Recent Changes in DB2 for z/OS Logging

Steve Thomas, CA Technologies

BeLux GSE, December 13th 2016

Agenda

- What is logged?
- Reordered Row Format
- Not Logged Tablespaces
- LOBs and XML data
- LRSN Spin Avoidance
- Summary of Logging changes in DB2 9, 10, 11 and 12

What data is logged during a standard UPDATE?

- Assuming Data Capture Changes is not used…
  – If DCC the full image of either the UNDO or REDO record is logged
  – Partial image of the other
- Fixed Data - from first to last updated byte
  – Also true for Variable data if the length does not change
- Variable Data – from first updated byte to end of row
  – Bearing in mind the first updated byte will be the column length field
- Leads to long standing good design practice
  – Locate Variable Columns at the end of the row
  – With the most updated ones towards the back end
  – Locate heavily updated fixed length columns together
- Usually only implemented when table first created
  – Rules tend to get relaxed as columns are added

Reordered Row Format (RRF)

- New Row format introduced in DB2 9 NFM
- Automatically implements some of these best practices
- Columns internally reordered on the data page
- All Fixed Columns at the beginning of the row
- All variable columns stored at the end of the row
- No lengths – uses offsets into row instead
  - Both offsets and lengths are 2 bytes so row length identical
  - Row header is not included in the offset
- Logical Column location in the table unaffected
- No Application impact – not even a Rebind required

Basic Row Format (BRF)

STAFF_NUMBER	INTEGER	NOT NULL
FIRST_NAME	VARCHAR(30)	NOT NULL
MIDDLE_INITIAL	CHAR(1)	NOT NULL
LAST_NAME	VARCHAR(30)	NOT NULL
DATE_JOINED	DATE	NOT NULL

001	05	STEVE	R	06	THOMAS	12/07/1999

2 byte Lengths
(in hex)

<u>Note</u>: Integer and Date columns both use 4 bytes of internal storage

Let’s look at the same row in Reordered Row Format

STAFF_NUMBER	INTEGER	NOT NULL
FIRST_NAME	VARCHAR(30)	NOT NULL
MIDDLE_INITIAL	CHAR(1)	NOT NULL
LAST_NAME	VARCHAR(30)	NOT NULL
DATE_JOINED	DATE	NOT NULL

001	R	12/07/1999	0D	12	STEVE	THOMAS

2 byte Offsets (hex)

