…
- **So we'll explore some of these "Mysteries"**
  - What do the logging parameters really mean?
  - What considerations do I use when deciding how to set them?
  - How to tell when the logger is behaving suboptimally
  - And how to tell when to look outside of MQ to improve logger performance

How are Messages Persisted?



- Messages *may* be written to queue files
- Persistent messages are *logged*
  - Persistent Messages are *always* recoverable
  - Logging has implication on performance
- Non-persistent Messages have 2 COS:
  - Messages are retained while Qmgr active
  - Messages can survive a normal shutdown and restart of the Qmgr
    - NPMCLASS(HIGH) on queue
  - NP messages never written to Recovery Log

What are Recovery Logs?

- **For each persistent (Read: Recoverable) update in MQ a log record is written**
  - The log record describes the persistent update
  - Enables commit/rollback of transacted messages
  - Makes recovery of messages after a failure possible

- **Log files are written to sequentially**
  - Sequential I/O is much quicker than random
  - Single point of writing rather than many

- **Under normal conditions the log is only written to**
  - MQ *only* reads log records:
    - During restart
    - When explicitly requested to recover a damaged object (**rcrmqobj**)

What do the Recovery Logs Contain?

- **Record of Transaction Activity**
  - Persistent messages
    - Non-persistent messages are never written to the log
  - Internal data about queue manager objects
  - Persistent channel status
- **MQ Uses a "Write-Ahead" Strategy**
  - Meaning the log is always more up-to-date than the actual data
  - Log and actual data are reconciled during strmqm
- **Recovery logs consist of a collection of files**
  - Three or more files of log data
    - S0000000.LOG – S9999999.LOG
  - Log control file
    - amqhlctl.lfh (in log directory)
  - Checkpoint file
    - amqalchk.fil (in qmgr directory)

Decisions, Decisions…

Logging Parameters

- **Several MQ parameters at your disposal**
  - Used to control the type and behavior of logging
- **Use (or misuse) of these can have a significant effect on performance**
  - For good or for bad
  - And when the latter, this can be difficult to diagnose
- **Some can only be set at the time a queue manager is created**
  - Type of logging
  - Size of log files
  - Log file location
- **Those that are changeable always require a queue manager restart**
  - Number of Primary log files
  - Number of Secondary log files
  - Size of log buffer

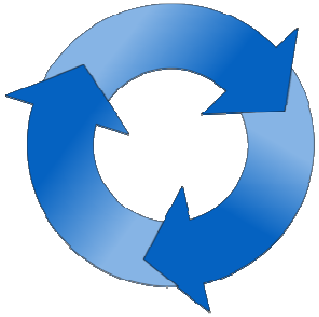
Default Logging Configuration

- **Logging values used for a specific queue manager**
  - Held in the *Log* stanza of the qm.ini file for that queue manager

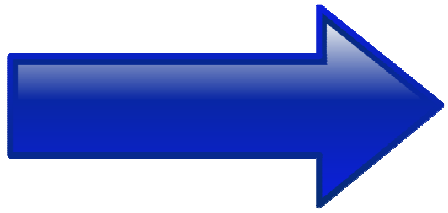
```
Log:
   LogPrimaryFiles=3
   LogSecondaryFiles=2
   LogFilePages=4096
   LogType=CIRCULAR
   LogBufferPages=0
   LogDefaultPath=/var/mqm/log
```

- **In practice this results in:**
  - A log buffer size of 2MB
  - A log file size of 16MB
  - A maximum active log of 80MB
- **You very often want to use different values**
  - So we'll be discussing these in detail

The Great Debate…



Circular

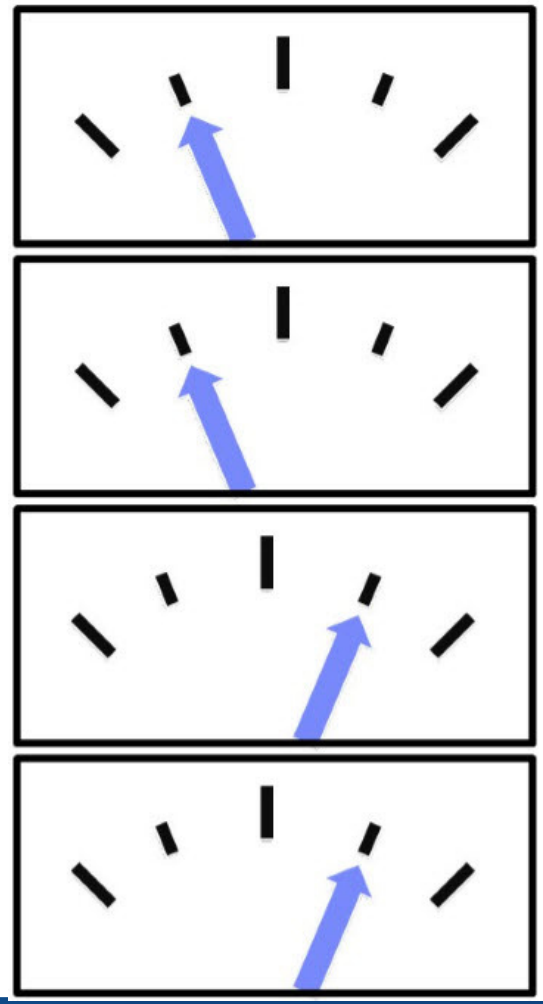


Linear

Two Types of Logging Available (1)

