Demystifying Logging &Recovery in DB2 for LUW

Tridug Meeting - June 18th, 2014

Kelly Schlamb (kschlamb@ca.ibm.com)
Executive IT Specialist, Worldwide Information Management Technical Sales
IBM Canada Ltd.

Agenda

- Logging Basics and Concepts
- Configuring the Active Log Space
- Monitoring Logging Activity

Logging Basics

- **Transaction logs written by DB2 to record updates to database**

- **Used during**
  - Rollback
    - Reverse changes made by a statement or transaction
  - Crash recovery
    - Redo committed work and undo uncommitted work to make database consistent
  - Rollforward recovery
    - Re-apply changes after a restore is performed

- **DB2 uses a Write-Ahead-Logging (WAL) protocol**
  - Log records always written to disk before affected data pages hit disk

Logging Basics (cont.)

- **Each log record identified by a Log Sequence Number (LSN)**
  - Ordering and addressing scheme for log records
  - Assigned when log record is written
  - Page LSN field in data page header updated with the LSN of the log record associated with the most recent change to the page
    - During recovery, allows us to tell what changes have already been made to a page
    - Think of it like a version number for the page that represents the time of the last update to it

- **Log records are initially written into an in-memory log buffer but are then flushed to disk when one of the following things happens**
  - The log buffer is full
  - A transaction commits (must maintain durability in case of system failure)
  - Data pages are written to disk (to ensure uncommitted changes to the page can be undone after system failure)

Logging Basics (cont.)

- **Log records are written into log files**
  - First two pages of each log file are reserved for metadata
  - Remaining pages are used to hold log records
  - # of 4KB pages in a log file is: `LOGFILSIZ + 2`

- **Log files are numbered starting with `S0000000.LOG`**

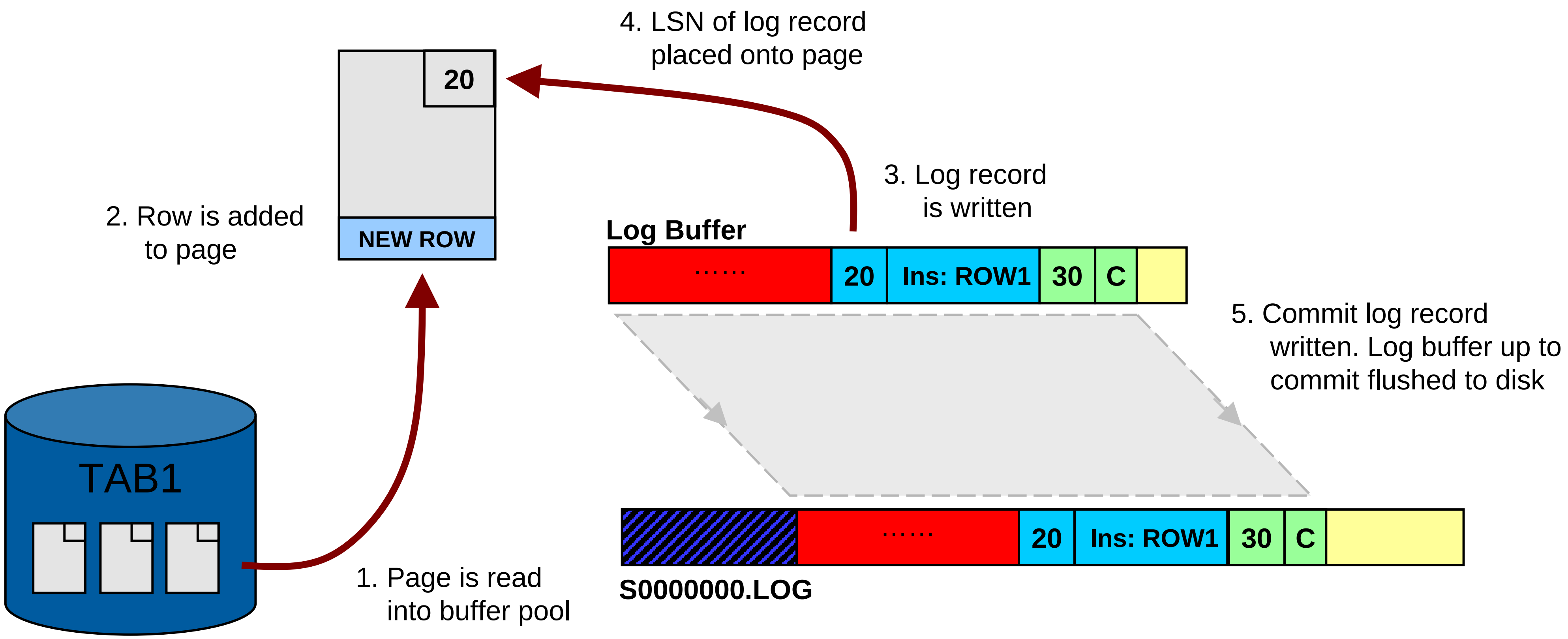
- **When log records are written by transactions, extra space is reserved in case a rollback occurs**
  - Undo (a.k.a. compensation) log records are written during rollback processing
  - May require as much space as corresponding redo log records but typically less

- **In DPF, each database partition has its own set of log files**

- **In pureScale, each member has its own set of log files**

Example of Writing Log Records

```
INSERT INTO TAB1 VALUES ('ROW1'); COMMIT;
```


4. LSN of log record placed onto page
20
NEW ROW
2. Row is added to page
3. Log record is written
Log Buffer
......
20
Ins: ROW1
30
C
5. Commit log record written. Log buffer up to commit flushed to disk
TAB1
1. Page is read into buffer pool
......
20
Ins: ROW1
30
C
S0000000.LOG

Note: Pre-9.8 log records do not contain the LSN. Done here for illustrative purposes.

