#IDUGDb2

# Db2 Advanced Log Space Management – Trying to Make Life Simple

**Michael Roecken**

*IBM*

Session code: C10

October 23, 2019, 9:30 – 10:30 Db2

IDUG
Leading the Db2 User Community since 1988

Trying to figure out the ins and outs of Db2 Log Space Management? Spent too much time trying to figure out what settings are best for your environment? Look no further than Db2's new Advanced Log Space Management. With the introduction of Db2 11.5 the journey has begun in making log space management more autonomic. This presentation will introduce you to some of the concepts and principles behind it all. We will do a deep dive into the technology and transformation behind making log space management more hands off, which in the end will make life simple for you and your organization.

## Objectives

Describe and discuss the following:

- Today's log space management and the challenges that arise
- Introduction to Db2 Advanced Log Space Management -- what is so advanced about it?
- Deep dive into the technology behind Db2 Advanced Log Space Management and is it right for you and your organization
- Monitoring and Problem Analysis – Identify log space issues and discuss what to do when Db2 log management is not behaving as expected
- What's next?  The next steps towards making log space management more automated

# Safe Harbor Statement



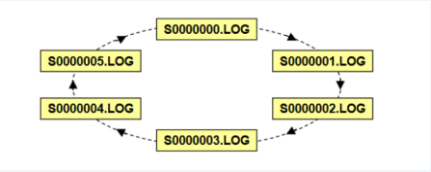
3

# Agenda

- **Refresher – Log Management Basics**
- What's New in Log Management
  - Version 11.1 and 11.5
- Advanced Log Space Management
  - Overview
  - Monitoring and Problem Analysis
  - Demo
- The Future in Log Management

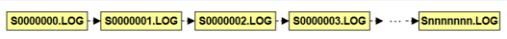
# Refresher – Logging Types

- **Circular**




  - Default for new databases
  - Log data is overwritten
  - Supports crash recovery
  - No online or table space level backup
  - No rollforward support

- **Recoverable**




  - Log retain or Archiving
  - Log data is never reused
  - Log retain keeps files in active log path
  - Archiving allows files to be stored in separate location
  - Supports crash recovery
  - Supports online and table space level backup
  - Rollforward support to point in time

# Refresher – Configuring Logging

- **Log paths**
  - NEWLOGPATH
  - MIRRORLOGPATH
  - OVERFLOWLOGPATH *
- **Archiving**
  - LOGARCHMETH1/2 *
  - LOGARCHCOMPR1/2 *
  - LOGARCHOPT1/2 *
  - NUMARCHRETRY *
  - ARCHRETRYDELAY *
  - FAILARCHPATH *
- **Log Space**
  - LOGBUFSZ
  - LOGPRIMARY
  - LOGSECOND *
  - LOGFILSIZ
  - LOG_DISK_CAP * (future)
- **Flushing**
  - PAGE_AGE_TRGT_MCR
  - PAGE_AGE_TRGT_GCR
  - SOFTMAX (deprecated)
- **Transaction**
  - BLK_LOG_DSK_FULL *
  - MAX_LOG *
  - NUM_LOG_SPAN *
  - BLOCKNONLOGGED *

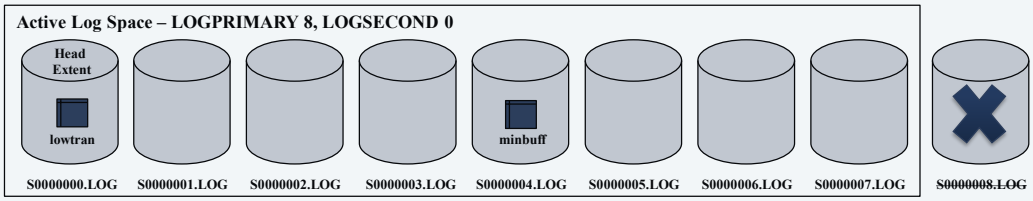
** Configurable online*

# Refresher – What is Transaction Log Full? (1|2)

- **Maximum active log space**
  - (LOGPRIMARY + LOGSECOND) * LOGFILSIZ
- **Fixed active log space**
  - LOGPRIMARY * LOGFILSIZ
- **lowtran**
  - First (lowest) log record belonging to oldest open transaction
- **minbuff**
  - Log record of the oldest (minimum) dirty page in buffer pool

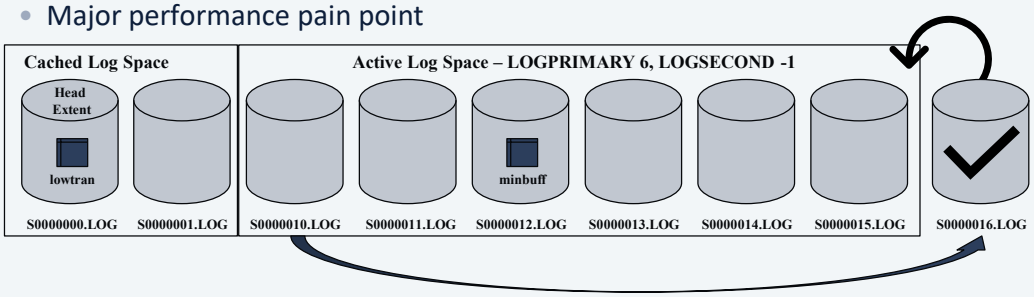
# Refresher – What is Transaction Log Full? (2|2)

- **Db2 saves log files from min(lowtran, minbuff) called head extent for rollback/crash recovery**
- **Transaction log full is when Db2 needs to create a new log file above LOGPRIMARY+LOGSECOND but cannot because lowtran and/or minbuff do not move up**
  - lowtran => open transaction; minbuff => bufferpool flushing slow



