IDUG

2022 NA **Db2** Tech Conference

Let Me Make This Clear (Vol. 3): Explaining Often-Misunderstood Db2 for z/OS Concepts and Facilities

Robert Catterall, IBM

Session Code: D13

Boston, MA

Agenda

- APPLCOMPAT
- The IBM Data Server Driver / Db2 Connect package collection
- Db2 12 function level 504 and “deprecated objects”
- Contiguous buffer pools
- MAXDBAT and DDF transaction queuing
- Active and idle DBATs
- Trimming the DBAT pool

APPLCOMPAT

- What it is: a Db2 package bind option, and a Db2 ZPARM parameter
- What plenty of people think:
  - *APPLCOMPAT affects access path selection*
  - *The value of APPLCOMPAT in ZPARM determines the functionality available in a Db2 system*
  - *Changing APPLCOMPAT is necessary after migrating to a new version of Db2, and after activating a new Db2 function level*
  - *All packages have an APPLCOMPAT value*

No, no, no and not necessarily

APPLCOMPAT – clearing things up

- The APPLCOMPAT package bind option has two purposes
- Purpose #1: it enables an application program to use SQL functionality delivered with a new version or function level of Db2
  - Example: new built-in aggregate function LISTAGG was added with Db2 12 function level 501, so if program needs to use LISTAGG then APPLCOMPAT value for program's package needs to be V12R1M501 or higher

APPLCOMPAT – clearing things up (cont'd)

- APPLCOMPAT purpose #2: it protects a program from the effect of a SQL incompatibility, if the program would be negatively impacted by that incompatibility
  - "SQL incompatibility" means same SQL, same data, *different result*
  - Example: starting with Db2 11 in new-function mode, a store clock value is no longer a valid input for TIMESTAMP function (results in -180 SQL error code)
  - If program needs to cast a store clock value AS TIMESTAMP, *it can do that* if its package has an APPLCOMPAT value of V10R1
    - That APPLCOMPAT value means program gets *SQL behavior* of Db2 10
- In Db2 12 or 13 system, a package's APPLCOMPAT value can be V10R1, V11R1 or any Db2 12 or 13 function level (e.g., V12R1M502)

APPLCOMPAT and access path selection

- APPLCOMPAT does NOT affect query access path selection
  - Example: the Db2 12 optimizer can use adaptive index selection to improve the performance of queries that involve “index ANDing”
    - To take advantage of that and other Db2 12 optimizer enhancements, must a program’s package have an APPLCOMPAT value of V12R1M100 or higher?
    - NO – only need to bind or rebind package in Db2 12 system (ideally without the APREUSE option), *regardless of package’s APPLCOMPAT value*

APPLCOMPAT and system functionality

- APPLCOMPAT parameter in ZPARM does NOT determine the level of functionality available in a Db2 system – it just provides default APPLCOMPAT value when needed:
  - For BIND PACKAGE, default value is needed when command is issued without an APPLCOMPAT specification
  - For REBIND PACKAGE, default value is needed if APPLCOMPAT is not specified *and the package does not already have an APPLCOMPAT value* (a package's existing APPLCOMPAT value is retained, by default, when package is rebound)

About changing APPLCOMPAT values for your packages…

- You do not have to change the APPLCOMPAT value for packages after migrating to a new version of Db2, or after activating a new Db2 function level
  - It is a good idea to rebind all plans and packages soon after migrating to new Db2 version, even while still at function level 100 – should deliver performance boost (even when access paths for package's statements don't change)
  - You can change APPLCOMPAT values when you do that large-scale package rebind, but you don't have to

More on changing package APPLCOMPAT values

- After the large-scale package rebind done while at function level 100, no technical need to do that again as higher Db2 function levels activated
- Remember: starting with Db2 12, APPLCOMPAT affects DDL as well as DML SQL
  - Packages used for DDL will need APPLCOMPAT change to support new syntax
  - Example: ALTER TABLE with KEY LABEL requires APPLCOMPAT(V12R1M502)
  - DDL-related packages might be for SPUFI, DSNTEP2, …
- Information to come on APPLCOMPAT value for IBM Data Server Driver / Db2 Connect packages

(or higher)

Can a package not have an APPLCOMPAT value?

- It's possible (more likely with Db2 12 versus Db2 13 – see next slide)
- Check SYSPACKAGE catalog table
- You might see packages for which APPLCOMPAT column is blank
  - What that likely means: old package, last bound or rebound a long time ago (APPLCOMPAT was introduced with Db2 11)
  - In Db2 12 system: for each such package, might want to see if it is still used (look for a relatively recent date in LASTUSED column of SYSPACKAGE)
  - If package is still used, consider rebinding with APPLCOMPAT value (V10R1 would be a good choice – likely reflects de facto SQL behavior)
    - This will cause package to have an APPLCOMPAT value – if it's rebound again with no APPLCOMPAT value specified, the current APPLCOMPAT value will be retained (otherwise, new APPLCOMPAT value will come from ZPARM parameter)

Can package in Db2 13 system not have an APPLCOMPAT value?