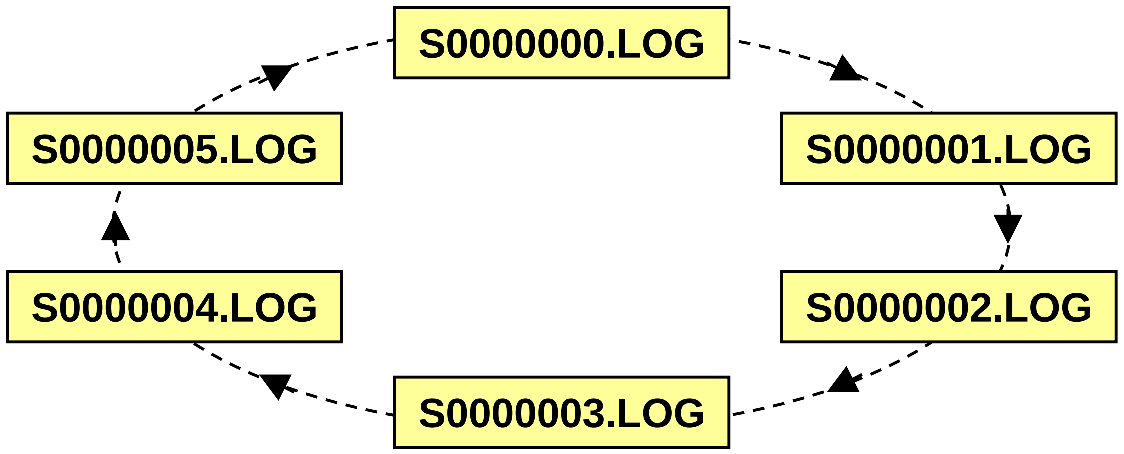
Recoverability – Circular Logging

- **Default for newly created databases**

- **Contents of log files are not permanent**
  - Log data is overwritten when log records are no longer needed for crash recovery purposes

- **Only offline database backups are permitted**

- **Rollforward not permitted**

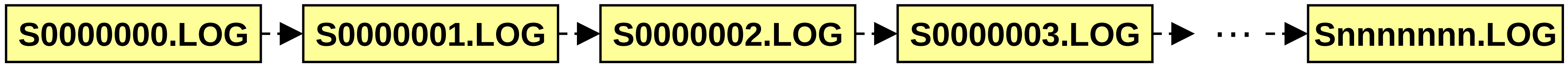


Recoverability – Recoverable/Archived Logging

- **Enabled by setting `LOGRETAIN`, `USEREXIT`, or `LOGARCHMETH1/2` database configuration parameters**
  - `LOGRETAIN/USEREXIT` parameters no longer exist in DB2 10.1
    - Use `LOGARCHMETH1` instead

- **Log files are retained (never reused) and can be archived**

- **Permits the use of**
  - Online backup
  - Table space backup and restore
  - Database and table space rollforward
  - Incremental backup
  - Rebuild from table space backups



Recoverability – Recoverable/Archived Logging (cont.)

- **For archived logging, consider using automated log file management**
  - Choose how long to retain recovery objects like backups and archived logs and when to automatically prune them
  - See `NUM_DB_BACKUPS`, `REC_HIS_RETENTN`, and `AUTO_DEL_REC_OBJ` database configuration parameters

- **Log archival compression (added in DB2 10.1)**
  - Log files can be compressed when they are archived
    - Applicable to disk, TSM, and vendor archiving methods
  - Enabled via `LOGARCHCOMPR1/2` database configuration parameters
  - Even if data inside log records is compressed, still see good (2-3x) compression on the log files
  - Still looks like a log file, but DB2 knows its compressed

Logging EDUs

EDU Name	Description
`db2loggr`	Log reader; also responsible for other things (soft checkpoints, reclaiming log space, suspending logging, etc)
`db2loggw`	Log writer
`db2logts`	Tracks what table spaces have log records in what log files (so unnecessary log files can be skipped during table space rollforward)
`db2lfr`	Log file reader for processing individual log files; used as an LFR/Shredder pair
`db2shred`	Shredder; extracts log record from log pages received from the LFR EDU (one per scan)
`db2redom`	Processes redo log records during recovery and assigns them to redo workers for processing
`db2redow`	Processes redo log records during recovery at the request of the redo master; multiple redo workers can exist
`Db2logmgr`	Log manager. Manages log files (archives/retrieves) for a recoverable database. Also does log file preallocation for `LOGRETAIN=ON` databases
`db2logalloc`	Asynchronous log file allocator (added in DB2 10.5)

Database Configuration Parameters for Logging

- **NEWLOGPATH (Log Path)**
- **MIRRORLOGPATH**
- **OVERFLOWLOGPATH** *

- **LOGRETAIN** *(removed in DB2 10.1)*
- **USEREXIT** *(removed in DB2 10.1)*
- **LOGARCHMETH1/2** *
- **LOGARCHCOMPR1/2** * *(new in DB2 10.1)*
- **LOGARCHOPT1/2** *
- **NUMARCHRETRY** *
- **ARCHRETRYDELAY** *
- **FAILARCHPATH** *

- **CUR_COMMIT**

- **LOGBUFSZ** **
- **LOGPRIMARY** **
- **LOGSECOND** * **
- **LOGFILSIZ** **

- **BLK_LOG_DSK_FUL** *
- **MAX_LOG** *
- **NUM_LOG_SPAN** *
- **MINCOMMIT** * ** *(ignored in DB2 10.1)*
- **BLOCKNONLOGGED**

- **LOG_APPL_INFO** *(new in DB2 10.1)*
- **LOG_DDL_STMTS** * *(new in DB2 10.1)*

- **SOFTMAX** ** *(deprecated in DB2 10.5)*
- **PAGE_AGE_TRGT_MCR** *(new in DB2 10.5)*
- **PAGE_AGE_TRGT_GCR** *(new in DB2 10.5)*

* Can be changed dynamically

** Updated by Configuration Advisor (AUTOCONFIGURE)

Rollforward Recovery

- **Only supported for recoverable databases**

- **Database rollforward**
  - Can be performed following a database restore
  - Can rollforward to end-of-logs or to a point-in-time
  - Offline operation

- **Table space rollforward**
  - Can be performed following a table space restore
  - Can rollforward to end-of-logs or to a point-in-time (at least to MRT)
  - Can be online

- **Logs are used to redo work and then subsequently undo any in-flight changes that exist at the end of the rollforward**

Crash Recovery

- **Crash recovery (a.k.a. restart recovery) required when database terminated abnormally**
  - Performed explicitly using `RESTART DATABASE` command – or –
  - Performed implicitly during first connect if `AUTORESTART= ON`
- **Occurs in two phases**
  - Redo phase: Logs are used to redo work that may not have yet been persisted to disk
  - Undo phase: Uncommitted (in-flight) work is rolled back
- **Crash recovery is an offline operation**
  - Connections are blocked until recovery completes
- **In DPF, crash recovery done on a per-database partition basis**
  - If non-catalog partition being recovered, will subsequently connect to catalog partition, driving crash recovery there if necessary
- **In pureScale, two types of crash recovery exist**
  - Member crash recovery: Individual recovery of member after member failure
  - Group crash recovery: Recovery of entire cluster after cluster failure