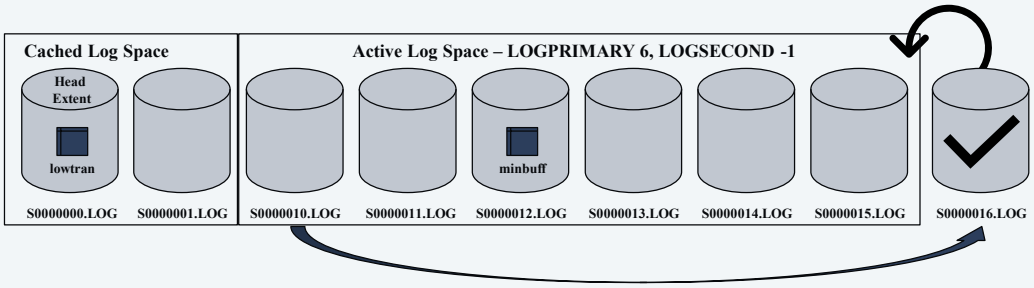
# Refresher – Infinite Logging (1|2)

- **One way to avoid transaction log full is use infinite logging (LOGSECOND = -1)**
  - Files from head extent and onwards not guaranteed to be in active log path
  - Avoid rogue transactions by using configuration parameters
    - NUM_LOG_SPAN and/or MAX_LOG
  - Rollback and crash recovery may have to retrieve log files from archives
    - Major performance pain point



# Refresher – Infinite Logging (2|2)

- Online backup has to include many more log files
  - Increased image size
  - Longer running backups
- Db2 caches some files (up to 8) above active log space to mitigate need to retrieve log files from archives



# Agenda

- Refresher – Log Management Basics
- **What's New in Log Management**
  - **Version 11.1 and 11.5**
- Advanced Log Space Management
  - Overview
  - Monitoring and Problem Analysis
  - Demo
- The Future in Log Management

# What's New in Log Management – Version 11.1 (1|3)

**11.1.0.0**

- **Fast pre-allocation for log file creation and resize**
  - DB2_USE_FAST_LOG_PREALLOCATION
- **Db2 log compression on POWER 7+/ 8 using NX842**
  - AIX only
  - compress comprlib libdb2nx842.a

**11.1.1.1**

- **Increase of limit on LOGFILSIZ (to 64GB)**
  - Maximum theoretical size is 256 log files * 64GB = 16TB

# What's New in Log Management – Version 11.1 (2|3)

**11.1.3.3**

- **The archival of log files using VENDOR or TSM methods can now be configured with a timeout on Unix environments**
  - LOGARCHOPT1/2: --VENDOR_ARCHIVE_TIMEOUT

# What's New in Log Management – Version 11.1 (3|3)

**11.1.4.4**

- **For UNIX databases configured with a mirrored log path a potential performance improvement can be achieved by writing log data to both files asynchronously in parallel**
  - DB2_USE_ASYNC_FOR_MIRRORLOG
- **Rollback performance improvements using buffered I/O when reading transaction log file data**
  - Internal tests show 3x improvement
  - DB2_USE_BUFFERED_READ_FOR_ACTIVE_LOG

# What's New in Log Management – Version 11.5 (1|4)

**11.5.0.0**

- **Changed default behavior (ON)**
  - For UNIX databases configured with a mirrored log path a potential performance improvement can be achieved by writing log data to both files asynchronously in parallel
  - Rollback performance improvements using buffered I/O when reading transaction log file data

# What's New in Log Management – Version 11.5 (2|4)

- **New registry variable**
  - **DB2_HADR_STANDBY_KEEP_UNARCHIVED_LOGS**
    - Controls whether standby will keep log files when the corresponding log files on primary are not archived
    - Set to FALSE on standby, standby will delete log files when the corresponding log files on primary are not archived
    - DEFAULT: TRUE

# What's New in Log Management – Version 11.5 (3|4)

- **Changes to logprimary / logsecond database configuration parameters**
  - Increased each from 256 to 4096
    - logprimary + logsecond <= 8192
  - Recoverable databases only
  - Be aware: Db2 instance can only have open 65536 concurrent files
  - Maximum theoretical size is:
    - Circular: 256 log files * 64GB = 16TB
    - Recoverable: 8192 log files * 64GB = 512TB

# What's New in Log Management – Version 11.5 (4|4)

- **Reduced logging**
  - Reduced undo logging on by default in 11.5 GA
    - Required log space cut in half
  - Reduced redo logging available only in Warehouse installations
    - Up to 95% less logging
- **Advanced log space management**
  - Tech Preview (not for production use)
  - Reduce transaction log full

# Reduced Logging

- **Applies to:**
  - Column organized tables only
  - Any bulk operation (e.g. upgrade or ingest) which drives insert internally
- **Reduced Undo logging improvements:**
  - Available in 11.5 GA by default
  - Avoid need to reserve log space for undo log records
    - Log space required cut in half
- **Reduced Redo logging improvements:**
  - Available only in Warehouse installations
  - Log meta data changes but skip logging of page contents
  - Similar to "Not Logged Initially" tables but with improved recoverability and concurrency
- **Table contents will be preserved during:**
  - Rollback
  - Crash recovery
  - Database rollforward recovery to end of backup ONLY

**Total impact: 95% reduction in required log space**

19

Db2 Warehouse columnar deployments typically focused on workload that make heavy use of Extract Load and Transform within thedatabase (ELT) operations.

Need to support massive data volumes in single INSERT statements simply can not spool the necessary log space for such largetransactions.

Historically NLI would be leveraged but has a number of properties not friendly to append or update operations– failures take table off line thus backs before/after required or at least recommended

Reduce Logging is a friendly version of NLI. In Db2 Warehouse is enabled by default Implicitly kicks in when a bulk INSERT or UPDATE operation is detected.

Significantly reduces the amount of logging by not logging data pages but still logs all necessary meta data changes to support rollback and crash recovery

While reduce logging can benefit performance due to the reduce logging it is not the main reason for introducing it to Db2 Warehouse.
And it does add additional impact as it requires flush on commit semantics to be enforced.