Note: Integer and Date columns both use 4 bytes of internal storage

RRF usually leads to reduced logging

001	05	STEVE	R	06	THOMAS	12/07/1999

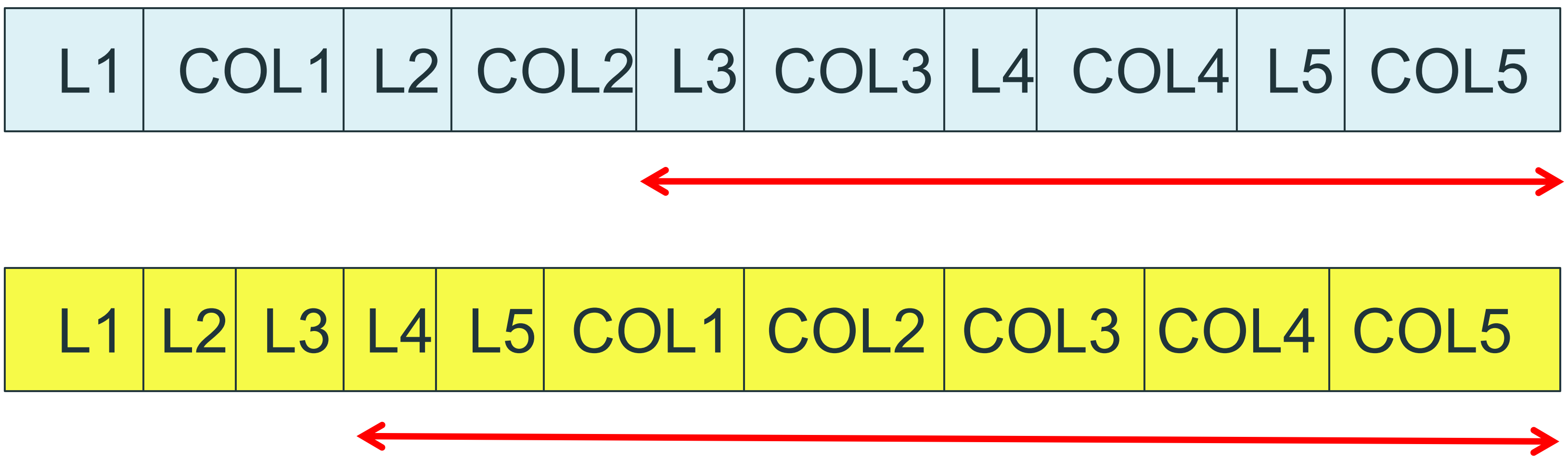
001	06	STEVEN	R	06	THOMAS	12/07/1999

001	R	12/07/1999	0D	12	STEVE	THOMAS

001	R	12/07/1999	0D	13	STEVEN	THOMAS

When might it get worse?

- Multiple Variable columns, Update one of the later ones
  - Example – 5 columns and update only the 3rd



- Mainly impacts non-compressed data - if the data is compressed DB2 usually logs the whole row anyway

- Most customers should gain much more than they lose

Test Results verify this

- What we did…
  - Created a table with 10 columns
    - 7 were VARCHAR of various sizes
  - Loaded 1,000 rows using both BRF and RRF
  - Updated the various columns on every row
  - Measured the results using a Log Analysis tool
- In most cases log volume between flat and 50% lower
  - Exception was updating the last columns
  - Worst case example we managed to double the log volume
    - Extreme example designed to cause maximum pain
- Conclusion: RRF should help in most cases
  - Beware situations with multiple VARCHAR columns

Not Logged Tablespaces

- Added to provide an SQL equivalent to LOG NO utilities
  - Reduce Logging volume when data can be rebuilt
  - Never intended to be a performance option
- Prevents Logging of UNDO and REDO records
  - Control records are still logged (OPEN, CLOSE etc.)
  - So are Open URID records
    - Needed for Data Sharing and Long running URID messages
- Cannot be set for Catalog, XML or Workfiles
- Ideal applications include MQTs & Summary objects
- Default is always to Log updates
  - Except for Workfile tablespaces

Not Logged Tablespaces

- Incompatible with Data Capture Changes
- No SYSLGRNX entries created
- No SHRLEVEL CHANGE Copy or Reorg
  - Must use SHRLEVEL REFERENCE
- QUIESCE WRITE (YES)
  - Issues a Drain, Flushes the Bufferpool as normal
  - But no SYSCOPY row is created
- When you Update a NOT LOGGED object
  - Status is changed to ICOPY (non restrictive)

-CANCEL THREAD and Aborted URIDs with NOT LOGGED

- Cancelling a thread on a NOT LOGGED object
  – Leaves the object in LPL
  – And in RECP, RBDP or AUXW depending on the type
  – Consider using –CANCEL THREAD…… NOBACKOUT
    - But remember the Logged objects as well!
- LPL can only be removed manually
  – No Automatic LPL recovery
  – Recover, Refresh (MQT), Load or Drop/Recreate
  – Can also issue DELETE without WHERE or TRUNCATE
- Same consideration applies to ROLLBACK
  – Except for LOB objects – see later
- Need to ensure no Duplicate Key or RI violations

Linking Objects

- NOT LOGGED usually managed via Base Tablespace
  - This is called being LINKED
- XML and Indexes are always Linked
- LOB data can become Unlinked
  - Under certain circumstances
  - See next slide for a graphical example
- Look for LOG column in SYSIBM.SYSTABLESPACE
  - Base Tablespace is set to ‘Y’ or ‘N’
  - Linked Objects are set to ‘Y’ or ‘X’
  - LOB objects can be ‘N’ if they are Unlinked

Let's see an example – LOG column values

TS	Y	N	Y	Y	N
IX	Y	X	Y	Y	X
LOB	Y	X	Y	N	N
XML	Y	X	Y	Y	X
		TS Not Logged	**TS Logged**	**LOB Not Logged**	**TS Not Logged**

technologies

Potential Impact on Data Sharing

- NOT LOGGED data is not protected by the Log
  – So it needs to be externalized from Buffers quickly
- Accomplished by treating them differently
  – Read Only Check times (PCLOSEN & PCLOSET) effectively set to 1
  – DB2 treats the object like it was Read Only
    - Buffers are Forced much faster
- Good for data integrity but may have DS impact
  – Objects switch GBP Dependency frequently if updated
- Another good reason to limit using this feature to objects that are genuinely read only

Large Objects (LOBs)

- LOBs being used a lot more today, including in Catalog
- LOB data has always supported NOT LOGGED
  - Syntax changed from LOG YES/NO in DB2 9
  - Old syntax is still recognized and supported
- Old 1Gb restriction for LOB data logging lifted in DB2 9 for z/OS
- Some data is always logged
  - System Pages (Space Map) even if LOG NO or NOT LOGGED used
- For LOG YES LOBs, not everything is logged
  - Only the REDO image of data is logged
    - INSERT pages, AFTER pages of UPDATEs
  - UNDO images not logged
    - DELETE pages, BEFORE pages of UPDATEs

How does LOB logging differ?

