# REBIND Phase-In
# DB2 for z/OS
## Function Level V12R1M505

**Tammie Dang**

**IBM Corporation**

**March 2021**

# Agenda

- Problem Statement and Solution Statement
- Use Cases (As-is vs To-be scenarios)
- REBIND PACKAGE PLANMGNT review
- REBIND Phase-In feature
- Recent Enhancements
- Questions

# Overview and Problem, Solution Statements

# Db2 Problem Statement

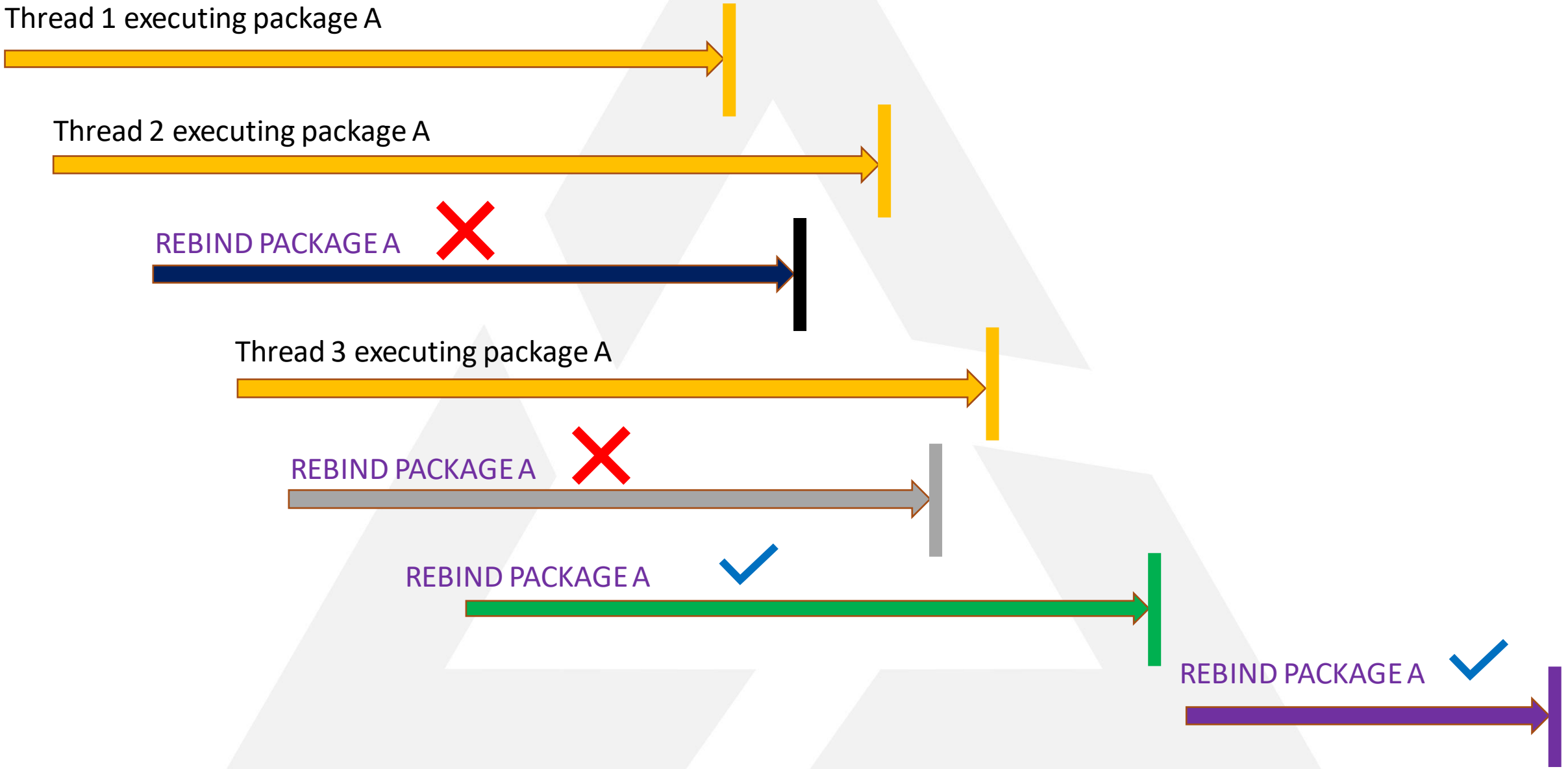
- In order for a DBA to enable new features by changing bind options or make changes to access paths for application packages that are currently being used, the DBA must
  - wait for a window in which the applications aren't running (unreasonable, not realistic)
  - take an application outage (disruptive), or
  - use a cumbersome workaround (unacceptable overhead).

- Applications (24x7 availability) are continuously executed by several threads
- REBIND the packages with the new APPLCOMPAT to deploy new application feature
- REBIND SWITCH when an access path regression is observed

Rebind activity cannot complete (*requires all threads that are executing the package to quiesce)*

With the requirement for 24x7 application availability, it is nearly impossible for the DBA to find a time when the rebind can run in an efficient manner.

