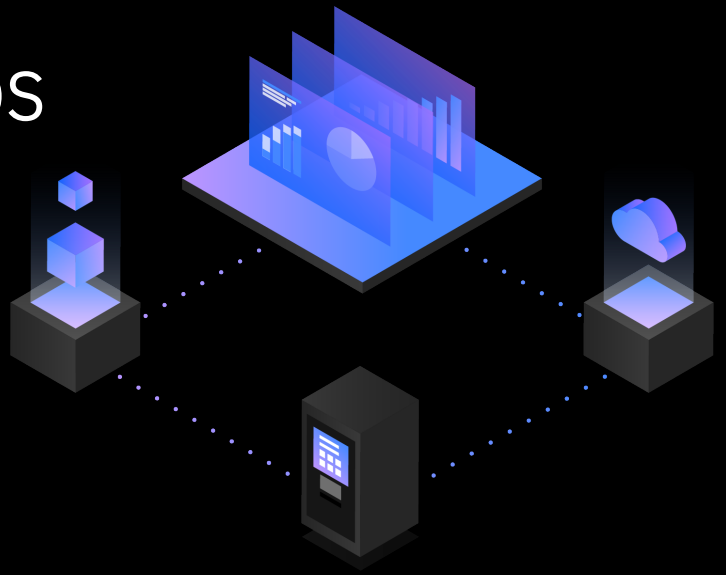
Database Administration Enhancements of Db2 13 for z/OS



New England Db2 Users Group

September 28, 2023

Robert Catterall, IBM
Principal Db2 for z/OS Technical Specialist

IBM

Agenda

- Online conversion of PBG table space to PBR
- More open data sets
- RPN: default page numbering mechanism for universal PBR table spaces
- Application-level lock timeout limit and deadlock priority
- Db2 profile table enhancements
- Online removal of active log data sets
- Improved processing of inserts for PBG table spaces
- Improved monitoring of index page split activity
- Reducing conflict between ALTER TABLE and dependent packages (and dependent cached dynamic SQL statements)

Online conversion from PBG to PBR: the need

- Partition-by-growth very popular since introduction (Db2 9)
  - Easy to set up: just determine appropriate partition size (DSSIZE) and maximum number of partitions (MAXPARTITIONS), and you're done
- Problem: when PBG table space gets really big (many millions of rows), advantages of partition-by-range become more apparent…
  - Performance: better support for parallelism (both user- and Db2-enabled), page-range screening (access path option), clustering within partitions, etc.
  - Availability: maximum partition independence for utility processing – can even LOAD at partition level without affecting access to other partitions
  - Much greater capacity (when relative page numbering used)
- Unfortunately, PBG to PBR required unload/drop/re-create/re-load

Enter Db2 13: online PBG-to-PBR

- Pending DDL change: ALTER TABLE, followed by online REORG
- The key: new ALTER PARTITIONING clause of ALTER TABLE

```
ALTER TABLE PROD.TB01
ALTER PARTITIONING TO PARTITION BY RANGE (COLINT, COLCHAR)
(PARTITION 1 ENDING AT ( 5, 'CCC'),
 PARTITION 2 ENDING AT (10, 'MMM'),
 PARTITION 3 ENDING AT (MAXVALUE, MAXVALUE))
```

- After execution of statement above, and subsequent materializing online REORG, the now-PBR table space immediately available for access
  - Indexes defined on the table in the PBG table space will still be there (they will be non-partitioned indexes on the table in the PBR table space)

Online PBG-to-PBR: additional information

- ALTER TABLE with ALTER PARTITIONING TO PARTITION BY RANGE must be issued via package with APPLCOMPAT value of V13R1M500 or higher
- DSSIZE of new PBR table space inherited from former PBG table space
  - Keep that in mind when deciding on partition limit keys for the PBR table space: *make sure that rows assigned to each partition will fit within that DSSIZE*
    - OK if new PBR table space has more or fewer partitions than former PBG table space (but not fewer if table defined with DATA CAPTURE CHANGES)
  - What If your preferred partitioning scheme will require a larger DSSIZE?
    - If fix for APAR PH51359 is applied, DSSIZE change and ALTER PARTITIONING change can be put into effect via one online REORG; otherwise, those two changes will require two REORGs of the table space
- PBR table space will use relative page numbering (more on RPN to come)