Circular and Linear

- Specified at queue manager creation
- Each has advantages
- Each involves trade-offs

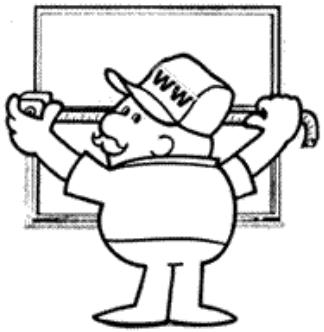


Decisions, Decisions…

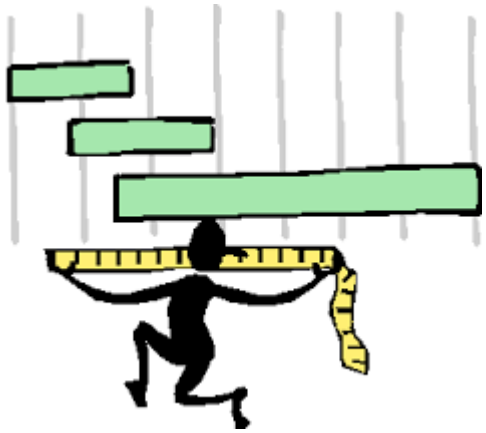
	Circular	*Linear*
> Recovery	Can reconcile in-flight UOWs that were incomplete	Same. And can also recover damaged queues
> Performance	Minimum overhead. Logs allocated once and reused	Logs must be allocated on an ongoing basis, causing I/O contention.
> Admin	Basically no administrative effort required during normal operations.	Administrators must ensure inactive log files deleted or archived
> Risk	Loss of a queue file means loss of all messages on that queue.	A normally running queue manager will eventually exhaust available disk space if log files are not managed regularly

Reference: http://www.ibm.com/developerworks/websphere/techjournal/0904_mismes/0904_mismes.html

Numbers and Sizes…



Log Buffer – How Big?

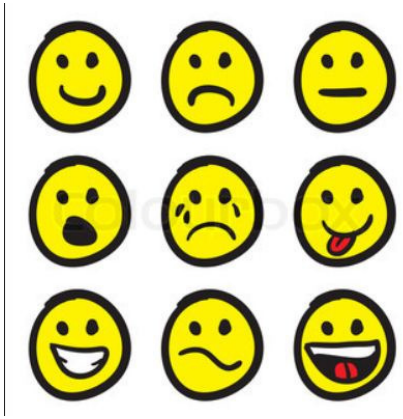


- **Memory space MQ uses to buffer log writes**
  - Not specified on the crtmqm command
    - Value can be specified on qm.ini
      - LogBufferPages parameter of Log stanza
- **Specified as number of 4KB pages. Values:**
  - Minimum: 18 (72KB)
  - Default: 0 (which really means “512”, or 2MB)
  - Maximum: 4096 (16MB)
- **Default is often used, but might be insufficient**
  - What are your typical message sizes?
  - What kind of volume do you expect?
  - How many concurrent transactions (esp. Putters) do you expect?
- **Bigger is usually better**
  - If undersized, can impose a significant performance penalty
  - If oversized…who cares?

Log Files – How Big?

- **Specified as a multiple of 4KB. Values:**
  - Default: 4096 (16MB)
  - Minimum: Unix: 64 (256KB) – Windows: 32 (128KB)
  - Maximum: 65535 (256MB)
- **Should give careful thought to this**
  - What are your typical message sizes?
  - What kind of volume do you expect?
  - How long do transactions typically run?
  - How much headroom needed for future growth?
- **Larger file size generally better**
  - Less frequent log file switches – slight performance gain
    - Linear – More efficient to format/archive 1 large file vs many
  - Allows for larger log overall if needed in future
    - Since log file size cannot be altered after queue manager creation

Primary Logs – How Many?