- Size of LOB data is the obvious consideration
- LOB Data in an Auxiliary TS is never updated in place
  - Old data marked as deleted and new data added
  - So the old data is still available if required
  - Known as Shadow Copy Recovery
- No Logging of UNDO records for LOB data
  - ROLLBACK/BACKOUT recovery for LOB data not possible
    - SQL ROLLBACK works because the space maps are logged
- Also no Logging of LOB data for DELETEs
  - Space maps provide information to allow roll forward recovery
- Remember NOT LOGGED data is Forced at Commit
  - May impact Application response times

LRSN Spin Avoidance

- Before DB2 9 each Log record required a unique LRSN
  - Could result in DB2 waiting until new LRSN available
  - Each LRSN represents around 16 microseconds
  - High CPU overhead and Log Latch Contention
- DB2 9 NFM allowed duplicate LRSNs
  - Must be successive Log records from the same member
  - Must be for different pages (commonly data and index pages)
  - Consecutive Log records for same page require unique LRSN
- DB2 10 NFM extended this capability
  - Consecutive Inserts for the same data page can share LRSN
  - Does not apply to Updates, Deletes or same index page
- Very useful for multi-row Inserts with no/few indexes
- DB2 11 Extended LRSN virtually eliminates this as a problem

DB2 9 – Major Logging related changes

- Archive Log process runs in 31 bit mode (in MSTR)
  - Reduces risk of Storage related failures when running Archive Logs
  - Allows much larger Log Buffers to be used
  - Also attempts to fill next Buffer while processing previous one
- Extended BSDS became compulsory
  - Supports 10,000 Archive and 93 Active Log pairs
- Increases in storage to improve performance
  - Active Log input Buffers increased from 15 to 120
    - Improves Fast Log Apply performance by up to 100%
  - Archive Log input buffer increased – 10 tracks per stripe
- All LOB Tablespaces can be Logged
  - Limited to <1Gb LOBs in previous releases

DB2 9 – Archive Log improvements

- Archive Log access changed from BDAM to BSAM
  - Allows features of DFSMS Extended Format datasets to be used
  - Archive Logs can now be striped
  - Compression ca also be used
    - May be a viable alternative to striping
  - See DB2 10 improvements which may also affect this decision
- Archive Logs can use Extended Volumes (EAV)
  - Specify DSNTYPE=LARGE
  - Prevents multi-volume or tape based archives
  - These were required if you were using 4GB Active Logs

DB2 10 – Latch Class 19

- Critical for data consistency
  - Used to Serialize updates
- Single latch per subsystem used before DB2 10:
  1. Latch obtained
  2. RBA range reserved
  3. Log record moved into the Log Buffer
  4. Latch released
- Now Latch is released once Buffer space reserved
  - Multiple log records can be written in parallel
  - Improves potential logging rates
  - Available in CM Mode

DB2 10 – Log Buffers

- Log Buffers are now fixed in Storage
  - DSNZPARM OUTBUFF – now defaults to 4MB (was 400KB)
- 4MB should be enough for most sites
  - IFCID 001 can indicate when your value may be too small
    - QJSTWTB – Log writes waits due to no available log buffers
  - Care required as all storage defined is now fixed
- Potential reasons to increase this include:
  - Improved ROLLBACK – hopefully not relevant!
  - CONSISTENT COPY
  - Products that use IFCID306 log reading interface
    - E.g. DB2 Data Replicator for z/OS, CONCURRENT REORGs