However it does mean that log based operations like Point-In-Time recovery are no longer available.

# Agenda

- Refresher – Log Management Basics
- What's New in Log Management
  - Version 11.1 and 11.5
- **Advanced Log Space Management**
  - **Overview**
  - Monitoring and Problem Analysis
  - Demo
- The Future in Log Management

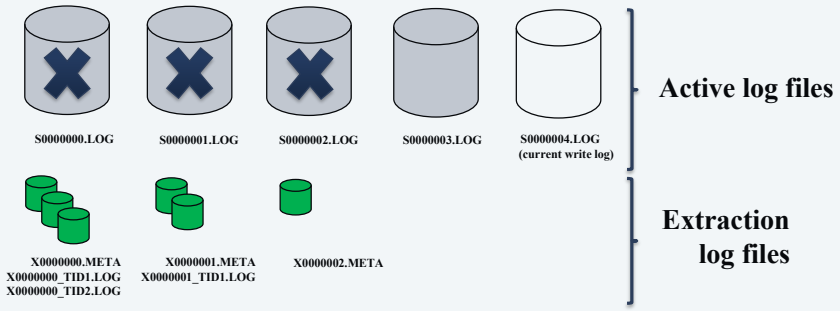
# Advanced Log Space Management – Problem and Use Case

- **Reduce transaction log full**
  - Often complaints about transactions hitting transaction log full
    - #1 request from many big customers
- **Quick short running transactions running in parallel with:**
  - Long running transactions
    - Low logging rate
    - Log one or two log records and then sit idle for some time
  - LOAD
  - CREATE INDEX
- **Long running (monster) transactions won't see much benefit**
  - High logging rate
  - Log volume issue
  - Same behavior as before ➔ transaction log full

# Advanced Log Space Management – Objective

- **First objective**
  - Log full avoidance
  - Provide monitoring tools to help manually tune active log space
- **Future objective**
  - More autonomic log space management
    - Today log space is fixed size
    - Move towards Db2 managing log space based on log path file system provided (e.g. LOGPRIMARY, etc. can become automatic)

# Advanced Log Space Management – Solution (1|9)

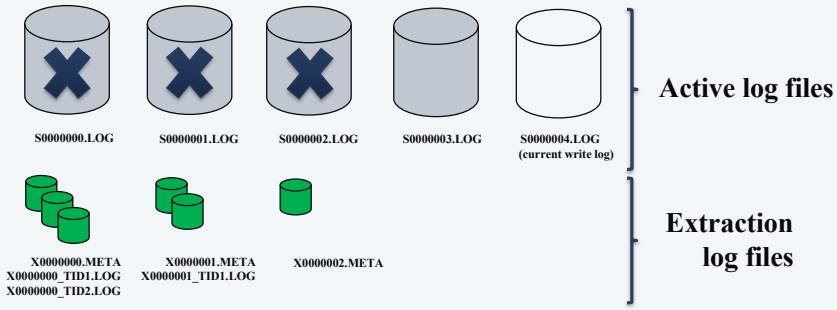



- Extract log records for long running active transactions to separate extraction log files under active log path
- Allows active log files to be closed, archived, and reused, thus avoiding transaction log full



Extraction will kick in when a certain percentage of log space has been consumed and will extract log records starting from head extent ID into separate extraction log files stored in the active log path.

# Advanced Log Space Management – Solution (2|9)




- **New files in active log path:**
  - **X<logFileNum>_TID<tranId>.LOG** - extraction transaction ID (TID) file. Extracted log records for a specific transaction used by rollback and crash recovery. 1 file per log file where log data is extracted for a transaction ID.
  - **X<logFileNum>.TMP** - meta data about extracted logs created during an in progress extraction for an active log file.
  - **X<logFileNum>.META** - meta data about extracted logs created after extraction completes for an active log file.

# Advanced Log Space Management – Solution (3|9)




- **Databases must be configured with archive logging**
- **Extraction takes place by new EDU – db2loggx**
- **No to minimal impact to active workloads**
- **Extraction will be throttled based on policies such as:**
  - Disk available
    - Not enough disk space, extraction will idle
  - Log space consumed
    - Log space consumption high, extraction will kick in
  - Producing a benefit
    - No benefit seen, maybe due to monster transaction, extraction will idle
- **Idle extraction means possible transaction log full can occur**

# Advanced Log Space Management – Solution (4|9)

- **An idle extraction scan can happen because:**
  - **Log archiving not healthy**
    - Log data from the active log files that is not archived yet is not extracted
    - Ensure log archiving is healthy and/or a FAILARCHPATH is configured
  - **Buffer pool flushing is slow**
    - Log data from the active log files that is at or above what has been flushed from the buffer pools is not extracted
    - Ensure PAGE_AGE_TRGT_MCR and PAGE_AGE_TRGT_GCR (or SOFTMAX on older database configurations) are set to appropriate values based on your workload throughput

# Advanced Log Space Management – Solution (5|9)

- **An idle extraction scan can happen because:**
  - **Extraction is slow**
    - It is possible that log writing is faster than log extraction or log extraction has triggered too slowly
  - **Extraction write error**
    - Including disk full

# Advanced Log Space Management – Solution (6|9)

- **Rollback**
  - A line is created to determine whether read from active log files or read from extraction log files
  - Error reading extraction log files will retrieve log data from archives
  - Has shown a performance improvement for a rollback of a single transaction
- **Currently committed**
  - A line is created to determine whether read from active log files or read from extraction log files
  - Error reading extraction log files will resort to lock wait behavior