Monitoring Recovery Progress

- **`LIST UTILITIES SHOW DETAIL`**

```
ID                             = 1
Type                           = CRASH RECOVERY (or ROLLFORWARD RECOVERY)
Database Name                  = SAMPLE
Member Number                  = 0
Description                    = Crash Recovery (or Rollforward Recovery)
Start Time                     = 04/29/2014 15:20:05.646020
State                          = Executing
Invocation Type                = User
Progress Monitoring:
   Estimated Percentage Complete = 75
   Phase Number [Current]      = 1
      Description              = Forward
      Total Work               = 163834 bytes
      Completed Work           = 124145 bytes
      Start Time               = 04/29/2014 15:20:05.646121
   Phase Number                = 2
      Description              = Backward
      Total Work               = 163834 bytes
      Completed Work           = 0 bytes
      Start Time               = Not Started
```

- **Similar information also available through db2pd `-recovery`**

Parallel Recovery

- **Log records are replayed in parallel by multiple recovery agents**
- **Applies to both crash recovery and rollforward recovery**
- **# of recovery agents is equal to # of logical CPUs**
  - Minimum of 3 recovery agents
  - One becomes the redo master, remainder are redo workers
  - Log records are read and placed onto queues for workers to handle

Log Paths and Metadata Files

- **Default log path**
  - 9.7: `<dbPath>/<instance>/NODE####/SQL#####/SQLOGDIR`
  - 10.1: `<dbPath>/<instance>/NODE####/SQL#####/LOGSTREAM####`

- **Non-default log path (if changed using `NEWLOGPATH`)**
  - 9.7: `<newLogPath>[/NODE####]`
  - 10.1: `<newLogPath>/NODE####/LOGSTREAM####`
  - The log path contains a tag file called `SQLLPATH.TAG`

- **Database directory contains metadata files associated with logging**

```
<dbPath>/<instance>/NODE####/SQL####/SQLOGCTL.GLFH.1/2
<dbPath>/<instance>/NODE####/SQL####/MEMBER####/SQLOGCTL.LFH.1/2
                                                 .../SQLOGMIR.LFH
```

Log Paths

- **Current active log path**
  - Read-only configuration parameter that shows path to active log files
  - Changed by setting `NEWLOGPATH` database configuration parameter
  - Default is `SQLOGDIR` in database directory
  - Raw logs are deprecated (so avoid using)
    - Raw logs can't be used with pureScale

- **Mirrored log path**
  - Second set of log files are maintained in a separate directory from the main log path
  - Set using the `MIRRORLOGPATH` database configuration parameter
  - Minor overhead associated with maintaining two sets of log files

Log Paths (cont.)

- **Overflow log path**
  - Used to specify where archived log files can be found (beyond typical archive location)
  - Can be set for the database using the `OVERFLOWLOGPATH` database configuration parameter
  - `ROLLFORWARD` and `RECOVER` commands also have an `OVERFLOW LOG PATH` option

- **Archive log path(s)**
  - Can configure one or two target locations for saving log files
  - Set using `LOGARCHMETH1` and `LOGARCHMETH2` database configuration parameters
  - Supports disk path, TSM, or vendor library
  - Logs are eligible for archiving when full or after truncation (e.g. online backup, `ARCHIVE LOG` command, clean database shutdown)

Agenda

- Logging Basics and Concepts
- Configuring the Active Log Space
- Monitoring Logging Activity

Configuring the Active Log Space

- **Active log space is determined by**
  - `((# primary log files + # secondary log files) x log file size)`

- **At a minimum, total active log space should be configured to accommodate**
  - All log records generated by the longest running transaction
  - Log records generated by all transactions executed concurrently within the span of the longest running transaction

- **Allocate more to avoid “out of log space” errors during peak usage**
  - Strategies can be employed to limit impact of long running transactions and affect on crash recovery time (e.g. `NUM_LOG_SPAN, MAX_LOG`)

- **With a total size chosen, pick an appropriate log file size and then determine the number of log files needed**

Choosing the Log File Size

- **Size of each primary & secondary log file determined by `LOGFILSIZ` database configuration parameter**

- **Maximum log file size is 4 GB (or 2 GB if pre-DB2 9.5 FP3)**

- **Consider the following when choosing the size:**
  - Frequency of archiving required (e.g. for DR and log shipping)
  - Limit of 256 logs so very large active log space requires large files
  - The larger the file, the longer it takes to physically allocate it, so applications may have to wait longer before a file can be used
    - DB2 tries to avoid this where possible by pre-allocating and renaming old files instead of allocating

- **Minimum suggested size is 20 - 50 MB**

- **Large is good provided that archiving requirements, etc. are met**

Number of Log Files

- **Determined by `LOGPRIMARY` and `LOGSECOND` database configuration parameters**

- **`LOGPRIMARY`: Number of primary log files**
  - Pre-allocated log files, always exist

- **`LOGSECOND`: Number of secondary log files**
  - Allocated on-demand when not enough space available in primary logs (and freed over time as they are no longer required)
  - Recommended setting is `0` (allocate all log files as primary)
  - Can be set to `-1` to enable infinite logging (more on this later)

- **Rule: `(LOGPRIMARY + LOGSECOND) <= 256`**

Configuring the Log Buffer Size

- **Specifies size of memory buffer used to hold log records before they are written to disk**
  - Set using `LOGBUFSZ` database configuration parameter

- **Log records are written to disk when one of the following actions occur**
  - A transaction commits
  - The log buffer is full
  - A data page is written disk (requiring all log records up to the page LSN to be written out to disk as well)
  - Some other internal database manager event

- **Size of buffer impacts performance for logging, rollback, and currently committed processing**
  - Impact is more significant for OLTP workloads

Configuring the Log Buffer Size (cont.)