More open data sets

- Factors driving growth in number of data sets in Db2 for z/OS systems:
  - New applications that require new table spaces and indexes
  - Ongoing conversion of table spaces from non-universal to universal
    - One universal table space can hold one table
    - All universal table spaces are partitioned
  - Existing PBG table spaces grow into new partitions
- What limits the number of open data sets for a Db2 subsystem?
  - It's the space needed for open data sets *below the 2 GB "bar" in the Db2 database services address space (DBM1)*
- Db2 12: limit on number of open data sets (DSMAX in ZPARM) is 200,000
  - Generally speaking, practical limit is less than that

Open data set scalability: Db2 13 + z/OS 2.5

- z/OS 2.5 enhancement – exploited by Db2 13 – moves some control blocks associated with open data sets above the 2 GB bar in virtual storage
  - To take full advantage of this z/OS enhancement, update the SWBSTORAGE parameter in the ALLOCxx member of PARMLIB:
  - `SWBSTORAGE (ATB)` ⟵ *Alternative value is SWA – data set control blocks go below bar*
- Also helpful: Db2 13 reduces ECSA consumption – a smaller ECSA means more below-the-bar space in DBM1 private for data set control blocks
- With Db2 13, max allowable DSMAX value goes from 200,000 to 400,000
  - Still true: practical Db2 13 open data limit likely less than max DSMAX value
  - That said, with Db2 13 + z/OS 2.5, should be able to support about 2X the number of open data sets versus Db2 12 system

For PBR table spaces, RPN becomes default

- RPN: short for relative page numbering – a new page numbering scheme for range-partitioned tables, introduced with Db2 12
  - Greatly increases data capacity: up to 280 trillion rows in one table
  - Removes linkages between page size, DSSIZE and max number of partitions
    - Can have up to 4096 partitions, regardless of page size and DSSIZE
  - DSSIZE flexibility: can be nG, where n is *any integer* between 1 and 1024
    - Without RPN, DSSIZE must be a power of 2 between 1G and 256G
  - Different partitions of the same table space can have different DSSIZE values
    - When ALTER PARTITION issued to increase DSSIZE value, change is immediate
- Db2 13: default for ZPARM parameter PAGESET_PAGENUM (default page numbering scheme for new PBR table space) is RELATIVE (was ABSOLUTE)

Application-level lock timeout limit

- Before Db2 13: lock timeout value specified via ZPARM parameter IRLMRWT applied to all application processes
- Often, a universal lock timeout limit will NOT be ideal for all applications
  - Suppose IRLMRWT is at its default value of 30 seconds
    - Developer of client-facing online application: “WHAT? You want a user to look at a spinning wheel FOR 30 SECONDS if we have to wait on a lock? NO! Time us out in 5 seconds and let the user re-drive the transaction!”
    - Developer of batch job: “WHAT? This job has been running for two hours, and you want to time it out because IT HAD TO WAIT 30 SECONDS for a lock? DON’T time it out unless it has to wait more than 10 minutes for a lock!”
    - Both these developers are right from the perspective of their application, but there’s only one lock timeout value – *or there was only one, before Db2 13…*

Db2 13 and lock timeout granularity

- New special register: CURRENT LOCK TIMEOUT
- Available once function level V13R1M500 has been activated
- Range of possible values: -1 to 32,767
  - -1: process will not time out waiting for a lock – it will either get the lock or it will be terminated if it deadlocks with another process and is the deadlock “loser”
  - 0: process will get an error message if requested lock cannot be obtained immediately (basically, externalizes “conditional” lock request functionality)
  - Between 1 and 32,767: process waits up to that number of seconds for a lock
  - If you want max allowable value for CURRENT LOCK TIMEOUT to be less than 32,767, lower value can be specified via SPREG_LOCK_TIMEOUT_MAX in ZPARM
- DSNT376I lock timeout message has info on application-set timeout limit

