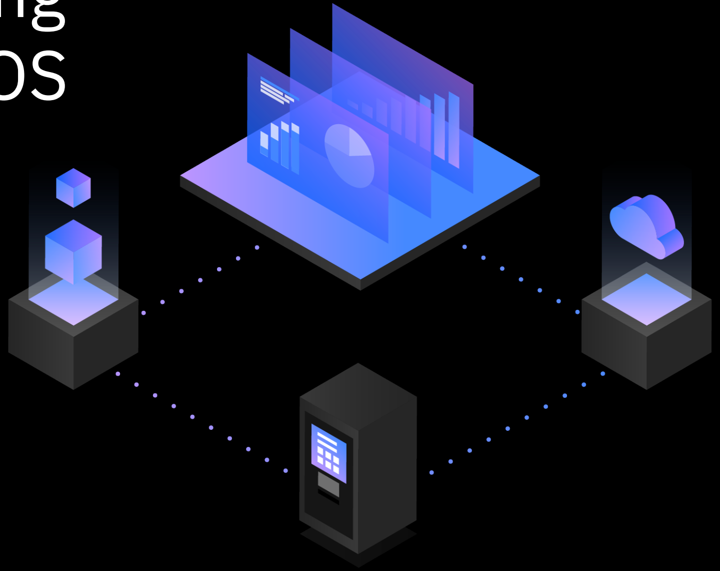
Let Me Make This Clear: Explaining Often-Misunderstood Db2 for z/OS Concepts and Facilities



Tridex Db2 User Group

May 20, 2021

Robert Catterall, IBM
Senior Consulting Db2 for z/OS Specialist

Agenda

- APPLCOMPAT
- The IBM Data Server Driver / Db2 Connect package collection
- MAXDBAT and DDF transaction queuing
- Db2 12 function level 504 and “deprecated objects”
- Db2 12 contiguous buffer pools
- Extended RBA and LRSN values

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 12 system*
  - *Changing APPLCOMPAT is necessary after migrating to Db2 12, and after activating a new Db2 12 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 the 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 system, APPLCOMPAT can be V10R1, V11R1 or any Db2 12 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 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, when the command is issued without an APPLCOMPAT specification
  - For REBIND PACKAGE, if APPLCOMPAT is not specified *and the package does not already have an APPLCOMPAT value* (if package has an APPLCOMPAT value, that value will be retained, by default, when the package is rebound)

About changing APPLCOMPAT values for your packages…

- You do not have to change the APPLCOMPAT value for packages after migrating to Db2 12, or after activating a new Db2 12 function level
  - It is a good idea to rebind all plans and packages soon after migrating to Db2 12, 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 package APPLCOMPAT values when you do that large-scale 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 12 function levels activated
- Remember: 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, ...

(or higher)

- Information to come on APPLCOMPAT value for IBM Data Server Driver / Db2 Connect packages

Can a package not have an APPLCOMPAT value?

- That's a possibility – check SYSPACKAGE catalog table
- You might see packages for which value in APPLCOMPAT column is blank
  - What that likely means: old package, last bound or rebound a long time ago (APPLCOMPAT was introduced with Db2 11)
  - 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 could be a good choice – likely reflects de facto SQL behavior)
    - Reason: if package is subsequently rebound without an APPLCOMPAT specification, you know that current APPLCOMPAT value will be retained (otherwise, new APPLCOMPAT value would come from ZPARM parameter)

The IBM Data Server Driver / Db2 Connect packages

- *These packages reside <u>only</u> 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
  - Accomplished via BIND COPY of packages from NULLID to alternate collection, with desired bind options specified

Using multiple IBM Data Server Driver collections

- Having several collections for the IBM Data Server Driver / Db2 Connect packages enables a Db2 team to get different behaviors for different DDF-using applications by pointing application 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 10 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)

- Use profile tables to point DRDA requester to alternate default collection at connection time, to enable (for example) high-perf DBAT functionality:
  - In DSN_PROFILE_TABLE, identify requester application as (for example) the one that connects to Db2 using authid ABC123, and call this (for example) 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

- DSN_PROFILE_TABLE 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 is new with Db2 12 – default value is NO)
    - Either way, Db2 will load into memory and activates 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)

The MAXDBAT parameter in ZPARM

- *Plenty of people feel that the value of MAXDBAT in ZPARM (number of connections to Db2 subsystem from DDF-using applications that can be concurrently active) should be high enough so that it will <u>never</u> 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 <u>need</u> to induce some level of DDF transaction queuing…*

When DDF transaction queuing can be beneficial

- A DDF application might occasionally generate a big surge of transactions
- If MAXDBAT value is high enough to allow all those transactions to come immediately into the Db2 system, could overwhelm system's processing resources, severely impacting performance of 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 for application(s) seeing transaction queueing, performance could well 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 DDF 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 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

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*
  - *When Db2 12 function level 504 is activated, you can't even <u>use</u> the traditional segmented table spaces that you have*
- Statements above are 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 <u>create</u> a traditional segmented table space
    - That package also cannot 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”

- Can you still use existing traditional segmented table spaces in a Db2 12 environment in which the activated function level is 504 or higher?
  - Absolutely (you can even continue to use existing *simple* table spaces in a Db2 12 function level 504 environment, if you still have any)
- If your Db2 12 system is running with function level 504 or higher activated, and you need to create *a traditional segmented table space*:
  - 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’ and then create 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

Db2 12 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 Db2 12 contiguous buffer pool is:
  - It is what we call a PGSTEAL(NONE) buffer pool in a Db2 12 environment
  - 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 with Db2 12?