# Advanced Log Space Management – Solution (7|9)

- **Crash recovery**
  - Use extraction log files for redo and undo
  - Post crash recovery, extraction scan will continue where it left off, so can support indoubt transactions or any deferred undo such as from DB2_ONLINERECOVERY
  - Error reading extraction log files will retrieve log data from archives
- **Set write suspend**
  - Extraction and set write suspend are serialized just like log writing
- **Encryption aware**
  - If database encrypted, extraction log files will be encrypted

## Advanced Log Space Management – Solution (8|9)

- **Monitoring:**
  - MON_GET_TRANSACTION_LOG
  - MON_GET_UNIT_OF_WORK
  - MON_GET_UNIT_OF_WORK_DETAILS
  - db2pd –logs
- **No change to usage of MAX_LOG db cfg parm**
  - Still works on active log space as before
- **Re-visit intent of NUM_LOG_SPAN db cfg parm**
  - Does not apply to utility workloads like LOAD, so no impact
  - For non-utility workloads if set too low extraction may never kick in

# Advanced Log Space Management – Solution (9|9)

- **Extraction enabled: No infinite logging vs. infinite logging**
  - No infinite logging
    - Transaction log full still possible in extreme cases
  - Infinite logging
    - Transaction log full will not happen
    - But possible workload lags in extreme cases

# Advanced Log Space Management – 11.5 GA Restrictions

- **Tech Preview**
  - **Do not use in production**
  - Enable with DB2_ADVANCED_LOG_SPACE_MGMT=ON
- **Future Support**
  - Databases configured with HADR
  - Databases configured with mirrored logging (MIRRORLOGPATH)
  - Databases in pureScale environments
- **No Plan to Support**
  - Databases configured with circular or log retain logging (LOGARCHMETH1/2)

# Advanced Log Space Management – 11.5 GA Limitations (1|2)

- **Disk space**
  - Will consume additional disk space to hold extraction log files
  - Should provide extra disk space otherwise extraction will not take place and log full can occur
- **Online backup**
  - Extraction log files will not be included in backup image
    - May need to retrieve active log files
  - Could increase range of log files that needs to be included
    - Larger image sizes
    - Take longer

# Advanced Log Space Management – 11.5 GA Limitations (2|2)

- **Crash recovery**
  - Redo phase will retrieve log files
  - Undo phase will use extraction log files but may not be optimal performance
- **Restore and rollforward**
  - Deletes all extraction log files
  - Will retrieve log files

# Agenda

- Refresher – Log Management Basics
- What's New in Log Management
  - Version 11.1 and 11.5
- **Advanced Log Space Management**
  - Overview
  - **Monitoring and Problem Analysis**
  - Demo
- The Future in Log Management

# Advanced Log Space Management – Monitoring

## New columns for MON_GET_TRANSACTION_LOG

| Column Name | Data Type | Description |
|---|---|---|
| LOG_EXTRACTION_PROCESSED_BYTES | BIGINT | Number of bytes analyzed for extraction |
| LOG_EXTRACTION_PROCESSING_TIME | BIGINT | Time spent to extract log records |
| LOG_EXTRACTION_WRITTEN_BYTES | BIGINT | Number of bytes written to extraction log files |
| LOG_EXTRACTION_WRITE_TIME | BIGINT | Time spent writing to extraction log files |
| LOG_EXTRACTION_ROLLBACK_READS | BIGINT | Number of lookups in extraction files for rollback |
| LOG_EXTRACTION_ROLLBACK_TIME | BIGINT | Time spent for rollback lookups in extraction log files |
| LOG_EXTRACTION_CUR_COMMIT_READS | BIGINT | Number of lookups in extraction files for currently committed |
| LOG_EXTRACTION_CUR_COMMIT_TIME | BIGINT | Time spent for currently committed lookups in extraction log files |
| LOG_EXTRACTION_DISK_SPACE_USED_TOTAL | BIGINT | Number of bytes used in extraction log files |
| LOG_EXTRACTION_DISK_SPACE_USED_TOTAL_TOP | BIGINT | High water mark of LOG_EXTRACTION_DISK_SPACE_TOTAL_USED since database member activation |
| LOG_EXTRACTION_LAST_EXTRACTED_LOG | BIGINT | Log extent number of the last log file successfully extracted |
| LOG_EXTRACTION_PROCESSED_LSO | BIGINT | The log sequence offset of last processed log record for extraction |
| LOG_EXTRACTION_PROCESSED_LSN | BIGINT | The log sequence number of last processed log record for extraction |
| LOG_EXTRACTION_NUM_DISK_FULL | BIGINT | Number of times log extraction stopped, because there was not enough disk space in active log path |

# Advanced Log Space Management – Monitoring

## New column for MON_GET_UNIT_OF_WORK

| Column Name | Data Type | Description |
|---|---|---|
| LOG_EXTRACTION_DISK_SPACE_USED | BIGINT | Number of bytes used in extraction log files |

## New XML element for MON_GET_UNIT_OF_WORK_DETAILS

| Column Name | Data Type | Description |
|---|---|---|
| LOG_EXTRACTION_DISK_SPACE_USED | xs:nonNegativeInteger | Number of bytes used in extraction log files |

# Advanced Log Space Management – Monitoring

## db2pd –db sample -logs

```
Logs:
Current Log Number            54
Pages Written                 15
Cur Commit Disk Log Reads     0
Cur Commit Total Log Reads    0
Method 1 Archive Status       Success
Method 1 Next Log to Archive  54
Method 1 First Failure        n/a
Method 2 Archive Status       n/a
Method 2 Next Log to Archive  n/a
Method 2 First Failure        n/a
Extraction Status             Active
Current Log to Extract        46
Log Chain ID                  0
Current LSO                   65116033
Current LSN                   0x00000000000565F1

Address            StartLSN         StartLSO            State      Size       Pages      Filename
0x00007F4F045C8C78 0000000000054EBD 64531233            0x00000000 16         16         S0000046.LOG
…  <truncated> …
```

### New rows:

**Extraction Status** - The current status of extraction. Values can be "n/a", "Active“, "Error“ or “Recovery”.

**Current Log to Extract** - The current log to extract. This is the active log file that extraction is extracting from or needs to extract from.