Note: Db2 11 REBIND break in addresses RELEASE(DEALLOCATE) packages


Thread 1 executing package A
Thread 2 executing package A
REBIND PACKAGE A
Thread 3 executing package A
REBIND PACKAGE A
REBIND PACKAGE A
REBIND PACKAGE A

# Db2 Problem Statement (specific)

Threads are executing a package. Concurrently, a REBIND PACKAGE subcommand is issued against the same package and fails with the following message:

```
DSNT500I -DB2A DSNTBRB2 RESOURCE UNAVAILABLE
          REASON 00E30083
          TYPE 00000801
          NAME COLL1.TEST19.1A92DAA90F08133F
```

# Solution Overview

**To improve usability, it'd be ideal if Db2 can allow REBIND to work concurrently with application execution by generating a new copy. The new copy of the package will be phased in for execution for future threads.**

**A Db2 for z/OS DBA can enable new Db2 application features for existing packages via REBIND without waiting for a window of opportunity, without incurring an outage, without performing a complex workaround, and without impacting application performance.**

# Example Scenarios

# As-is scenario 1

1. *Application developer implements and tests a new dynamic SQL statement (SELECT LISTAGG(C1) FROM T1) in a JCC application. The LISTAGG BIF is available with APPLCOMPAT(V12R1M501) in test system.*
2. *System administrator activates function level V12R1M501 in production system*
3. *Application developer codes the new SQL syntax and requests the JCC packages to be rebound with APPLCOMPAT(V12R1M501) in production system*
4. *The DBA quiesces all threads executing JCC applications, then rebinds the packages with APPLCOMPAT(V12R1M501)*
5. *Application developer schedules the application to be executed in production system*

# As-is scenario 2

1. *Application developer implements and tests a new dynamic SQL statement (SELECT LISTAGG(C1) FROM T1) in a JCC application. The LISTAGG BIF is available with APPLCOMPAT(V12R1M501) in test system.*

2. *System administrator activates function level V12R1M501 in production system*

3. *Application developer codes the new SQL syntax and requests the JCC packages to be rebound with APPLCOMPAT(V12R1M501)*

4. *The DBA binds copy the JCC packages to another collection ID.*

5. *The DBA inserts a row for CURRENT PACKAGE PATH in the Db2 Profile table with the new collection ID and restarts profile. - Or –*

6. *The new collection ID is specified in the JCC DataSource property and application server is recycled*

# As-is scenario 3

1. *The DBA rebinds an application package with static SQL statements on the weekend while application is not running. New access paths are chosen.*

2. *Monday morning, applications are executed and experience performance regression.*

3. *The DBA brings down all threads executing the application in order to rebind switch the package back to the previous copy*

4. *Applications are executed again with the previous access paths after the outage.*

# To-be Scenarios

# To-be scenario 1

1. *Application developer implements and tests a new dynamic SQL statement (SELECT LISTAGG(C1) FROM T1) in a JCC application. The LISTAGG BIF is available with APPLCOMPAT(V12R1M501) in test system.*
2. *System administrator activates function level **V12R1M505** in production system*
3. *Application developer requests the JCC packages to be rebound with APPLCOMPAT(V12R1M501) in production system.*
4. *Threads are executing JCC packages with APPLCOMPAT(V12R1M500).*
5. *The DBA issues the REBIND PACKAGE command with APPLCOMPAT(V12R1M501) for the JCC packages successfully. New copies of the packages are committed.*
6. *New threads executes package using the copy with APPLCOMPAT(V12R1M501), including the new LISTAGG application.*
7. *Threads existed prior to REBIND continue to execute packages using the copy with APPLCOMPAT(V12R1M500). Eventually when these threads finish, Db2 can free these old copies.*

# To-be scenario 2

1. *The DBA rebinds an application package with static SQL statements on the weekend while application is not running. New access paths are chosen.*
2. *On Monday morning, applications were executed and experience performance regression.*
3. *The DBA issues the REBIND SWITCH command to switch the package back to the previous copy. The command is successful and creates a new copy with the previous access paths.*
4. *Existing threads continue to execute with regressed access paths created on the weekend.*
5. *New threads execute the package copy with the previous (stable) access paths.*

# REBIND Phase-in

# REBIND PACKAGE existing support

- PLANMGMT bind option allows Db2 to create copies of the rebound package
- Copies are created prior to changing the package
- Copies contain older (stable, preferred) access paths
- Can switch between copies of package to overcome performance regression