- **Optimal value depends on workload and amount of concurrent I/U/D activity**
- **Default is way too low, but can be autoconfigured**
- **Consider setting to at least a couple of MB, but something in the 16MB – 64MB range is reasonable to start with**
- **Monitor for "log buffer full" conditions**
  - Watch `NUM_LOG_BUFFER_FULL` monitor element, want to keep this value low
- **Bigger isn't always better**
  - A very large buffer could hurt performance in some cases, so test!
- **Allocated from database heap, so adjust accordingly**

Configuring the Log Path Storage

- **Avoid sharing storage with other database objects**
  - Use dedicated storage with sufficient throughput capacity

- **RAID-5 or RAID-10 for best protection and performance**
  - Small strip size – e.g. 8KB
  - Spindles matter – don't skimp on the drives

- **Enable storage write cache and cache mirroring on the SAN**

- **SSDs for logs may or may not help the performance of the system**
  - Log I/Os may hit the storage cache anyway
  - However, if system is log bound then it could make a big difference
    - Long commit times, long log write times
  - Not usually the case for warehouse environments, but more common in OLTP

- **Raw logs have been deprecated – so best not to use them**
  - Direct I/O has similar performance characteristics
    - Used for runtime processing by default in 9.7
    - See `DB2_LOGGER_NON_BUFFERED_IO` registry variable for details
      - Default is `AUTOMATIC`, which means active logs are open as non-buffered (direct) but other access (crash recovery, rollforward, archiving) uses buffered I/O

Infinite Logging

- **No limit on the amount of log space that can be consumed by active transactions**
- **Enabled by setting `LOGSECOND` to `-1` (can be set dynamically)**
- **Database must also be configured for archive logging**
  - By setting `LOGARCHMETH1` to one of `DISK`, `TSM`, `VENDOR` or `USEREXIT`
- **A log file is archived once it has been filled, but may be kept in active path longer (to avoid possible retrievals)**
- **Consider setting `MAX_LOG` and `NUM_LOG_SPAN` to prevent errant applications from keeping the first active log file too far in the past**

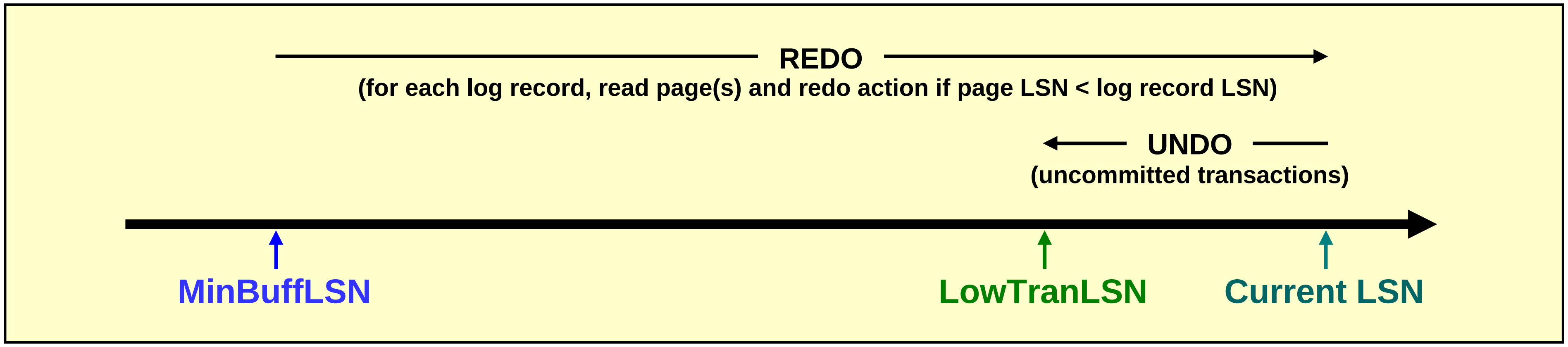
Infinite Logging: Implications of Use

- **Log files may need to be retrieved during rollback and crash recovery**
  - Depending on number of log files to process (and need for retrieval) it may take a very long time to perform crash recovery

- **Compensation log space is not reserved for potential rollback of transactions**
  - A full log path and archive location can result in failures trying to write undo log records, bringing the database down
  - `BLK_LOG_DSK_FUL` should be enabled to avoid this

Tuning Crash Recovery

- **Redo starting point is minimum of (MinBuffLSN, LowTranLSN)**
  - MinBuffLSN represents the oldest change to a page in the buffer pool that has not been written to disk yet
  - LowTranLSN represents the oldest active uncommitted transaction (specifically, the LSN of the first log record it wrote)


REDO
(for each log record, read page(s) and redo action if page LSN < log record LSN)
UNDO
(uncommitted transactions)
MinBuffLSN
LowTranLSN
Current LSN

- **SOFTMAX configuration parameter influences how far back MinBuffLSN is**
  - Value is percentage of one log file (for example, 200 = 2 log files)
  - Page cleaners triggered to write pages out if MinBuffLSN too far back (“LSN gap”)
  - Lower value means shorter recovery time, but increased page cleaning activity
  - Higher value means longer recovery time, but fewer page writes
  - SOFTMAX deprecated in DB2 10.5 (more details later)

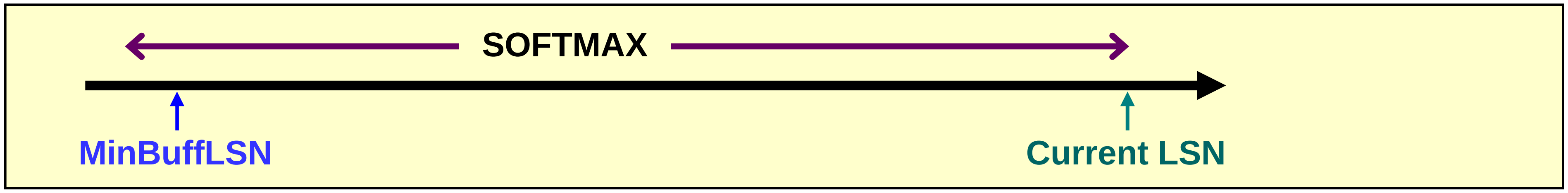
Tuning Crash Recovery (cont.)

- **LowTranLSN is in your hands**
  - Based on age of the transactions
  - Keep transactions short, commit frequently

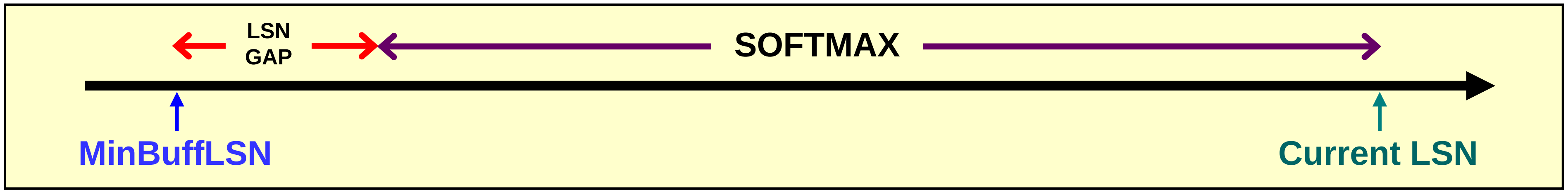
- **If LowTranLSN < MinBuffLSN then we'll still replay those log records in between, but only for the purposes of rebuilding the transaction table**
  - All data changes have already been persisted to disk, so no need to replay them

Tuning Crash Recovery (cont.)