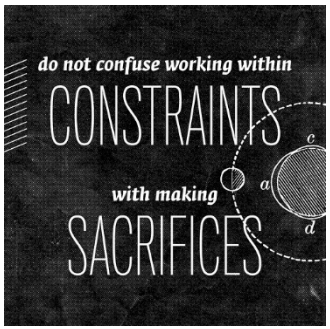


- **Specifies the initial, minimum number of log files**
  - These are allocated when the queue manager is created
- **Values:**
  - Minimum: 2
  - Default: 3
  - Maximum: 510 (Unix) or 254 (Windows)
- **Should give careful thought to this**
  - What are your typical message sizes?
  - What kind of volume do you expect?
  - How long do transactions typically run?
    - This is especially important!
  - How much headroom needed for future growth?
- **Can be altered after queue manager creation**
  - LogPrimaryFiles parameter of Log stanza in qm.ini
  - But note that change will not necessarily take effect immediately

Secondary Log Files

- **Specifies number of optional, “spare” log files**
  - To be created should the primary allocation become full
- **Secondaries too warrant careful thought**
  - Do you have occasional spikes in volume?
  - How long do transactions typically run?
    - This is especially important!
  - How much headroom needed for future growth?
- **Values:**
  - Minimum: 1
  - Default: 2
  - Maximum: 509 (Unix) or 253 (Windows)
- **Can be altered after queue manager creation**
  - LogSecondaryFiles parameter of Log stanza in qm.ini
  - Unlike primaries, secondaries will be freed if no longer needed
    - Though this will not happen immediately

Log File Constraints



- **You cannot have as many log files as the numbers imply!**
- **There is a limit on the number of *active* log files**
  - Never fewer then 3
  - Never more than:
    - Unix: 511
    - Windows: 255
- **Why does MQ let you specify more log files than you can use?**
  - To allow you to decide what the mix of primaries and secondaries should be
  - It's never a good idea to have a very large number of logs
  - Better to have fewer, but larger, logs
    - Better performance, room for growth
- **This maximum of 511/255 active log files is a *key constraint* for an active queue manager**
  - On Windows, the largest active log in total cannot exceed 64GB
  - On Unix, the largest active log in total cannot exceed 128GB
  - The good news is, it should never have to come anywhere close to that size

Log Write Integrity…

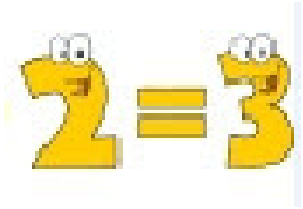
Log Write Integrity

- Specifies approach logger uses to ensure data integrity
- Needed because file systems can vary in terms of integrity
- Three values:
  - SingleWrite
  - DoubleWrite
  - TripleWrite
- Can be altered after queue manager creation
- Frequently misunderstood
- We'll look at each in detail

What Exactly is SingleWrite?

- **Can occasionally provide improved performance**
  - But in practice only with low-concurrency workloads
  - And only when logger writes partial pages
    - Which since V7.1 the logger tries very hard not to do!
- **But comes with a huge caveat!**
  - Can only be used if the file system / storage device provides an *absolute guarantee* that writes will be done atomically
    - Meaning if a write fails for **any** reason the data is guaranteed to be in only one of two possible states: Either the before image, or the after image
- **Recommendation: Do not use SingleWrite unless the file system / storage device can provide the above-mentioned *absolute guarantee***
  - Guarantee must apply to the entire I/O stack – not just the device
  - Even then, can you be sure some change to the underlying software or hardware will not negate such a guarantee in the future?
  - Safest bet is to avoid the use of SingleWrite – you'll sleep easier at night

What Exactly is DoubleWrite?



- **Can still be used but deprecated**
  - Long, long ago, DoubleWrite was the alternative to SingleWrite
  - For when the atomicity of writing 4KB pages to the MQ recovery log could not be assured
- **Unfortunately, there was a problem with DoubleWrite**
  - A small but real potential data integrity issue with the approach was uncovered
  - So the TripleWrite algorithm was implemented to overcome that issue
- **DoubleWrite no longer serves any useful purpose**
  - If specified it's treated the same as TripleWrite (since V7.1)

What Exactly is TripleWrite?

- **The safest option to use in all cases**
  - And given that, why would you want to use any other?
- **TripleWrite does *not* mean MQ writes *all* log data three times!**
  - What it does is address a potential problem that may exists when writing partial pages
- **When writing a partial page:**
  - The logger may subsequently need to add data to the partial page
    - It could simply overwrite the page with both previous and new data
      - But if a write error occurred, the original data could be lost if the write was not atomic
  - To add data to the partial page safely, the logger will:
    - First write the new page to a different location
    - If that succeeds, it will then overwrite the original log file page with the "new" page
  - Since the logger knows which writes succeeded, it will know which log file page to use in a recovery situation
- **Current releases of MQ try very hard not to write partial pages**
  - So the performance cost of TripleWrite in most cases is negligible
  - But the integrity cost of not using it can be substantial