Db2 13: influencing deadlock priority

- Problem: prior to Db2 13, if your process became deadlocked with another process, little you could do to influence Db2's selection of deadlock "winner" (proceed!) and "loser" (sorry!)
  - What if your process is one that you consider to be "must-complete?"
- Db2 13: new built-in global variable, DEADLOCK_RESOLUTION_PRIORITY, available once function level 501 has been activated
  - Value of DEADLOCK_RESOLUTION_PRIORITY can be in the range of 0-255
    - Higher the value, the more likely that process will be winner in case of deadlock
    - Note: even if DEADLOCK_RESOLUTION_PRIORITY is 255, process could still be deadlock loser if other process is changing data in table space defined with NOT LOGGED, or if it is a rollback or an abort or a backout process
- Schema name: SYSIBMADM (user-set-able built-in global variables)

You may be wondering…

- Why is CURRENT LOCK TIMEOUT a special register, while DEADLOCK_RESOLUTION_PRIORITY is a global variable?
- Here's why: any process can set the value of a special register, but a process has to have permission to set the value of a global variable
  - If every process could set value of DEADLOCK_RESOLUTION_PRIORITY, everyone might go for max value – that would defeat purpose of the feature
  - Expectation is that DEADLOCK_RESOLUTION_PRIORITY will be used selectively for processes (maybe database administration, maybe application-related) that truly need to have highest probability of being the winner in case of a deadlock
    - Note: finding right value for process that should have "higher" but not "highest" deadlock priority value could require some trial and error – IFCID 172 trace record provides priorities of processes involved in deadlocks

Setting lock timeout, deadlock priority values

- In addition to a process being able to do that directly with a SET statement, it can be done automatically for a process by way of the Db2 profile tables
  - And, this is true for local-to-Db2 processes as well as for DDF-using processes
    - Setting value of special register other than CURRENT LOCK TIMEOUT, and built-in global variable other than DEADLOCK_RESOLUTION_PRIORITY, via profile tables remains do-able only for DDF-using processes
- To do this via profile tables:
  1. Identify application process in SYSIBM.DSN_PROFILE_TABLE
     - Local-to-Db2 process: identifiers include auth ID and/or role; collection name and/or package name (DDF-using process: identifiers same as with Db2 12)
  2. In SYSIBM.DSN_PROFILE_ATTRIBUTES, provide appropriate SET statement
     - Db2 13 doc has info on doing this for special registers, built-in global variables

Another Db2 13 profile table enhancement

- New DSN_PROFILE_ATTRIBUTES keyword: RELEASE_PACKAGE
- Available for use with both local-to-Db2 and DDF-using processes
- One allowable value (specified as ATTRIBUTE1): COMMIT
- Purpose: override RELEASE(DEALLOCATE) specification, avoid contention with action (e.g., ALTER TABLE) that requires package invalidation
  - Package cannot be invalidated if it is in use, and if RELEASE(DEALLOCATE) is in effect then package is considered to be continuously in-use as long as thread to which is is allocated exists – ALTER TABLE fails if package cannot be invalidated
- “But wait,” you might say, “Didn’t a V11 enhancement enable Db2 to detect when an action is blocked by a RELEASE(DEALLOCATE) package, and respond by changing the package’s RELEASE behavior to COMMIT?”
  - Yes, but…

RELEASE_PACKAGE profile table enhancement

- The “why”: even with Db2 11’s “RELEASE(DEALLOCATE) contention detection and resolution,” still situations in which RELEASE(DEALLOCATE) package can block a database administration action
  - Thread to which package is allocated might be outside of Db2 (an issue particularly for DDF threads)
  - Package may be allocated to many threads, and can’t be separated from all of them in time to keep DBA action from timing out
- Db2 13 enhancement: DBA can start profile with RELEASE_PACKAGE keyword several minutes before executing (for example) ALTER TABLE, to “clear out” instances of package executing with RELEASE(DEALLOCATE)
  - Then do ALTER TABLE, then “turn off” RELEASE_PACKAGE profile by changing PROFILE_ENABLED to ‘N’ in DSN_PROFILE_TABLE, re-issuing -START PROFILE