DB2 10 – Log I/O Enhancements

- More log writes are asynchronous and in parallel
  - In DB2 9 DB2 performs re-writes of each Log CI serially
    - Typically at COMMIT (Forced Writes)
    - Avoided risk of data loss in the event of dual log I/O failures
    - But led to more waits for Log Writes at Commit
  - DB2 10 allows these to be performed in Parallel
    - Change made possible by improvements in Disk technology
    - No loss of data integrity
- Hardware and Channel Technology improvements
  - DS8800 Disks
  - High Performance FICON (HPF) I/O protocols
    - Can be used for Log I/O > 64KB on a z196

technologies

DB2 10 I/O Performance

- Taken together these changes make a big difference
  - Redbook reports 50% reduction in Log suspend time for 3 page Synchronous Writes (e.g. at Commit)
    - Absolute difference maintained for more pages
    - The relative Percentage savings are lower
- Maximum Log throughput now reported at up to 180MB/sec
  - Compared to around 100MB/sec on a DS8300 using DB2 9
- May prevent the need to use Log Striping?

DB2 10 - Dynamically adding an Active Log

- Before DB2 10 had to use DSNJU003 – needs DB2 shutdown
- New –SET LOG NEWLOG syntax added
  - Available in CM mode - Remember to add both Active log pairs
  - This is a permanent change
  - RUN DSNJLOGF (to preformat) before adding
- Can get you out of jail in an emergency
  - Archive Log fails or hangs and you're running out of active logs
  - If Log fills and you need to shut down DB2
- Look for message DSNJ110E if archive log waits
  - -ARCHIVE LOG CANCEL OFFLOAD may be required
    - Cancels hanging log offload and starts the process again

DB2 10 - Improved Checkpoint Flexibility

- DB2 9 checkpoints using Time or Number of log records
- Neither is ideal is all circumstances
- Time is usually best choice for most customers
  - But at busiest times there may be too many log records
  - This elongates subsystems restart time
- DB2 10 can checkpoint on either value
  - Set DSNZPARM CHKTYPE=BOTH
  - And provide values for both CHKFREQ and CHKLOGR
- Setting can be changed using –SET LOG command

Other DB2 10 changes

- Must use extended format BSDS
  - Will have been done already if you're migrating from DB2 9
  - Install CLIST does this if it hasn't been done already
    - But doesn't resize the BSDS
  - If migrating from V8 then you need to check - DSNJU004
    - Look for the DSNJCNVB job message
    - Better option is to track down a V8 Install Guide
- Log Apply Storage default changed to 500MB
  - DSNZPARM LOGAPSTG – used to be 100MB
  - Introduced by PM31641 after GA date

DB2 11 – Extended RBA/LRSN

- Expanded RBA/LRSN optionally available
- Will increase RBA from 6 to 10 bytes
  - Basically a full word available at the front of the existing 6 byte RBA
- Will also increase LRSN from 6 to 10 bytes
  - One byte added before existing LRSN
    - IBM reckon >30,000 years availability
      - My math – ~36,352 years!
  - Three bytes precision added at the end
    - Current LRSN has about 16 microsecond precision (10^{-6})
    - Current z/OS STCKE has precision to 244 picoseconds (10^{-12})
    - Expanded LRSN to 953 femtoseconds (10^{-15})
- Should virtually eliminate LRSN Spin issues for most sites

DB2 11 – Other Log related changes

- Log Reads and Writes are now zIIP eligible
- Log Output Buffer moved from MSTR to 64-bit common area
  - Avoids a cross-memory call to MSTR when updating
  - Does not require a REBIND
- IBM Reorg writes new Compression Dictionary to Log
  - Allows products such as IBM CDC to continue without pause
- Support for Not-Logged DGTTs
- Re-activation of Fast Log Apply during Restart
  - Accidentally disabled during life of DB2 9!
- Reductions in Log Writes due to Column Insert processing and Synchronous Log Writes during Index Structure modifications
- Further improvements to Log Latch Class 19

DB2 12 – Logging related changes

- 7-byte RIDs introduced by RPN Partitioning
  - Used from CM mode – needs fallback toleration PTF for DB2 11 support
- Support for Large Active Log datasets
  - Up to 768Gb – should be enough for most sites!
- Support for z/OS Hyperwrite (PPRC Log Write Accelerator)
  - Up to 30% reduction in Log Write latency using PPRC
  - Retrofitted to DB2 10 and 11
- Further significant reductions in Log Latch contention
  - IBM report up to 41% CPU and 6% Elapsed reductions
- Removed Log Write Force when caching Identity Columns and Sequences in Data Sharing

technologies

Steve Thomas
Senior Engineering Services Architect
Steve.thomas@ca.com

@cainc

Slideshare.net/CAInc

Linkedin.com/company/ca-technologies

CA.com