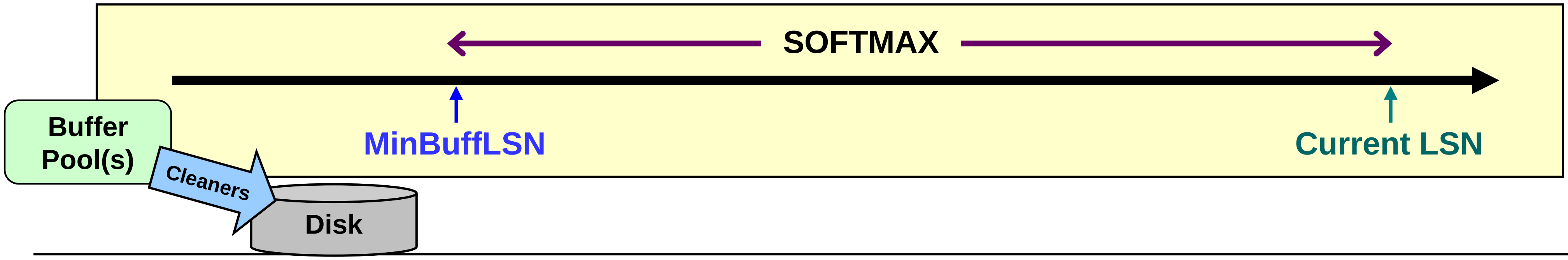
- Current state of database



- More work is done, moving Current LSN out



- Page cleaners react by cleaning pages in the LSN gap that fall outside of the `SOFTMAX` range – moving MinBuffLSN up



Tuning Crash Recovery in DB2 10.5

- **Same principles apply, but new database configuration parameters replace `SOFTMAX`**
  - `SOFTMAX` deprecated as of DB2 10.5
  - `SOFTMAX=0` means use new parameters
    - Set by default for new databases created in DB2 10.5
    - Old value maintained for upgraded databases

- **`PAGE_AGE_TRGT_MCR`**
  - Target duration (in seconds) for changed pages to be kept in the local buffer pool before being persisted to disk (or to the group buffer pool (GBP) in pureScale)
  - Applicable to both non-pureScale and pureScale environments

- **`PAGE_AGE_TRGT_GCR`**
  - Target duration (in seconds) for changed pages to be kept in the GBP before being persisted (cast out) to disk
  - Applicable to pureScale environments only

Agenda

- Logging Basics and Concepts
- Configuring the Active Log Space
- Monitoring Logging Activity

Monitoring Logging Activity

- **Many different things that can be monitored**
  - Log full situations
  - Overall database log usage
  - Log space usage at the transaction level
  - Log reads and writes
  - Time spent metrics for logging

Log Full Situations

- **When log full (SQL0964N) situations arise, information can be found in the notification log and db2diag.log**

- **Older dirty (modified) pages in the buffer pool(s)**

  – `ADM1822W The active transaction log is being held by dirty pages. Database performance may be impacted. The rate at which the database work load is generating dirty pages has exceeded the rate at which the page-cleaner agents are writing dirty pages to disk. Dirty pages that have not yet been written to disk are holding the transaction log.`

- **Long running transaction(s)**

  – `ADM1823E  The active log is full and is held by application handle "###".  Terminate this application by COMMIT, ROLLBACK or FORCE APPLICATION.`

Database Log Activity

- **`SYSIBMADM.SNAPDETAILLOG` admin view**
  - RetrOne row per database partition/member is returned
  - Deprecated in DB2 10.5 (replaced by `MON_GET_TRANSACTION_LOG`)

- **`FIRST_ACTIVE_LOG`**
  - File number of the first active log file

- **`LAST_ACTIVE_LOG`**
  - File number of the last active log file

- **`CURRENT_ACTIVE_LOG`**
  - File number of the active log file that DB2 is currently writing to

- **first active <= current active <= last active**

Database Log Space Usage

- `SYSIBMADM.LOG_UTILIZATION` **admin view**
  - One row per database partition is returned
  - Deprecated in DB2 10.5 (replaced by `MON_TRANSACTION_LOG_UTILIZATION`)
- **`LOG_UTILIZATION_PERCENT`**
  - Percentage of the total active log space used
  - Calculated as `(TOTAL_LOG_USED/(TOTAL_LOG_USED + TOTAL_LOG_AVAILABLE))`
- **`TOTAL_LOG_USED_KB`**
  - Total amount of active log space currently used in the database
  - Includes space reserved for compensation (not applicable for infinite logging)
- **`TOTAL_LOG_AVAILABLE_KB`**
  - Amount of active log space in database currently available for use by transactions
  - Includes space from the primary log files and secondary log files (even if not allocated yet)
  - If newer transactions commit – but older uncommitted transactions still exist – then the log space used by those newer transactions is still considered used and is not available (but their compensation space is no longer reserved)
- **`TOTAL_LOG_USED_TOP_KB`**
  - Maximum amount of total log space that has been used in the database at one time

Database Log Space Usage (cont.)