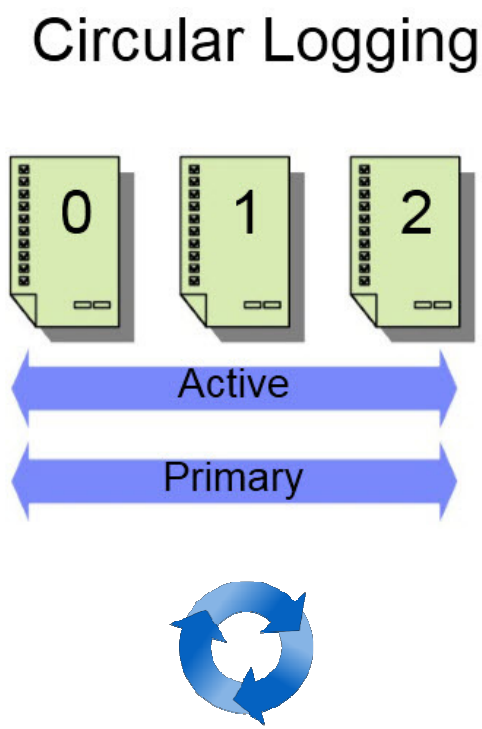
Log Usage…

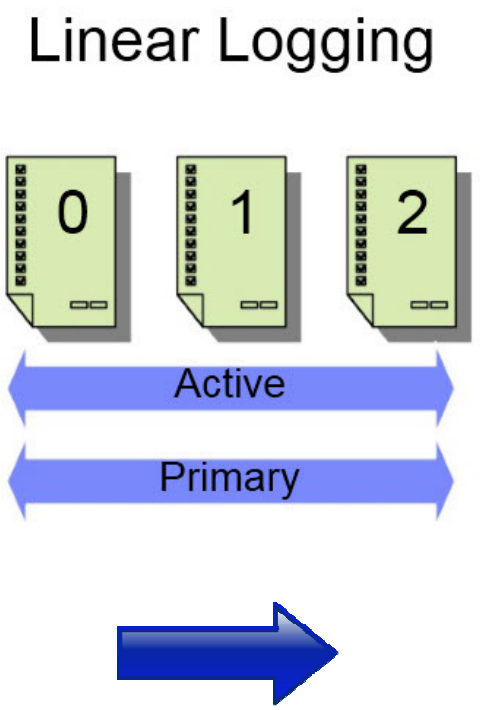
Side-by-Side Comparison

- **The following slides compare Circular and Linear logging in action**
- **Intent is to illustrate the following:**
  - MQ usage of Primary and Secondary log files
  - Active log files – Those needed for recovery purposes
  - Inactive log files – Those no longer needed for recovery purposes

Log Files after initial start of Queue Manager

- **Assume Log Defaults were used**
  - LogPrimaryFiles=3, LogSecondaryFiles=2

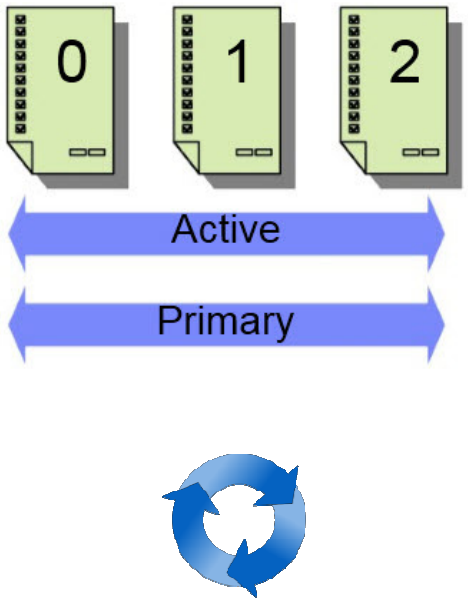




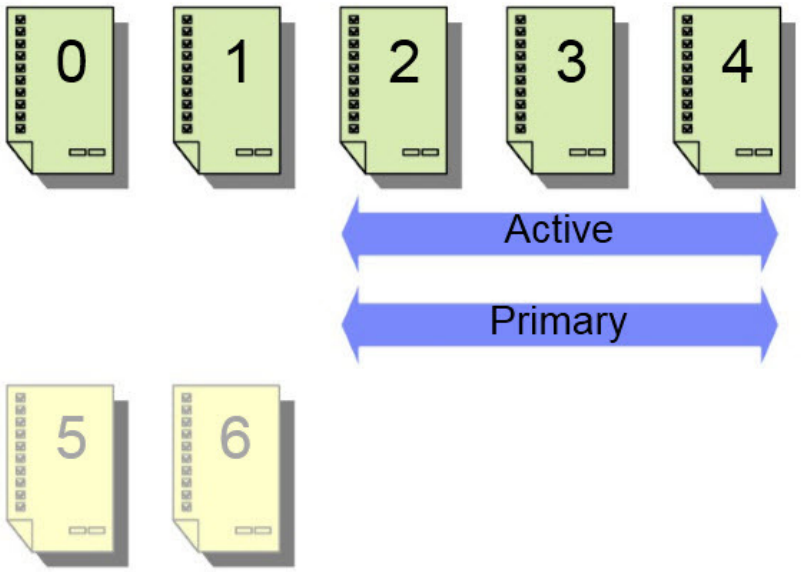
Working with Primary Log Files

- Circular – Wraps when active log full
- Linear – Marks oldest active file as inactive, and allocated a new file