Online removal of active log data sets

- How? Via new REMOVELOG option of Db2 command -SET LOG
- Previously, removal of active log data sets required use of DSNJU003 utility (change log inventory), and that utility can only be executed when Db2 is down
- New online active log removal capability nicely complements online active log addition enhancement delivered with Db2 10 (NEWLOG option of -SET LOG) – using these capabilities together sets up some nice use cases

Next slide

Leveraging online active log addition + removal

- Increase the size of your active log data sets…
  - Db2 12 greatly increased max size for active log data set (from 4 GB to 768 GB) – what if you want to replace smaller active log data sets with larger data sets?
    1. -SET LOG with NEWLOG: add new larger data sets to active log inventory
    2. -SET LOG with REMOVELOG: remove old smaller active log data sets
- Encrypt active log data sets using z/OS data set encryption functionality
  - Encryption key label associated with data set *when created,* but active log data sets were created long ago and keep getting reused – how can you encrypt them?
    1. -SET LOG with NEWLOG: add new active log data sets with encryption key
    2. -SET LOG with REMOVELOG: remove old unencrypted active log data sets
- End result: log data sets larger and/or encrypted, Db2 system always up

INSERT enhancement for PBG table spaces

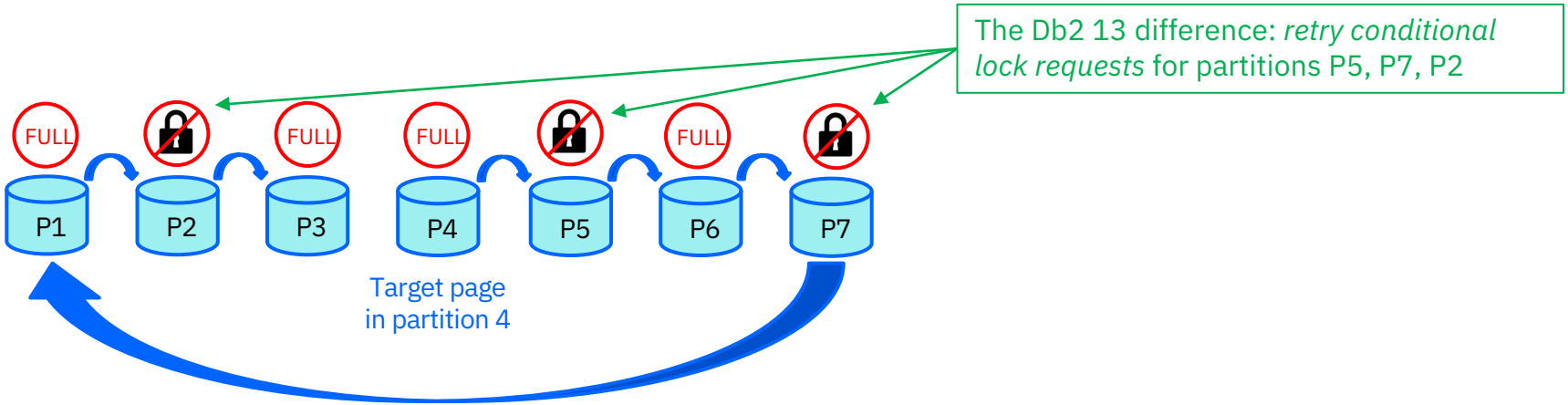
- In a Db2 12 system, insert into PBG table space might fail with -904 error code (resource unavailable) and reason code 00C90090 (partition lock failure) or 00C9009C (partition full)
  - And, reason code 00C9009C might be returned even when one or more partitions of the table space in fact have space to hold new rows
- In Db2 13 system, these PBG insert failures are less likely to occur
- Background: target page for insert determined via table's clustering index, and Db2 tries to get *conditional lock* on associated partition (what "conditional lock" means: if it can't be obtained *immediately*, move on)
  - If conditional lock request for partition fails (or if partition is seen to be full), Db2 tries again with the next partition, and if necessary tries again with partition after that, and so on…