- Not impossible, but unlikely – here's why:
  - Db2 12 function level 510 must be activated prior to migrating to Db2 13
  - Db2 12 function level 510 will not be activated if Db2 determines that there are any packages in the system that a) have been used within the past 18 months and b) were last bound or rebound prior to Db2 11
  - A package bound or rebound in a Db2 11 or Db2 12 system will have an APPLCOMPAT value
  - So, if you have a package in a Db2 13 system with no APPLCOMPAT value, it must be one that was last bound or rebound prior to Db2 11 and last used more than 18 months prior to the migration to Db2 13
    - A request to execute that package will result in auto-rebind (pre-Db2 11 package cannot execute in Db2 13 system), and auto-rebound package will have an APPLCOMPAT value

The IBM Data Server Driver / Db2 Connect packages

- *These packages reside only in the NULLID collection, right?*
- *Not necessarily…*
  - More and more organizations have multiple collections for the IBM Data Server Driver / Db2 Connect packages, differentiated by bind specifications
    - In one collection, packages might be bound with RELEASE(DEALLOCATE), to get high-performance DBAT functionality
    - In another collection, packages might be bound with CONCENTRATESTMT(YES), to get Db2 statement concentration functionality
    - In another collection, packages could have APPLCOMPAT value different from that of NULLID packages, for applications needing the different value
  - How this is done: BIND COPY the packages from NULLID to alternate collection, with desired bind options specified

Using multiple IBM Data Server Driver collections

- Having several collections for IBM Data Server Driver / Db2 Connect packages lets Db2 team *get different behaviors for different DDF-using applications* by pointing applications to appropriate collection
- *But how do you point a given DDF-using application to an IBM Data Server Driver / Db2 Connect collection other than the default NULLID?*
  - At one time, only way to do that at connection time was to specify the alternate collection on the client side as a data source property
  - Since Db2 11 for z/OS, you can accomplish the objective with a server-side action, via the Db2 profile tables more info, next slide

The Db2 profile tables and package collections

(identify a part of the DDF workload)

- Profile tables: SYSIBM.DSN_PROFILE_TABLE and SYSIBM.DSN_PROFILE_ATTRIBUTES

(tell Db2 what to do for that part of the DDF workload)

- Example: Use profile tables to point DRDA requester to alternate collection at connection time, to enable high-perf DBAT functionality:
  - In DSN_PROFILE_TABLE, identify requester application as (for example) the one that connects to Db2 using authid ABC123 – suppose this is profile 3
  - In DSN_PROFILE_ATTRIBUTES, indicate that for profile 3, automatically issue SET CURRENT PACKAGE PATH = X, where X is name of collection in which IBM Data Server Driver packages bound with RELEASE(DEALLOCATE)
  - More information: http://robertsdb2blog.blogspot.com/2018/07/db2-for-zos-using-profile-tables-to.html

More on profile tables and collections

- Profiles will not be in effect if they are not started
  - Two ways to do that:
    - Manually: issue the Db2 command -START PROFILE
    - Automatically: set value of the Db2 ZPARM parameter PROFILE_AUTOSTART to YES (this ZPARM parameter was introduced with Db2 12 – default value is NO)
    - Either way, Db2 will load into memory and activate all profiles for which the value of PROFILE_ENABLED in DSN_PROFILE_TABLE is 'Y'

Note: to get high-performance DBAT functionality, need at least some RELEASE(DEALLOCATE) packages to be used with DDF threads, and need to set value of DDF parameter PKGREL to BNDOPT (can be done via -MODIFY DDF command)

Also note: when a profile is used to (for example) automatically set the value of a special register for a DDF-using application, that action will be taken when the application next connects to the Db2 subsystem after the profile has been activated

Db2 12 function level 504 and “deprecated objects”

- Plenty of misunderstanding here – some people think:
  - *When Db2 12 function level 504 is activated, you can no longer create traditional segmented (i.e., not universal) table spaces*
- Statement above is false, statement below is true:
  - When a package through which DDL statements are executed (e.g., DSNTEP2 package) is executing at application compatibility level V12R1M504 or higher, that package cannot be used to create a traditional segmented table space
    - That package also cannot be used to create a Db2 synonym, or create a hash-organized table or alter an existing table to be hash-organized