# Advanced Log Space Management – Monitoring Would feature be beneficial? (1|2)

- **Three considerations of what "beneficial" means:**
  - Avoid transaction log full
  - Disk space consumption
  - CPU and I/O overhead
- **Ideally, the best of all**
  - Avoid transaction log full by extracting very little
  - Extraction process shown to be little overhead

# Advanced Log Space Management – Monitoring Would feature be beneficial? (2|2)

- **Monitor the longest running transactions on the database**
- **(A) Get total amount of log space used by the 5 longest running transactions:**

```
SELECT SUM(uow_log_space_used)
FROM (SELECT uow_log_space_used,
             uow_start_time
      FROM TABLE(MON_GET_UNIT_OF_WORK(NULL,-1)) AS t
      ORDER BY uow_start_time ASC FETCH FIRST 5 ROWS ONLY)
```

- **(B) Get total active log space used by the database:**

```
SELECT total_log_used
FROM TABLE(MON_GET_TRANSACTION_LOG (-1)) AS t
```

- **If ratio of A / B is low then extraction will be beneficial**
  - Indicates long running low logging volume

40

> "If ratio of A / B is low then extraction will be beneficial"

"low" here is relative. The lower the number the better the disk space saving. So need to decide what amount of disk space you are willing to save in order to avoid transaction log full.

## Advanced Log Space Management – Monitoring
## How much disk space do I need to run optimally with feature?

- **Depends on amount of data that can be extracted**
  - If very little to extract then potentially can reduce active log space
- **Bare minimum is enough disk space to extract from one active log file**
  - Non-infinite: (LOGPRIMARY + LOGSECOND + 1) * LOGFILSIZ
  - Infinite: (LOGPRIMARY + 1) * LOGFILSIZ
- **Recommend ~20% extra disk space**
- **Continue to monitor until you find right fit**

# Advanced Log Space Management – Monitoring
# Is feature enabled?

- **Turn on registry variable:**
  - db2set DB2_ADVANCED_LOG_SPACE_MGMT=ON

- Not enabled:
  - db2diag.log will state reason

```
db2pd -db sample -logs

Extraction Status             n/a
Current Log to Extract        n/a
```

- Enabled:

```
db2pd -db sample -logs

Extraction Status             Active
Current Log to Extract        0

db2pd -edus |grep loggx
901       140049278560000      7921                db2loggx (SAMPLE)


EDUID    : 901                  EDUNAME: db2loggx (SAMPLE)
FUNCTION: DB2 UDB, data protection services,
sqlpLogExtractionScanCB::loggxEnableExtractionScan, probe:1410
DATA #1 : <preformatted>
Log extraction under advanced log space management has been enabled for
database.
 Primary extraction path = /db2/NODE0000/SQL00001/LOGSTREAM0000/
  Mirror extraction path = Not set
```

42

Turn on the registry variable DB2_ADVANCED_LOG_SPACE_MGMT and activate database. The db2diag.log will display a message whether log extraction is enabled or not. db2pd –logs will also show state of extraction. All extraction is done by the new db2loggx EDU.

# Advanced Log Space Management – Monitoring
# How efficient is extraction?

- **This can be based on the extraction filter rate**
  - Data analyzed vs. data written

```
SELECT log_extraction_written_bytes,
       log_extraction_processed_bytes
FROM TABLE(MON_GET_TRANSACTION_LOG(-1)) as t

      LOG_EXTRACTION_WRITTEN_BYTES LOG_EXTRACTION_PROCESSED_BYTES
      ---------------------------- ------------------------------
                             16589                         647632
```

- **16,589 / 647,632 = ~3% of active log data written has been extracted = efficient**

# Advanced Log Space Management – Monitoring
# Is feature healthy and working? (1|3)

- **What constitutes extraction health:**
  - No transaction log full ☺
  - Right workload / configuration (extraction filter rate)
    - Using the extraction filter rate can tell if extracting too much
      - Main cause of slow extraction speed
  - Archiving
    - Monitor to ensure not falling behind or sick
    - Use FAILARCHPATH
  - Bufferpool flushing (minbuff)
    - Verify PAGE_AGE_TRGT_MCR / PAGE_AGE_TRGT_GCR (or SOFTMAX)
  - Disk full
    - Verify storage space assigned to active log paths

# Advanced Log Space Management – Monitoring Is feature healthy and working? (2|3)

- **Helpful queries/commands:**

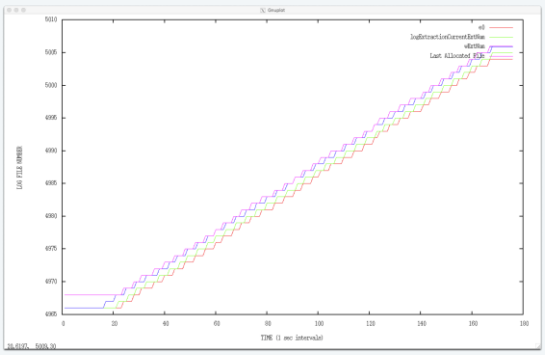
```
SELECT first_active_log,
       log_extraction_last_extracted_log AS last_extracted_log,
       log_extraction_num_disk_full AS num_extract_disk_full,
       archive_method1_status,
       method1_next_log_to_archive,
       current_active_log,
       last_active_log
FROM TABLE(MON_GET_TRANSACTION_LOG(-1)) as t
```

```
FIRST_ACTIVE_LOG     LAST_EXTRACTED_LOG   NUM_EXTRACT_DISK_FULL ARCHIVE_METHOD1_STATUS
-------------------- -------------------- --------------------- ----------------------
                 989                 1032                     0                      1

METHOD1_NEXT_LOG_TO_ARCHIVE CURRENT_ACTIVE_LOG   LAST_ACTIVE_LOG
--------------------------- -------------------- --------------------
                       1034                 1034                 1035
```