Circular Logging



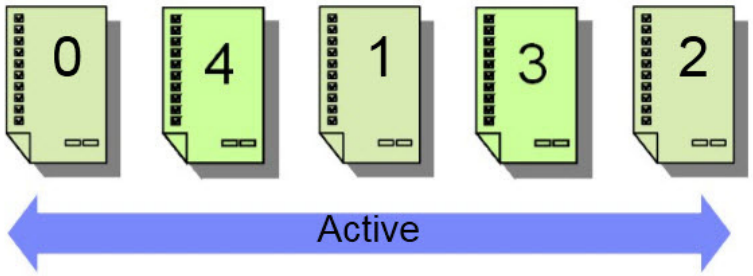
Linear Logging



Expanding to Secondary Log Files

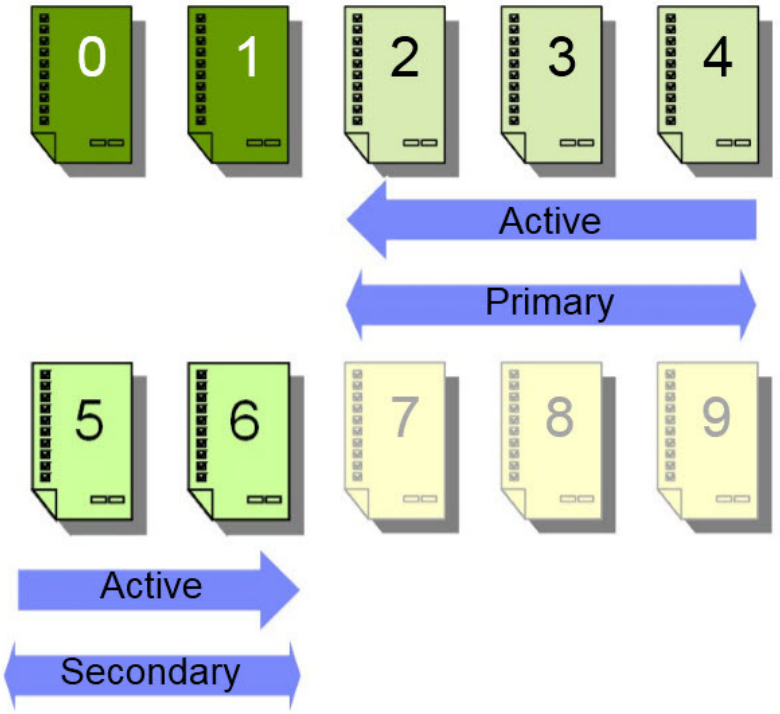
- **Circular – Active log expanded by inserting secondaries into log ring**
- **Linear – Active log "stretched" by adding secondaries onto the end**

Circular Logging



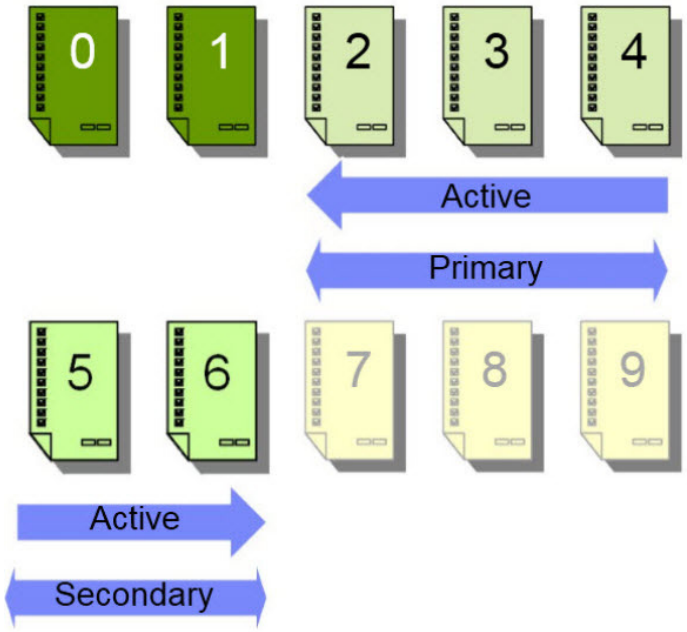
Note: The addition of the secondary logs into the active set depends on the location of the current write point in the active log set (called the head)

Linear Logging



Linear Logging Inactive Files

- **What about log file 0 and 1?**
- **Although not required for restart, they may be required for full recovery**
- **Messages logged periodically to identify:**
  - Oldest file needed for restart recovery
    - Oldest active log file - S0000002.LOG
  - Oldest file needed for media recovery
    - Oldest media image – S0000000.LOG
- **Can also request this via mqsc**
  - DISPLAY QMSTATUS ALL



- **Fundamental difference between linear and circular logging!**

AMQ7467: The oldest log file required to start queue manager QM1 is S0000002.LOG

AMQ7468: The oldest log file required to perform media recovery of queue manager QM1 is S0000000.LOG

What are Checkpoints?

