

Demystifying Logging and Recovery in DB2 for LUW

Session Number 2230

Maciej Mazur,
mmazur@ca.ibm.com

(Originally presented by
Kelly Schlamb,
kschlamb@ca.ibm.com)

IBM Software

Information On Demand **2011**

Acknowledgements and Disclaimers:



Availability. References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS-IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

© **Copyright IBM Corporation 2011. All rights reserved.**

- **U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.**

IBM, the IBM logo, ibm.com, and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml

Other company, product, or service names may be trademarks or service marks of others.





Agenda

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



Logging Basics

- Log records are used to record changes to the database and are used during
 - Rollback – to reverse changes made by a statement or transaction
 - Crash recovery – to redo and undo work to make the database consistent
 - Rollforward – to re-apply changes after a restore is performed
- Each log record is identified by a Log Sequence Number (LSN)
- LSN of log record gets written to data page after performing a logged operation
 - 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 has committed and so all of its log records must be persisted to disk
 - Data pages are written to disk (to ensure changes to the page can be undone in the case of a crash)



Logging Basics (cont.)

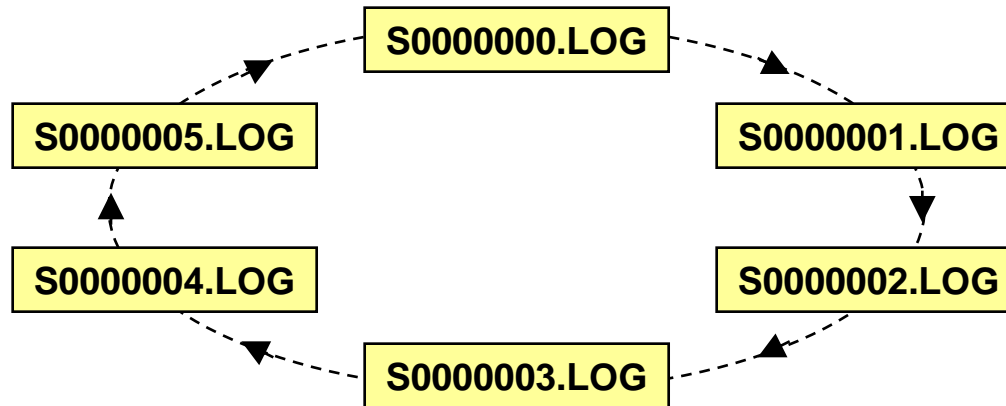
- Log records are written into log files
 - First two pages of each log file are reserved for metadata
 - Remaining pages are used to hold log records
 - # of 4KB pages in a log file is: `LOGFILSIZ + 2`
- Log files are numbered starting with `S0000000.LOG`
- When log records are written by transactions, 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



Types of Logging/Recoverability

- **Circular logging**

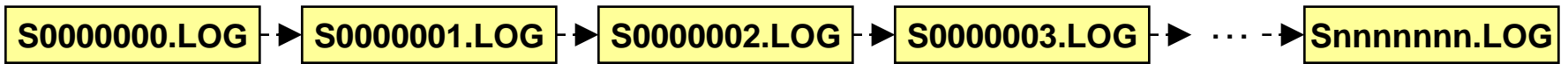
- Contents of log files are not permanent
- Log data is overwritten when log records are no longer needed for crash recovery purposes
- Only offline database backups are permitted
- Rollforward not permitted
- Circular logging is default for newly created databases



Types of Logging/Recoverability (cont.)

- Recoverable/archived logging:

- Enabled by setting LOGRETAIN, USEREXIT, or LOGARCHMETH1 / 2 database configuration parameters
- Log files are retained (never reused) and can be archived
- Permits the use of:
 - Online backup
 - Table space backup and restore
 - Database & table space rollforward
 - Incremental backup
 - Rebuild from table space backups