PBG inserts: the Db2 13 difference

- After trying all partitions and either failing to get conditional lock on partition or finding that partition is full, Db2 12 fails the insert as described on previous slide
- Db2 13 will try again for conditional lock on up to 5 partitions for which that lock could not initially be obtained – good chance a retry will succeed
  - If conditional lock retries are unsuccessful, Db2 13 will try once for *unconditional* partition lock and will wait for IRLM timeout period to acquire that lock – if unsuccessful, *then* issue -904 with code 00C90090
- Also: Db2 13 does a better job of tracking partition-full situations, and if all partitions are seen to be full, a new partition will be added to hold new row (assuming number of partitions has not reached MAXPARTITIONS value)
  - Should avoid “false full” insert failures that could be seen with Db2 12

Db2 13 PBG insert enhancement: the big picture

- Scenario:
  - PBG table space has 7 partitions, target page (per clustering index) in partition 4
  - Partition 4 is seen to be full, and Db2 tries other partitions (Db2 might go forward or backward through partitions), in each case either failing to get conditional lock on partition (P5, P7, P2) or finding that partition is full (P6, P1, P3)



Enhanced monitoring of index page splits

- Index page splits – necessary when Db2 has to insert an entry into an index page that is full – can can have a significant negative impact on performance of high-volume INSERT processes
  - Especially true for GBP-dependent indexes in Db2 data sharing environment
- In Db2 12 system, information that could help in monitoring and mitigating index page split activity often *not available* and *incomplete*
  - *Often not available* – index page split activity captured in IFCID 359 trace records
    - IFCID 359 activated via performance trace class 4, which is not on by default (relatively high-overhead trace record – generated for every index page split)
  - *Incomplete:* even if IFCID 359 active, trace records lack important diagnostic information such as unit of recovery (UR) ID and data sharing member number

Db2 13 index page split information – tracing

- When Db2 13 function level 500 is activated, new IFCID 396 trace records generated when statistics trace class 3 is active (stats class 3 is active by default)
  - IFCID 396 is low-overhead: only generated when elapsed time of index page split unusually high (> 1 sec)
  - More-complete information, including *UR ID* and *data sharing member number* associated with index page split action

Db2 13 index page split information – real-time stats

- When catalog level goes to V13R1M501 (do-able when function level 500 activated), 3 new columns related to index page split activity added to SYSINDEXSPACESTATS table (and SYSIBM.SYSIXSPACESTATS_H):
  - REORGTOTALSPLITS – total number of index page splits since last REORG or rebuild of index
  - REORGSPLITTIME – total/aggregated elapsed time of index page splits since last IX REORG or rebuild
  - REORGEXCSPLITS – total number of index page split actions with unusually high elapsed time (> 1 second) since last reorganization or rebuild of index

Conflict between ALTER TABLE and dependent SQL

- ALTER TABLE actions tend to conflict with execution of dependent packages (same thing is true regarding dependent cached dynamic SQL statements)
- That being the case, if dependent packages are continuously in use then ALTER TABLE action can fail
  - Stopping application activity to get dependent packages out of the way of an ALTER TABLE action is not an attractive solution

ALTER TABLE conflict reduction – initial step

- Starting with Db2 13 function level 500, ALTER TABLE with DATA CAPTURE CHANGES (related to Db2 log-based data replication) does not require quiescing of dependent packages (or of dependent cached dynamic SQL)
- Result: ALTER TABLE with DATA CAPTURE CHANGES can succeed with no impact to related application workload

"That's nice, but what about conflict between dependent SQL statements and other ALTER TABLE actions?"

Watch this space – our work here is not done

Robert Catterall

rfcatter@us.ibm.com