- **Synchronize queue manager data and log files**
  - Mark a point of consistency from which log records can be discarded
  - Frequent checkpointing makes recovery quicker
- **When are checkpoints taken?**
  - When endmqm or rcdmqimg commands issued
  - Also after either:
    - Every a defined number of recoverable log operations
    - Or every 30 mins if at least 100 recoverable operations have occurred
- **You can tune checkpointing**
  - CheckPointLogRecdMax
    - A checkpoint is taken after this many logged operations (default: 10000 in 7.1/7.5, 50000 in 8.0)
  - CheckPointWaitLen
    - Default: 30 minutes. Minimum: 5 minutes. Maximum: 60 minutes
  - CheckPointLogRecdMin
    - Checkpoint taken every *CheckPointWaitLen* only if this many logged operations have occurred. Default: 100. Minimum: 0. Maximum: 2000
- **Note:** Not advisable to use these unless required
  - Doing so will have an impact on the performance – perhaps not the one you intend!

The Active Log and Long-Running UOWs

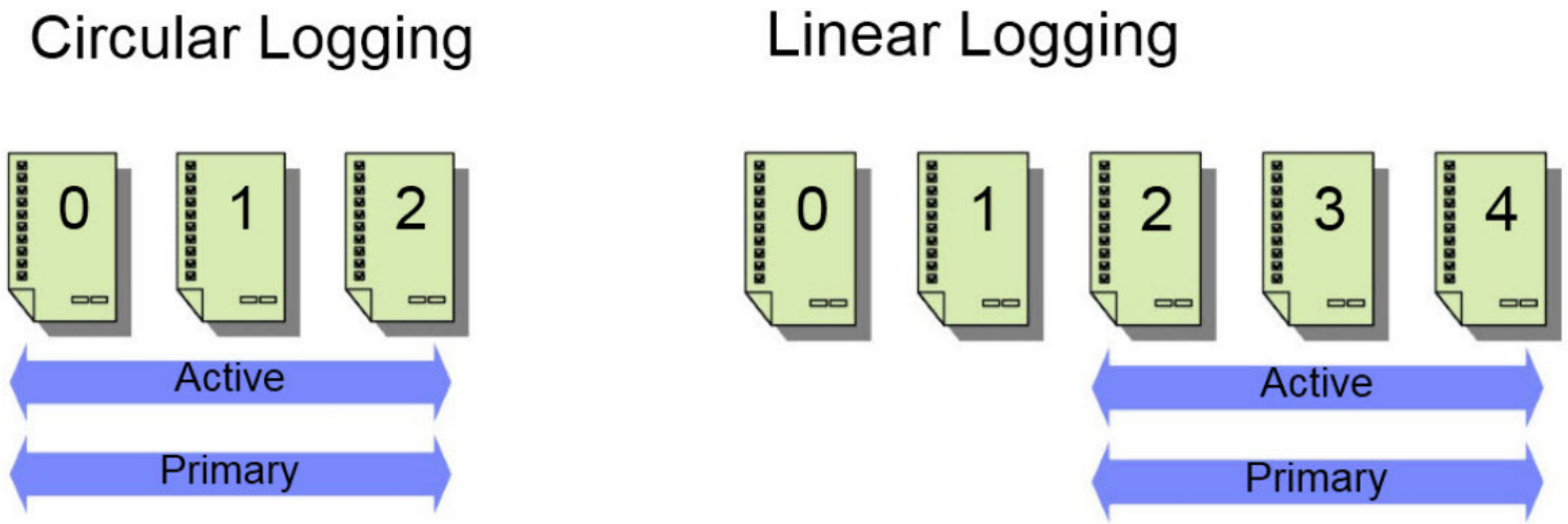
- **The Active Log has a further constraint**
  - An running UOW cannot span the entire active log! Why?
- **Logger holds a portion of the potential active log "in reserve"**
  - ~20% of the active log space is held in reserve
  - This is to allow a cushion for the logger to take checkpoints
  - Remaining 80% represents the maximum "distance" between the head and the tail of the log
- **If encroached on…**
  - The logger will abort ("roll back") the longest-running UOW
  - Better that than risk a stall if log space is exhausted
  - More than one UOW may be rolled back (or "rolled forward) if necessary
  - When this happens you will see one or more AMQ7469 errors
  - Easy to demonstrate with amqsblst ("MQ Blast") sample program

Dealing with Long-Running Transactions

- **With both circular and linear logging, transactions cannot be infinite**
  - Entire transaction must be contained in the active log
  - Reasons differ but result is the same
- **Active log must be sufficiently large to contain all active transactions**
  - Can be addressed by increasing the number of log files
  - But this is not necessarily the right thing to do
- **MQ must take action if this situation arises**
  - Reported in the queue manager error logs. Can take two forms:
    - AMQ7469: Transactions rolled back to release log space
    - AMQ7469: Transactions rolled forward to release log space
- **What are these messages saying?**

AMQ7469 (Transaction Rolled Back) - Why?