- 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



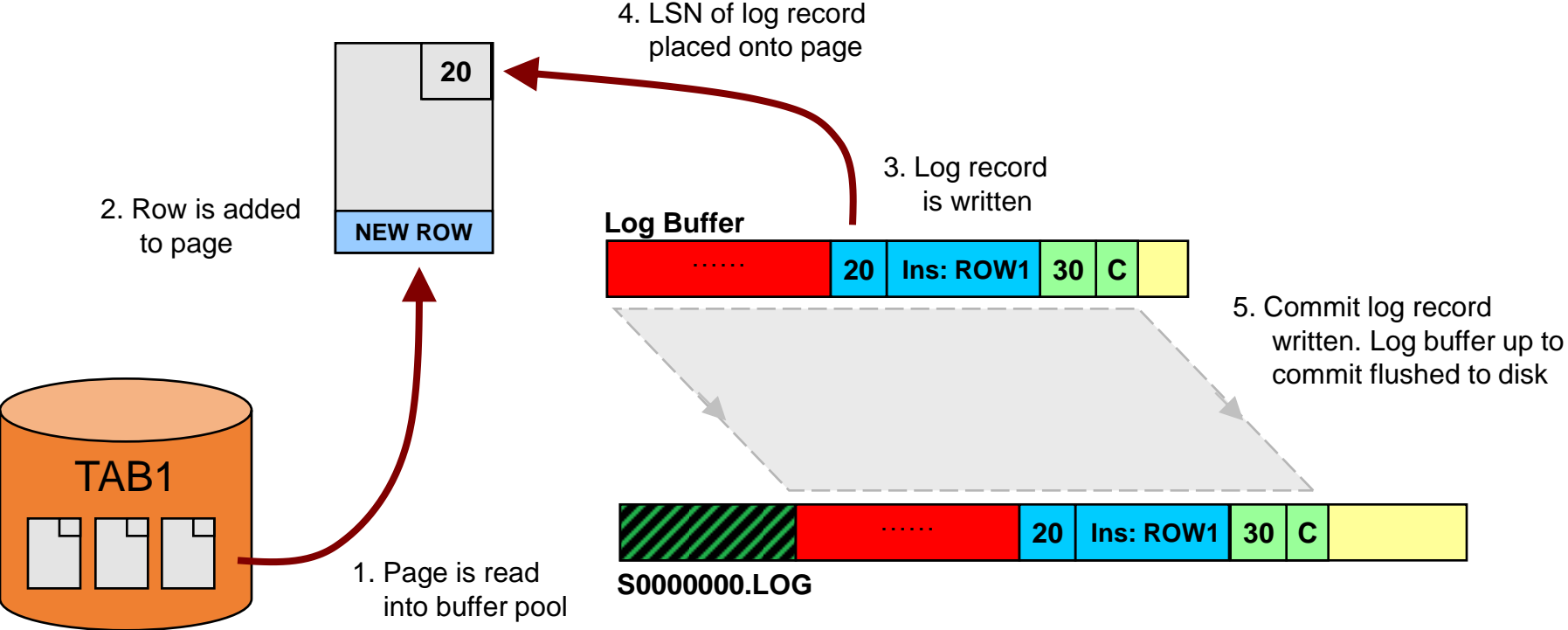
Log Sequence Numbers (LSN)

- Ordering and addressing scheme for log records
- Each log record is assigned an LSN when it is written (ever-increasing value)
- A “page LSN” on pages is 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
- Every log record has a unique LSN
 - Represents the physical address of the log record in a log file (so no need to store actual LSN value in the log record)
 - This has changed in pureScale (DB2 9.8) where it has become a counter and is included
 - LSN was increased from 6 bytes to 8 bytes in DB2 9.7
 - Now stored as native integer type instead of big-endian string of bytes



Example of Writing Log Records

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



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

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

Database Configuration Parameters for Logging

- NEWLOGPATH (Log Path)
- MIRRORLOGPATH
- OVERFLOWLOGPATH *
- LOGRETAIN
- USEREXIT
- LOGARCHMETH1/2 *
- LOGARCHOPT1/2 *
- NUMARCHRETRY *
- ARCHRETRYDELAY *
- FAILARCHPATH *
- LOGBUFSZ *
- LOGPRIMARY *
- LOGSECOND * *
- LOGFILSIZ *
- BLK_LOG_DSK_FUL *
- MAX_LOG *
- NUM_LOG_SPAN *
- MINCOMMIT * *
- SOFTMAX *
- BLOCKNONLOGGED