- When Db2 12 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 12 will set aside a portion of a PGSTEAL(NONE) buffer pool to be the “steal area”
  - Any required buffer stealing will be from pool’s “steal area”
  - Buffers in the “steal 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 the pages in the pool to fit within the contiguous part of the pool – in other words, the pool's buffer steal area will ideally be empty
  - What that means, specifically: you ideally want all pages of all objects assigned to a Db2 12 PGSTEAL(NONE) buffer pool to fit within 90% of the pool's buffers
    - Or for a pool with VPSIZE >= 64000, within VPSIZE - 6400 buffers

Db2 12, PGSTEAL(NONE) and FRAMESIZE

- Backing buffer pool with large real storage page frames – do-able when pool defined with PGFIX(YES) – improves CPU efficiency of page access
- In Db2 12 system, 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 page frame holds pages belonging to 1 and only 1 object
  - If 2G page frames used, this "one object's pages in one page frame" rule could mean that a whole lot of space in some 2G page frames could be wasted
- If you want large frames for a PGSTEAL(NONE) pool, use FRAMESIZE(1M)
  - Having some wasted space in some 1M frames should not be a big deal

** 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) will be used, instead); otherwise, PGSTEAL(NONE) will be honored and FRAMESIZE(2G) will not be honored (4K page frames will be used, instead)*

Extended RBA and LRSN values

- Information in a Db2 system's transaction log is located by way of RBA (relative byte address) values
  - In a Db2 data sharing system, LRSN values as well as RBA values are written to a member subsystem's log (LRSN is short for log record sequence number)
- With Db2 11 for z/OS in new-function mode, support was added for 10-byte RBA and LRSN values, versus the old 6-byte values
  - That was an important enhancement, tremendously increasing the capacity of a Db2 system's transaction log
- *Plenty of people know that there is a Db2 12 connection to extended RBA and LRSN values, but they are not sure as to what must be done, and when*

Extended RBA and LRSN values, and Db2 12

- The one RBA/LRSN-related requirement for Db2 12 is a bootstrap data set that can accommodate extended RBA and (if applicable) LRSN values
  - Done by executing Db2 utility DSNJCNVT in Db2 11 system, prior to migration
- But wait! Aren't there RBA or LRSN values in table spaces and indexes?
  - **Yes** – RBA or LRSN values are in PGLOGRBA field of table space and index pages
- Doesn't Db2 12 require that those objects be converted to accommodate extended RBA or LRSN values?
  - **No** – those objects eventually need to be converted to accommodate extended RBA or LRSN values, but timing is up to you – Db2 12 itself has (probably almost) nothing to do with it

Extended RBA or LRSN values in database objects

- If Db2 12 itself does not require that table spaces/indexes be converted to accommodate 10-byte RBA or LRSN values, when does that need to be done?
  - Urgency depends on how close you are to maxing out the 6-byte values
  - This tends to be more of an issue for RBA than for LRSN values
    - In Db2 data sharing environment, pages of table spaces and indexes do not contain RBA values in PGLOGRBA field – LRSN values are stored there, instead
    - Absent a "delta" (which could be introduced when standalone Db2 subsystem is converted to data sharing), 6-byte LRSN value won't hit its limit until 2042
  - What about 6-byte RBA values (if Db2 operating in standalone mode)? next slide

If table spaces/indexes have 6-byte RBA values…

- Use command -DISPLAY LOG see where your system’s RBA value is now
- If high-order value in 6-byte RBA field is ‘C’ or less (hexadecimal format), you have a good bit of headroom
- If that high-order value is ‘D’ or higher, lean into getting database object conversions done

A little more on extended RBA and LRSN values

- Even with data sharing group, or standalone subsystem with RBA value not close to 6-byte maximum, go ahead and get table spaces and indexes converted to use extended RBA or LRSN values, sooner rather than later
  - Reason: it will eventually <u>need</u> to be done, so get it done
- Done for catalog and directory objects by Db2-supplied job DSNTIJCV
- Typically done for user/application objects via online REORG
  - Can also be done via LOAD REPLACE or REBUILD index
- “Keep score” via RBA_FORMAT column of SYSTABLEPART and SYSINDEXPART tables in the Db2 catalog

One more thing about RBA values...

- After BSDS converted to accommodate extended RBA and LRSN values, pace at which Db2 subsystem's log RBA value advances quickens
- Why? Because after BSDS conversion, Db2 starts writing 10-byte RBA values to log – more bytes written means quicker advance of RBA value
- Degree to which RBA value increase accelerates will vary by system
  - If logging dominated by inserts of long rows, extra 4 bytes in RBA values will likely have modest impact
  - If logging dominated by updates (especially of shorter fixed-length columns), log records are shorter and extra 4 bytes in RBA values makes more of a difference
- Be aware of this – headroom you thought you had in avoiding maxing out 6-byte RBA value may be little less generous than you'd thought

Robert Catterall

rfcatter@us.ibm.com