- **In-flight transactions must be held in the active log only**
  - Active log has a finite size
  - Because MQ cannot arbitrarily commit a transaction, it must be rolled back
- **Why does this happen?**
  - Active log is too small for current workload
  - One or more applications may be poorly designed



AMQ7469 (Transaction Rolled Forward) - Why?

- **A transaction may be complete ("Prepared") but not committed**
  - Could be waiting for the "commit" flow from an external TM
  - MQ could roll it back, but that could lead to problems
  - Besides, it's not necessary
    - As the transaction is prepared there will be no further additions to it
  - So MQ "rolls it forward", maintaining its state in the active log
- **Why does this happen?**
  - Usually not a log space issue
  - Could be an in-doubt transaction, or an issue with an external TM

Behind the Curtain…

Operational View of the Logger

- **You may want better insight into how the Logger is operating**
  - Not easy to come by
  - Although this is improving
- **Service aid "amqldmpa" allows a peek behind the curtain**
  - Not easy to interpret
  - Much of what's reported is not useful outside L3/Development
  - But we'll look at some useful tidbits that can prove insightful
- **Realize this is undocumented for a reason**
  - What is reported can change at any time
  - Examples that follow are from MQ V8
  - But most of what we'll explore can be found in prior releases
- **MQ Appliance makes this information easier to obtain**
  - Its monitoring interface provides access
  - Same may be in the cards for Installable MQ in the future

How to tell if you are using Secondary Logs?

- **You may want better insight into log file usage**
  - You may see messages that your log is almost full
  - You may see messages that transaction are being rolled back
  - You may want to more intelligently configure primary and secondary logs
    - Especially with Circular logging

- **You can use the amqldmpa command to dump the state of the logger**
  - *amqldmpa -m <qmgr> -c H -f <file>*
  - In <file> look for the following

```
logactive:                3
loginactive:              47
...
FileCount:                37
filenum                   [4,5,6,7,8,9,10,11,
                           12,13,14,15,16,17,18,19,
                           20,21,22,23,24,25,26,27,
                           28,29,30,31,32,33,34,35,
                           36,2,0,1,3]
```

  - logactive: Number of primary log files
  - loginactive: Number of secondary log files
  - FileCount: Number of active log files – if greater than logactive than using secondaries
  - Filenum: List of log files currently in use

- **See the following Technote:** http://www-01.ibm.com/support/docview.wss?uid=swg21623541

Looking at Logger Performance (1)

- **Poor performance due to logger delays can be difficult to diagnose**
  - Many MQ performance PMRs turn out to be logger-related
    - Many of these turn out to be disk-related
- **You can use the amqldmpa command to dump the state of the logger**
  - *amqldmpa -m <qmgr> -c H -f <file>*
  - In <file> look for the following

```
logBufSz;                    512
...
WriteSizeShort    :          16941
WriteSizeLong     :          78059
WriteSizeMax      :          2072576
```

  - logBufSz: Size of log buffer in 4K pages (512 = 2MB buffer)
  - WriteSizeShort: Short-term weighted average size (in bytes) of 64 most recent writes
  - WriteSizeLong: Longer-term weighted average size (in bytes) of 1024 most recent writes
  - WriteSizeMax: Largest single write (in bytes)
- **In this example…**
  - Averages are pretty decent (~16KB to 76KB)
  - But the buffer is almost maxed out (2MB)
  - So might be worthwhile to increase log buffer in this case

Looking at Logger Performance (2)

- **Many logger delays turn out to be disk-related**
  - For these must look outside MQ for resolution - but MQ can provide some clues
- **You can use the amqldmpa command to dump the state of the logger**
  - *amqldmpa -m <qmgr> -c H -f <file>*
  - In <file> look for the following

```
WriteTimeShort    :             611
WriteTimeLong     :             2722
...
WriteTimeShortMax:              199610
WriteTimeLongMax  :             199610
```

  - WriteTimeShort: Short-term weighted average time (in µs) of 64 most recent writes
  - WriteTimeLong: Longer-term weighted average time (in µs) of 1024 most recent writes
  - WriteTimeShortMax: Short-term weighted average time HWM (in µs)
  - WriteTimeLongMax: Longer-term weighted average time HWM (in µs)
- **In this example…**
  - Near-term average write times good (<3ms)
  - But there have been spikes (~200ms)
  - May be worth monitoring over time to see longer term averages

Looking at Logger Performance (3)