* 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
 - Redo performed in parallel by recovery master/workers



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` is `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
 - Parallel recovery
- 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



Log Paths and Metadata Files

- Default log path

`<databasePath>/<instance>/NODE####/SQL#####/SQLOGDIR`

- Non-default log path (if changed using `NEWLOGPATH`)

- non-DPF: `<newLogPath>`

- DPF: `<newLogPath>/NODE####`

- The log path contains a tag file called `SQLLPATH.TAG`

- Database directory contains metadata files associated with logging

`<databasePath>/<instance>/NODE####/SQL#####/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 `SQLLOGDIR` in database directory
 - Raw logs are deprecated (so avoid using)
- 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
 - Overhead associated with maintaining two sets of log files
- 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



Log Paths (cont.)

- 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

- Set 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:** $(\text{LOGPRIMARY} + \text{LOGSECOND}) \leq 256$



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 occurs:
 - 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, but here are some recommended starting points (in 4KB pages):

Machine Memory	LOGBUFSZ
< 16 GB RAM	4096
16 – 64 GB RAM	8192
> 64 GB RAM	16384

- Monitor for “log buffer full” conditions
 - Watch `NUM_LOG_BUFFER_FULL` monitor element
- 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
 - 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 will likely make a big difference
 - Not usually the case for warehouse environments, but more common in OLTP
- Raw logs have been deprecated – so don't 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



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 DB2 may actually choose to keep it around 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



Recovery Window

- Database configuration parameter `SOFTMAX` influences the number of log files that need to be recovered following a database crash
 - Value is percentage of one log file (e.g. 200 = 2 log files)
 - `SOFTMAX` also determines the frequency of soft checkpoints but generally considered secondary to the size of the recovery window
- Page cleaners are triggered to write pages to disk that were updated by transaction log records outside of the specified recovery window
 - Referred to as LSN Gap triggers
 - Won't always be exact – page cleaners need to be able to keep up
- A lower `SOFTMAX` value means
 - Shorter recovery time
 - Increased page cleaning frequency
 - Could mean more writes due to cleaning & re-cleaning of frequently updated pages
- A higher `SOFTMAX` value means
 - Longer recovery time
 - Potentially fewer page writes (so possibly better performance)



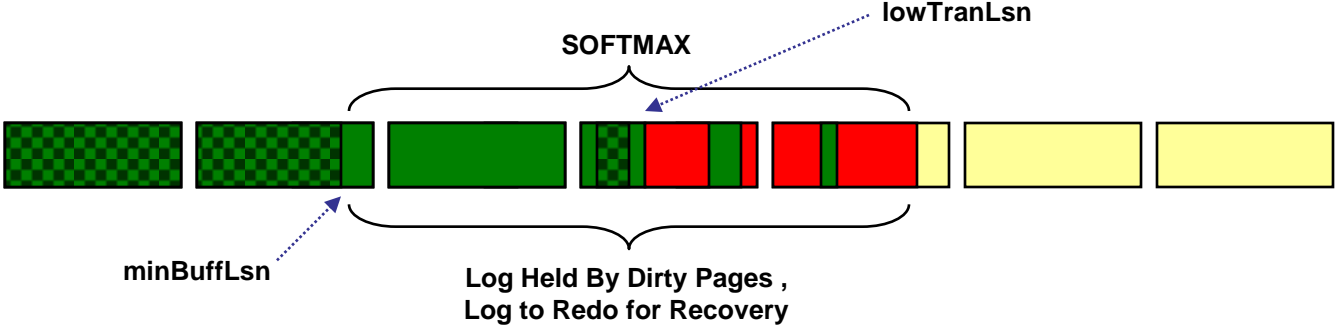
Recovery Window (cont.)