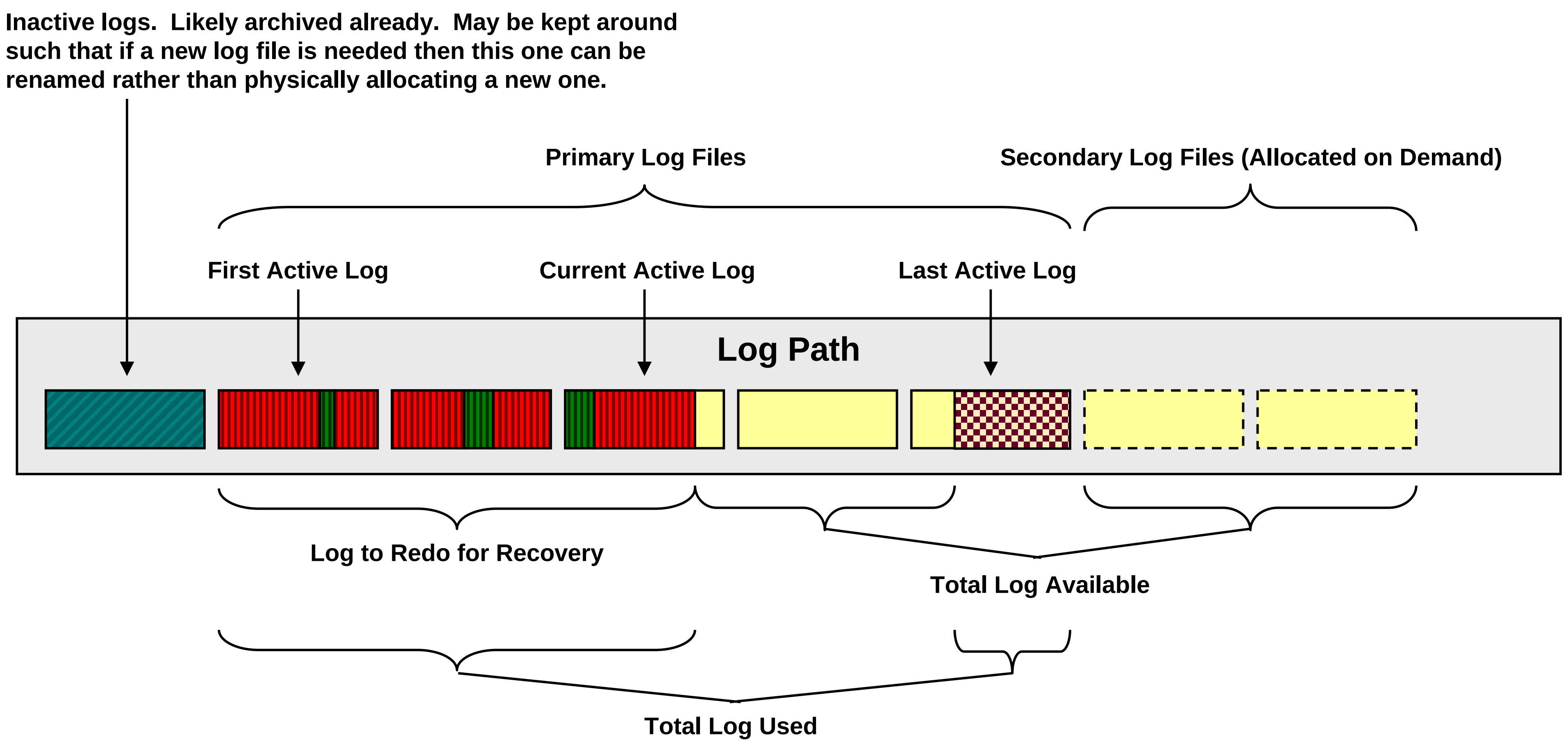
- **SYSIBMADM.SNAPDB admin view**
  - One row per database partition is returned
  - Deprecated in DB2 10.5 (replaced by MON_GET_TRANSACTION_LOG)
- **TOTAL_LOG_AVAILABLE**
  - Same as TOTAL_LOG_AVAILABLE_KB on previous page (but in bytes, not KB)
- **TOTAL_LOG_USED**
  - Same as TOTAL_LOG_USED_KB on previous page (but in bytes, not KB)
- **LOG_TO_REDO_FOR_RECOVERY**
  - Amount of log space (in bytes) that will have to be redone if the database is abnormally terminated and crash recovery is performed
  - Impacted by long running transactions and age of dirty pages in the buffer pool
    - See APPL_ID_OLDEST_XACT for oldest transaction running in the system
- **LOG_HELD_BY_DIRTY_PAGES**
  - Directly related to the age of the oldest dirty page in the database's buffer pool(s)
  - Amount of log space (in bytes) corresponding to the "distance" between the log record of the first update that dirtied that page and the current spot being written to in the logs
  - Impacted by the SOFTMAX database configuration parameter and page cleaning

New Monitor Administrative Views in DB2 10.x

- **New in DB2 10.1: `MON_GET_TRANSACTION_LOG`**
  - Returns information about the transaction logging subsystem for the currently connected database
  - Includes important elements from `SYSIBMADM.SNAPDB` admin view
  - Also includes information previously only available through `db2pd`
    - Currently committed metrics
    - Archive logging status
    - Current LSN
    - Current log chain
    - LSN of oldest active transaction

- **New in DB2 10.5: `MON_TRANSACTION_LOG_UTILIZATION`**
  - Returns information about transaction log utilization for the currently connected database
  - Includes same elements from `SYSIBMADM.LOG_UTILIZATION` admin view such as total log space used and available

Database Log Space Usage – Example



Inactive logs. Likely archived already. May be kept around such that if a new log file is needed then this one can be renamed rather than physically allocating a new one.
Primary Log Files
Secondary Log Files (Allocated on Demand)
First Active Log
Current Active Log
Last Active Log
Log Path
Log to Redo for Recovery
Total Log Available
Total Log Used

🟨 Unused log space	🟥 Logs for uncommitted transactions	▩ Reserved for compensation of uncommitted transactions (not actually tied to a particular location, more like a moving window within the physically allocated log file space
▨ Inactive/archived log files	🟩 Logs for committed transactions	

Log Monitoring with db2pd

```
db2pd -db <dbname> -logs
```

```
Current Log Number            29
Pages Written                 312
Cur Commit Disk Log Reads     0
Cur Commit Total Log Reads    0
Method 1 Archive Status       n/a
Method 1 Next Log to Archive  n/a
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
Log Chain ID                  0
Current LSN                   0x00000000098608D9

Address            StartLSN         State      Size       Pages      Filename
0x07000000796AD950 000000008B28010 0x00000000 1024       1024       S0000026.LOG
0x07000000796AE1B0 000000008F28010 0x00000000 1024       1024       S0000027.LOG
0x07000000796AEA10 0000000009328010 0x00000000 1024       1024       S0000028.LOG
0x070000003038D8D0 0000000009728010 0x00000000 1024       1024       S0000029.LOG
0x070000003038E130 0000000009B28010 0x00000000 1024       1024       S0000030.LOG
0x07000000796A10B0 0000000009F28010 0x00000000 1024       1024       S0000031.LOG
```

Current active log file being written to

LSN where we'll start writing the next log record to; compare to a transaction's first LSN to see "distance" (in bytes) between start of transaction and the current logging location