More on function level 504 and “deprecated objects”

- If Db2 12 system is running with function level 504 or higher activated, and you need to create *a traditional segmented table space*, you have two options:
  - Create table space using a program (DSNTEP2, SPUFI, whatever) whose package is bound with APPLCOMPAT(V12R1M503) or lower,

  -or-

  - If using a dynamic SQL-issuing program (e.g., DSNTEP2) whose package is bound with APPLCOMPAT(V12R1M504) or higher:
    - First, issue SET CURRENT APPLICATION COMPATIBILITY = ‘V12R1M503’
    - Create the traditional segmented table space

Dynamic SQL-issuing program can change application compatibility level to value lower than – but not higher than – the APPLCOMPAT value of the package it is using

Contiguous buffer pools

- *Plenty of folks don't know what these are,* and plenty of folks who do know what they are *have a misunderstanding of how they are managed by Db2*
- What a contiguous buffer pool is:
  - It is what we call a PGSTEAL(NONE) buffer pool, starting with Db2 12
  - PGSTEAL(NONE) buffer pool specification introduced with Db2 10
    - PGSTEAL(NONE) does not mean that Db2 cannot steal a buffer in the pool – it means that buffer stealing is not expected for the pool, because the pool is supposed to have enough buffers to hold all pages of all assigned objects
    - When object assigned to PGSTEAL(NONE) buffer pool is first accessed, Db2 will quickly retrieve whatever data the requesting process needs, and then will asynchronously load all the other pages of the object into the buffer pool

PGSTEAL(NONE) – what is different starting with Db2 12?

- When Db2 loads pages of an object into a PGSTEAL(NONE) buffer pool, pages will essentially be *arranged in memory as they are arranged on disk*
  - Thus the term, "contiguous buffer pool"
  - This arrangement of pages in the pool improves CPU efficiency because every page access will be a direct access
    - Meaning: Db2 knows exactly where each page is located in the pool, without having to deal with the hash and LRU chains associated with buffer management in standard pool
  - As was true before Db2 12, accommodation must be made for buffer stealing for a PGSTEAL(NONE) pool, in case the pool does not have enough buffers for all pages of all assigned objects
    - How can that be done while maintaining "contiguous-ness" of pages in pool?

Buffer stealing for a contiguous buffer pool

- To allow for possibility of buffer stealing for PGSTEAL(NONE) pool, Db2 will set aside a portion of a PGSTEAL(NONE) buffer pool to be the "overflow area"
  - Any required buffer stealing will be from pool's overflow area
  - Buffers in the overflow area are not contiguously arranged, and stealing – if needed – will be accomplished using the FIFO (first-in, first-out) algorithm
- Size of steal area: 10% of pool's buffers (but not more than 6400 or fewer than 50 buffers)

Maximizing CPU benefit of a contiguous buffer pool

- Because max efficiency comes from direct access to pages in the pool, you'd like for all pages in the pool to fit within contiguous part of the pool – in other words, the pool's overflow area will ideally be empty
  - What that means, specifically: you ideally want all pages of all objects assigned to a PGSTEAL(NONE) buffer pool to fit within 90% of the pool's buffers
    - Or, for a pool with VPSIZE >= 64000, within VPSIZE - 6400 buffers

PGSTEAL(NONE) and FRAMESIZE

- Buffer pool can be backed with large real storage page frames when pool defined with PGFIX(YES) – improves CPU efficiency of page access
- FRAMESIZE(2G) and PGSTEAL(NONE) will not both be honored for a PGSTEAL(NONE) buffer pool* – here's why:
  - For direct page access to work in contiguous part of PGSTEAL(NONE) pool, need to ensure that a given frame holds pages belonging to 1 and only 1 object
  - If 2G page frames used, this "one object's pages in one frame" rule could mean that a whole lot of space in some 2G page frames would be wasted
- If you want large frames for PGSTEAL(NONE) pool, use FRAMESIZE(1M)
  - Having some wasted space in some 1M frames should not be a big deal

** In Db2 12 system, if fix for APAR PH22469 applied and FRAMESIZE(2G) specified for PGSTEAL(NONE) pool, FRAMESIZE(2G) will be honored but PGSTEAL(NONE) will not be honored (PGSTEAL(LRU) used, instead); otherwise, PGSTEAL(NONE) will be honored and FRAMESIZE(2G) will not be honored (4K page frames used, instead)*

The MAXDBAT parameter in ZPARM

- MAXDBAT: number of connections to Db2 subsystem from DDF-using applications that can be concurrently active
  - Connection needs to be paired with a DBAT to be active
- *Plenty of people feel that value of MAXDBAT should be high enough so that it will never be reached*
  - The rationale: if MAXDBAT limit has been reached and DDF transaction comes in from an application, transaction will be queued until a DBAT is freed up to service it – you want to avoid DDF transaction queuing, right?
- *As it turns out, in some situations you might need to induce some level of DDF transaction queuing…*