# Advanced Log Space Management – Monitoring
# Is feature healthy and working? (3|3)

- **db2pd -db sample -logs**



```
Logs:
Current Log Number              1034
…
Method 1 Archive Status         Success
Method 1 Next Log to Archive    1034
…
Extraction Status               Active
Current Log to Extract          1033
…

StartLSN         StartLSO             State      Size        Pages       Filename
00000000000729AB 78259201             0x00000000 4           4           S0001026.LOG
…
0000000000000000 78405937             0x00000000 4           4           S0001035.LOG
```



No log data will be extracted from an active log file that has not been archived yet. This would duplicate disk space. Ensure methx_status is 1 (healthy), not 0 (error).

No log data will be extracted from an active log file where minbufflsn exists. This is due to recovery algorithm that needs to replay all log records >= minbufflsn. So no benefit of extracting such data as it would duplicate disk space.

# Advanced Log Space Management – Monitoring
# What is the disk space consumption of extraction?

- Current total extraction disk space consumed
- Maximum total extraction disk space consumed since last activation

```
SELECT log_extraction_processed_bytes AS processed_bytes,
       log_extraction_written_bytes AS written_bytes,
       log_extraction_disk_space_used_total AS disk_space_used_total,
       log_extraction_disk_space_used_total_top AS disk_space_used_total_top
FROM TABLE(MON_GET_TRANSACTION_LOG(-1)) as t
```

```
PROCESSED_BYTES WRITTEN_BYTES DISK_SPACE_USED_TOTAL DISK_SPACE_USED_TOTAL_TOP
--------------- ------------- --------------------- -------------------------
         266882           165                 35165                     54461
```

47

This query tells you that since the last activation extraction processed 266,882 bytes of log data from the active log files. From that amount, 165 bytes of log data was written to extraction TID files. The current total amount of disk space consumed by extraction files, including log data and meta data, is 35,165 bytes. Since the last activation, extraction has taken up 54,461 bytes.

# Advanced Log Space Management – Monitoring
# What transaction consumes the most extraction disk space?

```
SELECT application_handle,
       substr(char(APPLICATION_NAME), 1, 16) AS app_name,
       uow_log_space_used AS active_disk_space_used,
       log_extraction_disk_space_used AS extract_disk_space_used
FROM TABLE(MON_GET_UNIT_OF_WORK(NULL,-1)) AS t
ORDER BY extract_disk_space_used DESC fetch first 1 rows only
```

```
APPLICATION_HANDLE   APP_NAME         ACTIVE_DISK_SPACE_USED EXTRACT_DISK_SPACE_USED
-------------------- ---------------- ---------------------- ----------------------
                   9 db2bp                                841                    293
```

- **db2pd -db sample -applications**

```
AppHandl …   Appid
9         …   *LOCAL.user.190926023238
```

- **db2pd -db sample -transactions**

```
AppHandl …   TID
9         …   0x00000000115A
```

**NOTE: Same TID that matches directory listing in log directory**

```
> ls -l
293 X0000868_TID000000000000115A.LOG
```

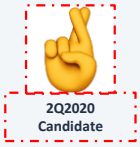
48

To find the transaction that is consuming the most extraction log space allows one to understand if this is a known expectation or not, maybe possibly a rogue transaction.

You can map a transaction ID (TID) from a directory listing or you can use a combination of commands to track down which application/transaction is consuming the most amount of extraction space.

The above example shows that this particular transaction has written 841 bytes of log data to the active files, but only 293 bytes have been extracted so far.

# Advanced Log Space Management – Problem Analysis
# db2fmtlog – Format and display log file information command




- **No support in 11.5 GA**
- **Future support coming**
  - New "-xlog" option to handle extraction log files (both META and TID files)
  - Any time a TID file is formatted associated META file will be formatted
  - Will display mainly meta data about files, but no log record data

```
Log File Options:

>--log_file_number_start--+----------------------+------><
                          +--log_file_number_end--+

>--- -xlog--+-------------------------------------------------+---+----------------+--><
            +--filename---------------------------------------+   +- -tid--tidNum--+
            +--log_file_number_start--+----------------------+
                                      +--log_file_number_end--+
```



db2fmtlog - Format and display log file information command
https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.5.0/com.ibm.db2.luw.admin.cmd.doc/doc/r0070378.html

Example:

X0000000.META
X0000000_TID0000000000000124.LOG
X0000000_TID0000000000000125.LOG
X0000001.META
X0000001_TID0000000000000125.LOG
X0000001_TID0000000000000126.LOG

Each db2fmtlog call will format the listed files:

db2fmtlog -xlog 0

X0000000.META
X0000000_TID0000000000000124.LOG
X0000000_TID0000000000000125.LOG

db2fmtlog -xlog -tid 125

X0000000.META
X0000000_TID0000000000000125.LOG
X0000001.META
X0000001_TID0000000000000125.LOG

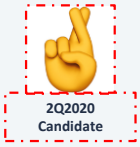
db2fmtlog -xlog 1 -tid 125

X0000001.META

X0000001_TID0000000000000125.LOG

# Advanced Log Space Management – Problem Analysis
# db2cklog – Check archive log file validity command

- **No support in 11.5 GA**
- **Future support coming**
  - No changes to the syntax of db2cklog
  - If S0000000.LOG exists we check
  - If S0000000.LOG does not exist:
    - If X0000000.META/X0000000_TID<tid>.LOG exist we verify and raise warning
    - If X0000000.META/X0000000_TID<tid>.LOG does not exist we raise error




```
                .-CHECK-.
>>-DB2CKLOG--+-------+--log_file_number1--+----------------------+-->
                                           '-TO--log-file-number2-'

>--+-------------------------------+--------------------------->
   '-ARCHLOGPATH--archive_log_path-'
```



db2cklog - Checking archive log files with the db2cklog tool
https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.5.0/com.ibm.db2.luw.admin.trb.doc/doc/t0058518.html