Starting LSN can show what log a particular transaction starts in

Transaction Log Space Usage

- **Determine how much space is used (already written and reserved) by all of the current write transactions**
  - Requires that the UOW monitor switch be turned on

```
SELECT SUBSTR(AI.PRIMARY_AUTH_ID,1,10) AS AUTHID,
       SUBSTR(AI.APPL_NAME,1,10) AS APPLNAME,
       SUBSTR(AI.APPL_STATUS,1,10) AS STATUS,
       INT(A.UOW_LOG_SPACE_USED) AS LOG_SPACE,
       A.UOW_START_TIME
FROM SYSIBMADM.SNAPAPPL A,
     SYSIBMADM.SNAPAPPL_INFO AI
WHERE A.AGENT_ID = AI.AGENT_ID AND
      A.UOW_LOG_SPACE_USED > 0
ORDER BY A.UOW_LOG_SPACE_USED DESC;
```

```
AUTHID     APPLNAME   STATUS     LOG_SPACE   UOW_START_TIME
---------- ---------- ---------- ----------- --------------------------
KSCHLAMB   db2bp      UOWWAIT       12552621 2014-03-06-15.36.12.891110
KSCHLAMB   db2bp      UOWWAIT          13711 2014-03-06-15.36.15.078040
```

Transaction Log Space Usage (cont.)

```
db2pd -db <dbname> -transactions
```

LSN of the **first** log record written by the transaction;
(Smallest non-0 value indicates oldest write transaction)

LSN of the **last** log record written by the transaction

```
AppHandl TranHdl  Locks   Firstlsn           Lastlsn            SpaceReserved LogSpace
662      2        217090  0x0000000008DCC413 0x00000000098607AA 1758880       12552621
84       24       29      0x000000000909CD4A 0x000000000909E61C 7304          13711
```

Application handle can be used to get further information from application snapshots

Amount of log space reserved (but not used) by the transaction

Total log space used by the transaction including log records written and reserved space

Total log space used by log records written = (LogSpace – SpaceReserved)

Monitoring Log Reads and Writes

- **SYSIBMADM.SNAPDB admin view**
  - One row per database partition is returned
  - Deprecated in DB2 10.5 (replaced by MON_GET_TRANSACTION_LOG)

- **Number of reads should typically be low – relative to writes**

- **log_reads**
  - # of log pages read from disk by the logger
  - Rollback, currently committed processing, etc.

- **num_log_read_io**
  - Number of read requests by the logger for reading log pages from disk

- **log_read_time**
  - Elapsed time spent by the logger reading log pages from disk

- **num_log_data_found_in_buffer**
  - Number of log page accesses that are satisfied by the log buffer
  - Configure LOGBUFSZ to improve log buffer hit ratio

Monitoring Log Reads and Writes (cont.)

- **`log_writes`**
  - Number of log pages written to disk by the logger
  - <u>Not</u> the overall number of log pages produced
    - A particular page may be written multiple times as it is filled

- **`num_log_write_io`**
  - Number of write requests by the logger for writing log data to disk

- **`log_write_time`**
  - Elapsed time spent by the logger writing log data to the disk
  - Includes writing log data to both locations for mirrored logging

Monitoring Log Reads and Writes (cont.)

- **`num_log_buffer_full`**
  - Number of times agents are unable to copy records into log buffer because it is full
    - Log buffer is flushed to disk and agents have to wait
  - Goal is to keep this value low
  - Possible reasons for it being high
    - `LOGBUFSZ` is too small for your environment
    - Workload consists of long running transactions with infrequent commits
    - Log storage cannot keep up with logging demands

- **`num_log_part_page_io`**
  - Number of write requests by the logger for writing a partial log page to disk
  - Included as part of `num_log_write_io` count as well
  - Usually indicates a lot of short frequently committing transactions
  - Informative value, not a lot of reason to monitor

Monitoring Log Reads and Writes (cont.)

- **Average log read time**
  - Calculated as (`log_read_time / num_log_read_io`)
  - Should be low single digit milliseconds during normal operations (ideally 1 millisecond or less)
  - If higher then storage may be under configured

- **Average log write time**
  - Calculated as (`log_write_time / num_log_write_io`)
  - Should be low single digit milliseconds during normal operations (ideally as low as 1-3 milliseconds, but slightly higher is okay too)
  - If higher than single digits then storage may be under configured

Time Spent Metrics for Logging

- **Reported through monitor table functions**
  - e.g. `MON_GET_UNIT_OF_WORK` and `MON_GET_WORKLOAD`

- **`log_disk_waits_total`**
  - Number of times agents have to wait for log data to write to disk
  - e.g. Commit processing, flushing logs when writing data page to disk

- **`log_disk_wait_time`**
  - Amount of time an agent spends waiting for log records to be flushed to disk (in milliseconds)

- **`log_buffer_wait_time`**
  - Amount of time an agent spends waiting for space in the log buffer (in milliseconds)
  - If time seems excessive then consider increasing `LOGBUFSZ`

- **`num_log_buffer_full`**
  - Number of times agents are unable to copy records into log buffer because it is full (log buffer is flushed to disk and agents have to wait – as above)

Example Queries

Q1. How much of the active log space is currently being used?

```
SELECT LOG_UTILIZATION_PERCENT AS "LOGSPC_USED (%)",
       INT(TOTAL_LOG_USED_KB / 1024) AS "LOGSPC_USED (MB)",
       INT(TOTAL_LOG_AVAILABLE_KB / 1024) AS "LOGSPC_FREE (MB)",
       INT(TOTAL_LOG_USED_TOP_KB / 1024) as "MAX_LOGSPC_USED (MB)"
  FROM SYSIBMADM.MON_TRANSACTION_LOG_UTILIZATION;
```

```
LOGSPC_USED (%) LOGSPC_USED (MB) LOGSPC_FREE (MB) MAX_LOGSPC_USED (MB)
--------------- ---------------- ---------------- --------------------
          10.52               26              227                  149

  1 record(s) selected.
```

Example Queries (cont.)

Q2. What is the range of the active log files and the current position in them?

```
SELECT FIRST_ACTIVE_LOG, LAST_ACTIVE_LOG,
  CURRENT_ACTIVE_LOG FROM TABLE(MON_GET_TRANSACTION_LOG(NULL)) AS T;

FIRST_ACTIVE_LOG     LAST_ACTIVE_LOG      CURRENT_ACTIVE_LOG
-------------------- -------------------- --------------------
                  15                   40                   35

  1 record(s) selected.
```

Example Queries (cont.)