When DDF transaction queuing can be beneficial

- DDF application might occasionally generate big surge of transactions
- If MAXDBAT value is high enough to allow all those transactions to come immediately into Db2 system, could overwhelm processing resources, severely impacting performance for all applications
- Instead, set MAXDBAT high enough to not be reached most of the time, low enough to protect system from overwhelming surge of transactions
  - Yes, if surge causes MAXDBAT limit to be reached, some DDF transactions will queue up waiting for DBATs, but by protecting Db2 system, throughput will be good and queued transactions won't have to wait long for a DBAT
  - Even when application gets some transaction queueing, performance could be better versus situation in which Db2 system overwhelmed with transactions

Granular management of DDF thread limits

- MAXDBAT applies to the whole of a Db2 subsystem's DDF workload
- Via Db2 profile tables, you can induce transaction queueing for a certain DDF application if it generates a transaction surge, while providing enough DBATs to prevent queuing for other applications
  - Insert row into DSN_PROFILE_TABLE to create profile (identify DDF application by auth ID it uses to connect to Db2 system, or by IP addresses of servers on which application runs, or by another identifier)
  - Insert row into DSN_PROFILE_ATTRIBUTES to set limit on number of threads application associated with profile can use
  - More information in IBM Db2 12 for z/OS Knowledge Center on the Web: https://www.ibm.com/support/knowledgecenter/SSEPEK_12.0.0/admin/src/tpc/db2z_createprofiles.html

Active and idle DBATs

- *Some people think that a DBAT goes from being active to being idle, and then is subject to the Db2 subsystem's idle thread timeout limit*
- Several clarifications needed here:
  - *All DBATs are active* – they are either in-use (i.e., paired with a connection for the purpose of servicing a transaction) or disconnected (i.e., in the DBAT pool)
  - Db2's idle thread timeout value (IDTHTOIN in ZPARM) is applicable only to in-use DBATs – not to DBATs in the pool
    - DBATs in pool are subject to a different time limit – POOLINAC (more to come on that)
  - It is normal for an in-use DBAT to have some idle time – transaction gets going, SQL statement is executed, then a bit of "idle" time before next SQL statement
    - Only a problem when idle time gets too long

A little more on idle thread timeout

- Default value for IDTHTOIN is 120 seconds – if you want to make it higher or lower, have a good reason for doing so
- Db2 profile tables can be used to specify different idle thread timeout values for different DDF-using applications
- By default, when DBAT is idle for a time that exceeds the Db2 limit, Db2 will terminate the transaction *and the associated connection* between the application and Db2
  - If you would prefer for Db2 to terminate the transaction but preserve the associated connection, you can do that for a profile table-specified idle thread limit, via specification of EXCEPTION_ROLLBACK as profile attribute

(Db2 12 enhancement)

Trimming the DBAT pool

- Suppose you have a surge of DDF transactions that takes DBAT count way up – once surge has passed, it is time to bring the number of DBATs down to better align with back-to-normal transaction volume
- *A lot of folks are not clear as to how this happens, and anyway the process recently changed*
- The scoop: at an internally-set frequency (DBAT purge cycle), Db2 checks DBATs in pool to see if any have exceeded the POOLINAC limit
  - POOLINAC: a ZPARM parameter that specifies maximum amount of time a DBAT can be in the pool without being reused – default is 120 seconds
  - A DBAT that has been in the pool, without being reused, beyond the POOLINAC limit is subject to termination by Db2

Trimming the DBAT pool – recent change

- The fix for Db2 12 APAR PH36114 (June 2021) changed how Db2 trims the number of DBATs in the pool, in two ways:
  1. The DBAT purge cycle was changed from 2 minutes to 15 seconds
  2. In a given purge cycle, Db2 will terminate a maximum of 50 DBATs in the "too-long-in-the-pool-without-reuse" category
- Reason for change: previously, if many DBATs went into the pool at around the same time following a DDF transaction surge, Db2 might terminate a lot of DBATs at one time in a subsequent purge cycle
  - Terminating many DBATs at one time could put pressure on z/OS LPAR's ESQA resource, which could cause spill-over into ECSA, which could be bad for LPAR with small cushion of unused ECSA space
  - Now, DBAT pool will be worked down more frequently *and more gradually*

Thank You

Speaker: Robert Catterall

Company: IBM

Email Address: rfcatter@us.ibm.com

Session Code: D13

Please fill out your session evaluation before leaving!