# Advanced Log Space Management – Problem Analysis
# Corruption of Extraction Log Files

- **If any META or TID is bad, can always make use of Snnnnnnn.LOG**
  - If still local then will use
  - If in archive then will retrieve and use
- **Rollback / Crash Recovery / Rollforward / HADR Standby Replay**
  - Will always retrieve in order to bring or keep database online
- **Online Backup (future)**
  - Will always retrieve from error point onwards to produce recoverable unit
- **Currently Committed**
  - Will NOT retrieve and resort to lock wait behavior
  - Will assume wait time shorter than retrieve time

# Advanced Log Space Management – Problem Analysis
# Still hit transaction log full, why? (1|4)

**db2pd -db sample -logs**

```
Logs:
Current Log Number            1047
Pages Written                 0
Cur Commit Disk Log Reads     0
Cur Commit Total Log Reads    0
Method 1 Archive Status       Failure
Method 1 Next Log to Archive  1047
Method 1 First Failure        1038
Method 2 Archive Status       n/a
Method 2 Next Log to Archive  n/a
Method 2 First Failure        n/a
Extraction Status             Active
Current Log to Extract        1038
Log Chain ID                  0
Current LSO                   78605624
Current LSN                   0x00000000000735A6
```

**db2diag.log:**

```
EDUID   : 24                  EDUNAME: db2loggr (SAMPLE)
FUNCTION: DB2 UDB, data protection services, sqlpgadf,
DATA #1 : <preformatted>
Active log S0001038.LOG has not been archived yet.
Active log S0001038.LOG has not been extracted from yet.

EDUID   : 24                  EDUNAME: db2loggr (SAMPLE)
FUNCTION: DB2 UDB, data protection services, sqlpgadf,
DATA #1 : <preformatted>
Current log extraction information:
          loggxLastProcessedLsn = 0000000000072FEE
          loggxLastProcessedLso = 78454802
     logExtractionCurrentExtNum = 1038
             logExtractionState = IDLE
          logExtractionFlushLsn = 0000000000000000
                throttleReason = LOG_ARCHIVING
```

52

With extraction running as per the Extraction Status set to Active, your workload still hits transaction log full. You run the db2pd -logs command and it shows you that log archive method 1 is in an error state on file 1038. Extraction is also currently trying to extract from the same file. By going to the db2diag.log and finding the SQLP_NOSPACE error, you see that the extraction scan is being throttled due to log archiving holding extraction up. Look into the archiving issue and attempt to resolve, at which point extraction will begin again.

# Advanced Log Space Management – Problem Analysis
# Still hit transaction log full, why? (2|4)

**db2pd -db sample -logs**

```
Logs:
Current Log Number            1060
Method 1 Archive Status       Success
Method 1 Next Log to Archive  1060
Method 1 First Failure        n/a
Extraction Status             Active
Current Log to Extract        1051
Current LSO                   78818610
Current LSN                   0x0000000000073E38

StartLSN         StartLSO  State      Filename
0000000000073802 78666801  0x00000000 S0001051.LOG
```

**db2diag.log:**

```
EDUID   : 24                  EDUNAME: db2loggr (SAMPLE)
FUNCTION: DB2 UDB, data protection services, sqlpgadf,
DATA #1 : <preformatted>
Active log S0001051.LOG has not been extracted from yet.

EDUID   : 24                  EDUNAME: db2loggr (SAMPLE)
FUNCTION: DB2 UDB, data protection services, sqlpgadf,
DATA #1 : <preformatted>
Current log extraction information:
          loggxLastProcessedLsn = 0000000000073801
          loggxLastProcessedLso = 78666799
     logExtractionCurrentExtNum = 1051
        logExtractionState = IDLE
          logExtractionFlushLsn = 0000000000000000
                throttleReason = DISK_FULL
```

- Extraction induced or outside induced?

53

With extraction running as per the Extraction Status set to Active, your workload still hits transaction log full. You run the db2pd -logs command and it shows no error but Current Log to Extract is equal to the first active log in the active log path still, which usually is a sign that extraction is stalled in some fashion.

By going to the db2diag.log and finding the SQLP_NOSPACE error, you see that the extraction scan is being throttled due to a disk full situation. Look into resolving the disk space issue, at which point extraction will begin again.

Disk space issue can be extraction induced or outside induced. If extraction induced, you may want to see what the extraction filter rate is or what the disk consumption of the extraction log files are. It may be possibly that the workload has caused extraction to extract too much.


Your workload still hits transaction log full. You run the db2pd -logs command and it shows the Extraction Status state as Error.

By going to the db2diag.log and finding the SQLP_NOSPACE error, you see that the extraction scan is being throttled due to a scan error situation. Before the SQLP_NOSPACE error, the extraction scan records the error:

```
EDUID    : 79                    EDUNAME: db2loggx (SAMPLE)
FUNCTION: DB2 UDB, data protection services,
sqlpLogExtractionScanCB::loggxSetScanError, probe:1374
MESSAGE : ZRC=0xFFFFFFFF=-1
DATA #1 : <preformatted>
Log extraction scan error.
                  Function = sqlpshrScanNext
       File Array Element 0 = 1073
                 Head Extent = 1050
            Group Head Extent = 1050
       loggxScanStartExtNum = 1079
          loggxScanStartLsn = 0000000000074AC5
  loggxMinLsnToStartOnError = 0000000000074AF3
   loggxLastProcessedExtNum = 1079
      loggxLastProcessedLsn = 0000000000074AF1
      loggxLastProcessedLso = 79139424
     loggxLastProcessedByte = 79139471
 logExtractionCurrentExtNum = 1079
```

```
logExtractionPendingReadLso = 79139471
       logExtractionReadLso = 79123332
```

Most likely this is not something you can resolve and you will need to contact IBM Support if the issue becomes problematic.