- Also impacted by age of uncommitted write transactions in the system
- Even if `SOFTMAX` set to something aggressive, if old transactions exist then they will hold the recovery window back
 - If held back by $(\text{LOGPRIMARY} + \text{LOGFILSIZ})$ log files then applications will start getting “out of log space” errors (more on this later)
- These ages are represented by LSNs
 - **minBuffLSN**: Oldest change to a data page that is in the buffer pool but has not been persisted to disk yet
 - **lowTranLSN**: Start of oldest write transaction still active (uncommitted) in system

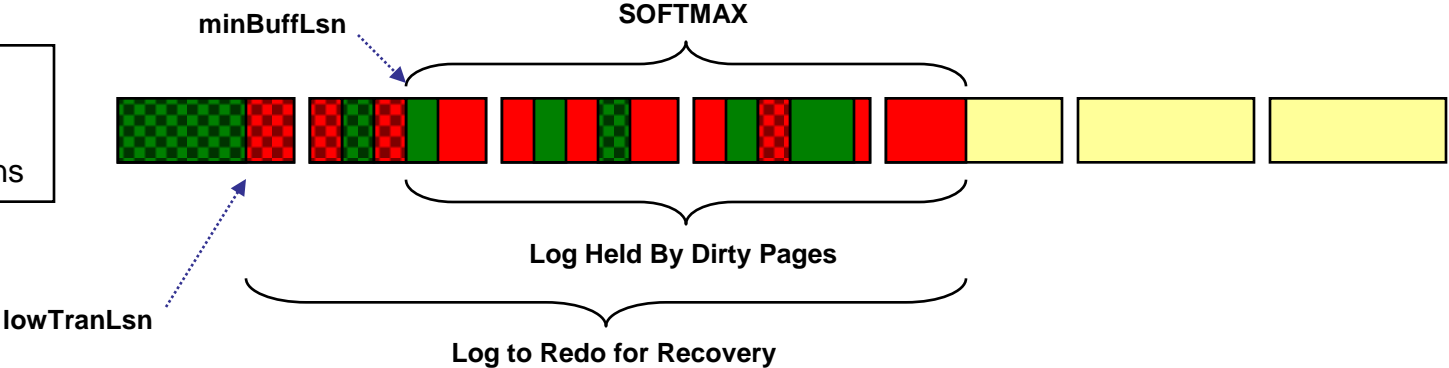


Recovery Window – Examples

Example 1:
 SOFTMAX=300
 Short running transactions



Example 2:
 SOFTMAX=300
 Long running transactions



- Log records for committed transactions
- Log records for uncommitted transactions
- Affected pages already written from buffer pool to disk
- Unused log space



Log Full Situations

- When log full (SQL0964N) situations arise, information can be found in the notification log and db2diag.log
- Old dirty pages in the buffer pool(s):
 - **ADM1822W** The active log is being held by dirty pages. This is not an error, but database performance may be impacted. If possible, reduce the database work load. If this problem persists, either decrease the SOFTMAX and/or increase the NUM_IOCLEANERS DB configuration parameters.
- 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.





Agenda

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





Monitoring Logging Activity

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



Database Log Space Usage

- **SYSIBMADM.SNAPDETAILLOG** admin view
 - One row per database partition is returned
- **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 (cont.)

- **SYSIBMADM.LOG_UTILIZATION** admin view
 - One row per database partition is returned
- **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.)

- **SYSDIBMADM.SNAPDB** admin view
 - One row per database partition is returned
- **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 below)
- **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





Database Log Space Usage (cont.)

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

```

← 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

Address	StartLSN	State	Size	Pages	Filename
0x07000000796AD950	0000000008B28010	0x00000000	1024	1024	S0000026.LOG
0x07000000796AE1B0	0000000008F28010	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

Starting LSN can show what log a particular transaction starts in





Transaction Log Space Usage

```
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	LogSpace	SpaceReserved
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 = (SpaceReserved – LogSpace)



Monitoring Log Reads and Writes

- **SYSIBMADM.SNAPDB** admin view
 - One row per database partition is returned
- 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
 - NOT 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 $(\text{log_read_time} / \text{log_reads})$
 - 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 $(\text{log_write_time} / \text{log_writes})$
 - 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 new monitor table functions in DB2 9.7
 - 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`