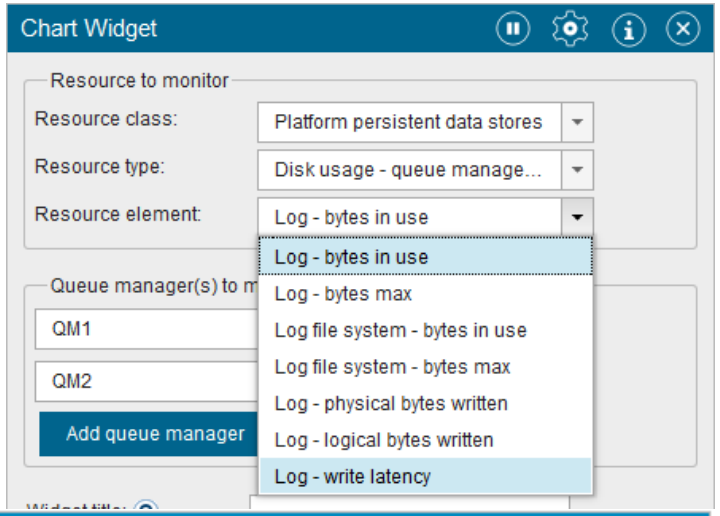
- **Write-time HWMs may be more useful for monitoring over time**
- **You can use the amqldmpa command to dump the state of the logger**
  - *amqldmpa -m <qmgr> -c H -f <file>*
  - In <file> look for the following

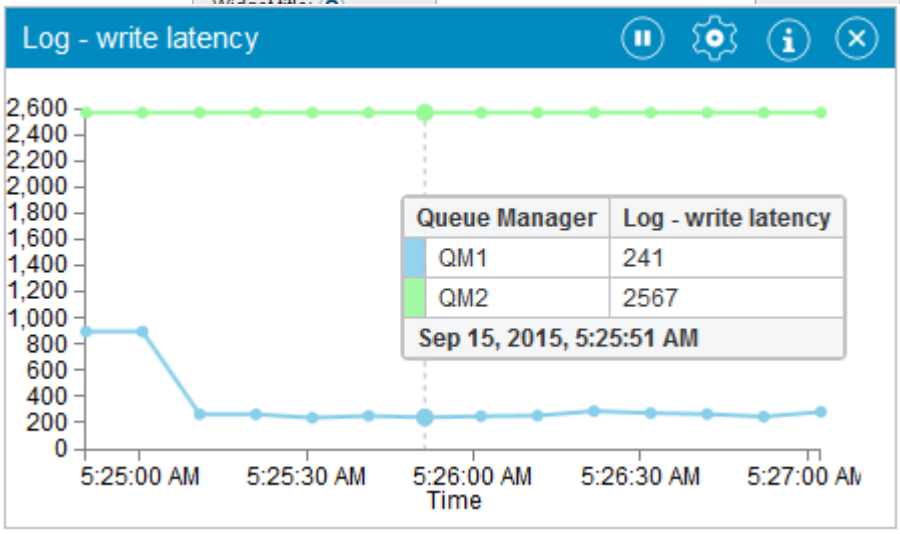
```
WriteTimeMax     :          2757339 at 2015-09-16 11:23:23.122
WriteTimeMax[0]:            2757339
WriteTimeMax[7]:            541377
WriteTimeMax[6]:            341761
WriteTimeMax[5]:            2713306
WriteTimeMax[4]:            314304
WriteTimeMax[3]:            249737
WriteTimeMax[2]:            77320
WriteTimeMax[1]:            57664
```

  - WriteTimeMax: Longest log write time since queue manager started (in µs)
    - V8 will report the date/time this occurred
    - V8 will also report longest write time for 8 most recent log files used
- **In this example…**
  - HWM > 2.757 seconds – ouch! (I've seen customer examples > 60 seconds!!!)
  - Of the last 8 log files none were very good (from 57ms to 2713ms)
  - Capture over time and sit down with SAN/NAS teams to resolve

Logging and the MQ Appliance

- **Appliance-hosted queue managers use recovery logs as well**
  - Circular logging only
  - Same defaults as Installable MQ
    - Which can be configured
- **Appliance HDD a finite resource**
  - Monitoring important
  - Especially if hosting multiple queue managers
- **Chart widgets make this easy**
  - Free space
  - Log usage
  - Write Latency
- **New event generation scheme**
  - Published on well-known topics
  - Can create your own alert monitor
  - Sample shipped with MQ 8.0.0.3





Recent Changes…

- **More granular measure of maximum log write time**
  - Before V8 this was a single number with the HWM since queue start
  - In V8 it now reports the HWM for each of the 8 most recent log files
    - Allows monitoring over time using amqldmpa
- **Changes to Reduce Log I/O:**
  - When possible V8 uses 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 operation
    - Older MQ versions suffered an additional log force when log buffer wrapped
  - Default checkpoint frequency is now every 50,000 recoverable operations
    - Older versions of MQ default to every 10,000 operations
    - If you used *CheckPointLogRecdMax* you may want to revisit
  - Default checkpoint delay length is now 0.5 seconds
    - Older releases of MQ used 0.25 seconds

Agenda

What are logs and what are they used for

Logging parameters

Logging configuration

Creation of logs

Log usage

Log management

Recovery

Summary and questions

Summary

- **Mysteries and Decisions**
  - Circular and Linear
  - Understanding Logging Parameters
  - Log Write Integrity!
- **Log Operation**
- **AMQ7469!**
- **Behind the Curtain**
  - Tools to help you better understand the Logger
- **What's New**
  - MQ V8
  - MQ Appliance
- **Wrap-Up**

Questions?