Once the oldest transaction completes, the issue will resolve itself.

# Advanced Log Space Management – Problem Analysis
# Still hit transaction log full, why? (4|4)

- **Bufferpool flushing / dirty pages (minbuff)**
  - db2pd -db sample -dirtypages | grep minbuflsn
    - minbuflsn : 0000000000073802
  - db2flsn -db sample 0000000000073802
    - Given LSN is in log file S0001060.LOG

- Heavy workload and flushing parameters not well tuned?
- Monster transaction?
- Need manual FLUSH BUFFERPOOLS statement?

**db2diag.log:**

```
EDUID   : 24                  EDUNAME: db2loggr (SAMPLE)
FUNCTION: DB2 UDB, data protection services, sqlpgadf,
DATA #1 : <preformatted>
Active log S0001060.LOG has not been extracted from yet.

EDUID   : 24                  EDUNAME: db2loggr (SAMPLE)
FUNCTION: DB2 UDB, data protection services, sqlpgadf,
DATA #1 : <preformatted>
Current log extraction information:
          loggxLastProcessedLsn = 0000000000073801
          loggxLastProcessedLso = 78666799
     logExtractionCurrentExtNum = 1060
           logExtractionState = IDLE
       logExtractionFlushLsn = 0000000000074801
             throttleReason = SLOW_BP_FLUSH
```

55

With extraction running as per the Extraction Status set to Active, your workload still hits transaction log full. You run the db2pd -logs command and it shows no error but Current Log to Extract is equal to the first active log in the active log path still, which usually is a sign that extraction is stalled in some fashion.

By going to the db2diag.log and finding the SQLP_NOSPACE error, you see that the extraction scan is being throttled due to a slow bufferpool flush situation. Look into resolving this, at which point extraction will begin again.

Slow bufferpool flushing can be due to a mis-configured database and/or heavy workload, like a monster transaction. Maybe even a manual FLUSH BUFFERPOOLS statement is required.

## Agenda

- Refresher – Log Management Basics
- What's New in Log Management
  - Version 11.1 and 11.5
- **Advanced Log Space Management**
  - Overview
  - Monitoring and Problem Analysis
  - **Demo**
- The Future in Log Management

# Advanced Log Space Management – Demo

- https://tinyurl.com/y5prhu85

## Agenda

- Refresher – Log Management Basics
- What's New in Log Management
  - Version 11.1 and 11.5
- Advanced Log Space Management
  - Overview
  - Monitoring and Problem Analysis
  - Demo
- **The Future in Log Management**

# Advanced Log Space Management – The Future (1|3)


- **Full crash recovery support**
  - Redo phase make use of extraction log files
  - Redo / undo performance

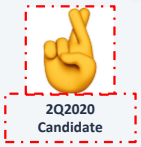
2Q2020 Candidate

- **HADR support**
  - Primary has runtime extraction in progress which requires a log space X
  - Standby does not have these same extraction log files, but would like the same log space requirement
  - Standby has the additional challenge of replay falling behind log shipping, so the disk space challenge on standby is more
    - How to manage when extraction alone is not able to relieve the disk space
      - Drop out of PEER or do we hold up transactions on primary?

2Q2020 Candidate

# Advanced Log Space Management – The Future (2|3)


- **Online backup and database rollforward support**
  - Include extraction log files in backup image
    - When we grab starting point information, include extraction log files from current snapshot (e.g. lowtran to minbuff)
  - Database rollforward makes use of extraction log files initially
  - What to do once extraction log files all read but more to replay?
    - Will retrieve log files but what if not enough disk space to hold all retrieved log files needed for undo?




- **pureScale support**
  - Integration with active log file management
    - Runtime and merged log recovery operations (e.g. group crash recovery / database rollforward)

# Advanced Log Space Management – The Future (3|3)


- **Mirror logging support**
  - Extraction log files in both primary and mirror log path
  - Options?
    - Mirror extraction log files in both log paths
    - One version of extraction log files but exists in whatever path is healthy
- **On by default in future mod pack or release**
  - Need to behave “well” for all workloads

## log_disk_cap - Active log space disk capacity configuration parameter (1|2)


- **Defined in 11.5 GA but not supported until a later time**
- **Allows you to specify the maximum disk capacity for storing transaction log records in the active log path:**
  - Active and extraction log files needed for inflight transactions
  - Inactive log files that have not been archived yet (and not moved to failarchpath)
  - Retrieved log files (if overflowlogpath parameter is not set)

## log_disk_cap - Active log space disk capacity configuration parameter (2|2)


- logprimary / logsecond used as guidance
- The number of files created on disk for logging of inflight transactions might be adjusted based on other consumption
- logfilsz is still used to specify the size of the active log files
- Primary and mirror log paths should be able to hold this amount
- DPF/MPP and pureScale all partitions/members should be able to hold this amount

## Resources

- **IBM Data & AI (formerly Analytics) Ideas (RFEs)**
  - https://ibmanalytics.ideas.aha.io/?project=DB24LUW
- **D09 – Db2 Logging 101 – What Is Really Happening Under the Covers**
  - Dale McInnis
- **My Blog - Db2 Availability & Recovery Insider**
  - https://www.idug.org/p/bl/et/blogid=703


Please complete your evaluations before leaving.

Michael Roecken is a senior software developer with Db2 for Linux, UNIX, and Windows platforms at the IBM Toronto Lab. Michael has worked since 2000 designing, implementing and supporting various features and capabilities in the areas of: backup/restore, crash/rollforward recovery, high availability/disaster recovery, and logging/transaction management.

Please connect with Michael on Twitter at @roecken and LinkedIn.