Q3. How much of the current log space is being used, how large is the recovery window, and how much log space is being held up by dirty (modified) pages?

```
SELECT (TOTAL_LOG_AVAILABLE / (1024 * 1024)) AS "TOT_LOG_AVAIL (MB)",
       (TOTAL_LOG_USED / (1024 * 1024)) AS "TOT_LOG USED (MB)",
       (LOG_TO_REDO_FOR_RECOVERY / (1024 * 1024)) AS "LOG_REDO_RECOV (MB)",
       (LOG_HELD_BY_DIRTY_PAGES / (1024 * 1024)) AS "LOG_HELD_BY_DP (MB)"
  FROM TABLE(MON_GET_TRANSACTION_LOG(NULL)) AS T;
```

```
TOT_LOG_AVAIL (MB) TOT_LOG USED (MB) LOG_REDO_RECOV (MB) LOG_HELD_BY_DP (MB)
------------------ ----------------- ------------------- -------------------
               229                24                  20                  20

  1 record(s) selected.
```

Example Queries (cont.)

Q4. What is the oldest write transaction running and how much log space is it consuming?

```
SELECT INT(T.APPLID_HOLDING_OLDEST_XACT) AS OLD_APP_ID,
       SUBSTR(AI.PRIMARY_AUTH_ID,1,8) AS AUTH_ID,
       SUBSTR(AI.APPL_NAME,1,8) AS APP_NAME,
       INT(A.UOW_LOG_SPACE_USED) AS LOG_USED_B,
       SUBSTR(AI.APPL_STATUS,1,8) AS STATUS,
       A.UOW_START_TIME
  FROM TABLE(MON_GET_TRANSACTION_LOG(NULL)) AS T,
       SYSIBMADM.SNAPAPPL A,
       SYSIBMADM.SNAPAPPL_INFO AI
  WHERE T.APPLID_HOLDING_OLDEST_XACT = A.AGENT_ID AND
        A.AGENT_ID = AI.AGENT_ID;
```

```
OLD_APP_ID  AUTH_ID  APP_NAME LOG_USED_B  STATUS   UOW_START_TIME
----------- -------- -------- ----------- -------- --------------------------
        241 KSCHLAMB db2bp           1678 UOWWAIT  2014-06-13-14.39.30.958091

  1 record(s) selected.
```

Example Queries (cont.)

Q5. What are the largest consumers of log space running in the database?

```
SELECT SUBSTR(AI.PRIMARY_AUTH_ID,1,8) AS AUTH_ID,
       SUBSTR(AI.APPL_NAME,1,8) AS APP_NAME,
       SUBSTR(AI.APPL_STATUS,1,8) AS STATUS,
       INT(A.UOW_LOG_SPACE_USED) AS LOG_USED_B,
       A.UOW_START_TIME
  FROM SYSIBMADM.SNAPAPPL A,
       SYSIBMADM.SNAPAPPL_INFO AI
  WHERE A.AGENT_ID = AI.AGENT_ID AND
        A.UOW_LOG_SPACE_USED > 0
  ORDER BY A.UOW_LOG_SPACE_USED DESC;
```

```
AUTH_ID  APP_NAME STATUS   LOG_USED_B  UOW_START_TIME
-------- -------- -------- ----------- --------------------------
KSCHLAMB db2bp    UOWWAIT    203518230 2014-06-13-16.26.31.140008
KSCHLAMB db2bp    UOWWAIT     52089768 2014-06-13-15.05.11.691016
KSCHLAMB db2bp    UOWWAIT         6450 2014-06-13-15.42.18.424949

  3 record(s) selected.
```

Example Queries (cont.)

Q6. What are the average log read and log write times (in milliseconds) for the database?

```
SELECT DEC(((((LOG_READ_TIME_S * 1000000000) + LOG_READ_TIME_NS) /
             FLOAT(NUM_LOG_READ_IO)) / 1000000),15,3) AS MS_PER_READ,
       DEC(((((LOG_WRITE_TIME_S * 1000000000) + LOG_WRITE_TIME_NS) /
             FLOAT(NUM_LOG_WRITE_IO)) / 1000000),15,3) AS MS_PER_WRITE
  FROM SYSIBMADM.SNAPDB;
```

```
AVG_MS_PER_READ   AVG_MS_PER_WRITE
----------------- -----------------
            0.583             1.537

  1 record(s) selected.
```

Example Queries (cont.)

Q7. What are the number of pages read per read I/O and number of pages written per write I/O for the database?

```
SELECT DEC((FLOAT(LOG_WRITES) /
          (NUM_LOG_WRITE_IO + NUM_LOG_PART_PAGE_IO)),15,3)
         AS PAGES_PER_WRITE_IO,
       DEC((FLOAT(LOG_READS) / NUM_LOG_READ_IO),15,3)
         AS PAGES_PER_READ_IO
  FROM TABLE(MON_GET_TRANSACTION_LOG(NULL)) AS T;
```

```
PAGES_PER_WRITE_IO PAGES_PER_READ_IO
------------------ -----------------
            20.723             8.000

  1 record(s) selected.
```

Example Queries (cont.)

Q8. When needing to read from the logs, what percentage of the time was the data found in the in-memory buffer versus having to go to disk?

```
SELECT NUM_LOG_DATA_FOUND_IN_BUFFER,
       NUM_LOG_READ_IO,
       DEC((FLOAT(NUM_LOG_DATA_FOUND_IN_BUFFER) * 100 /
            (NUM_LOG_READ_IO + NUM_LOG_DATA_FOUND_IN_BUFFER)),15,2)
         AS PERCENT_LOG_DATA_FOUND_IN_BUFFER
  FROM TABLE(MON_GET_TRANSACTION_LOG(NULL)) AS T;
```

```
NUM_LOG_DATA_FOUND_IN_BUFFER NUM_LOG_READ_IO    PERCENT_LOG_DATA_FOUND_IN_BUFFER
---------------------------- ------------------ --------------------------------
                      243564               2631                            98.93

  1 record(s) selected.
```

Example Queries (cont.)

Q9. How often was the in-memory log buffer pool, resulting in the agent waiting for the log data to be flushed to disk?

```
SELECT NUM_LOG_BUFFER_FULL
  FROM TABLE(MON_GET_TRANSACTION_LOG(NULL)) AS T;
```

```
NUM_LOG_BUFFER_FULL
--------------------
                   9

  1 record(s) selected.
```

Further Reading

- **Lots of other topics related to logging and recovery that we didn't have time to get to today**

- **Logging-related sections in the Information Center**
  - Search for:
    - Database Logging
    - Configuring Database Logging Options
    - Configuration Parameters for Database Logging
    - Data Recovery

Kelly Schlamb (kschlamb@ca.ibm.com)
Executive IT Specialist, Worldwide Information Management Technical Sales Acceleration
IBM Canada Ltd.