# REBIND PACKAGE existing support

- PLANMGMT(EXTENDED)
  - CURRENT (copy ID 0) in SYSPACKAGE, SYSPACKDEP, SPTR
  - ORIGINAL (copy ID 2) in SYSPACKCOPY, SYSPACKDEP, SPTR
  - PREVIOUS (copy ID 1) in SYSPACKCOPY, SYSPACKDEP, SPTR
- PLANMGMT(BASIC)
  - CURRENT and PREVIOUS copies
- PLANMGMT(OFF)
  - CURRENT copy only
  - Enforced when change OWNER, QUALIFIER, PATH, PATH_DEFAULT, DISABLE/ENABLE, IMMEDIATEWRITE, SYSTIMESENSITIVE, BUSTIMESENSITIVE, ARCHIVESENSITIVE

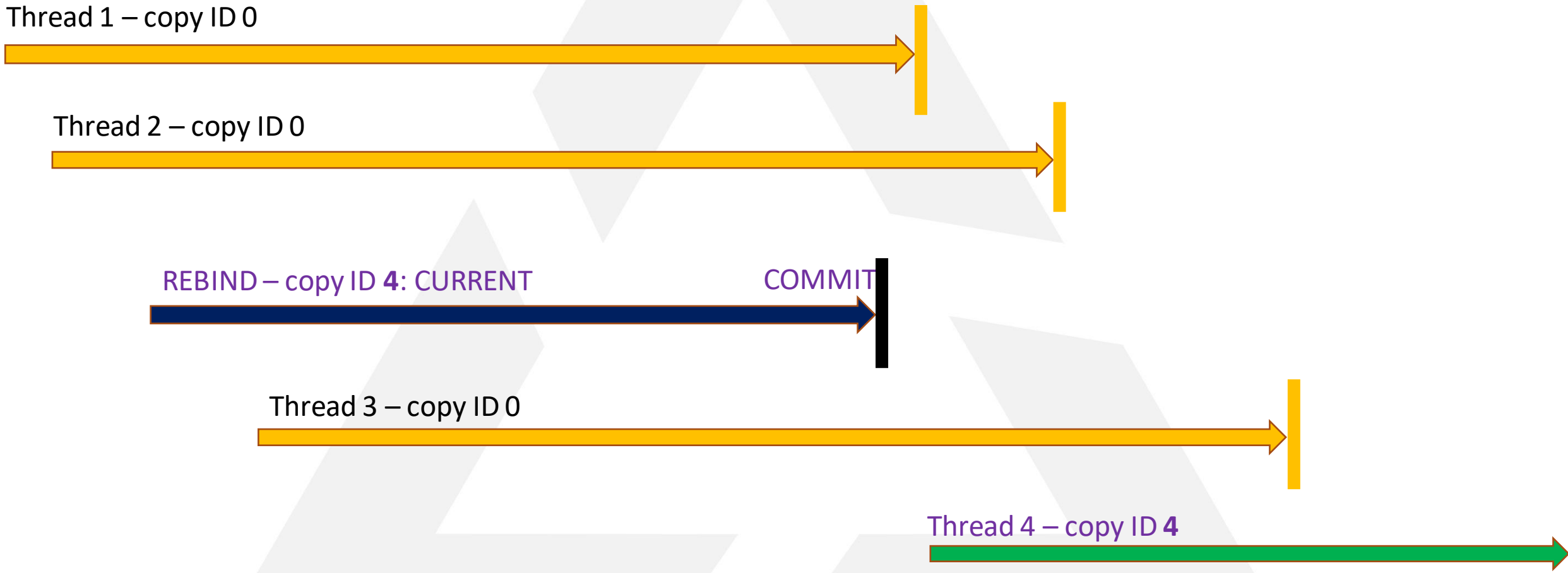
# REBIND PACKAGE existing support

- REBIND SWITCH to swap between copies
- FREE PACKAGE PLANMGMTSCOPE(ALL) deletes all copies
- FREE PACKAGE PLANMGMTSCOPE(PREVIOUS|ORIGINAL|INACTIVE)
- EXPLAIN PACKAGE COPY(PREVIOUS|ORIGINAL)
- Serialization between executing threads and DDL, Rebind via package lock
  - Execution: S state
  - REBIND: SIX state
  - DDL: X state
- Copy ID 3 used for APREUSE(PREVIOUS|ORIGINAL) bind option

# REBIND Phase-In

- Threads executing CURRENT package copy (copy ID n)
- REBIND
  - Obtain SIX lock* followed by U lock on package name
  - Create a new CURRENT copy of the package with a new copy ID (n+1)
  - Move copy ID n to SYSPACKCOPY as the phased-out copy
  - Replicate phased-out copy (n) to copy ID 1 (previous) and 2 (original) if needed
- New threads load and execute new CURRENT copy (n+1) when that copy is committed

# REBIND Phase-In Example



# REBIND Phase-In

- Current copy
  - 1 row in SYSPACKAGE. Note COPYID column
  - Copy ID generated as 0, 4, 5, 6, .., 16 then wrap back
- Phased-out copies
  - In SYSPACKCOPY (with COPYID other than 1, 2)
  - Cleaned up on subsequent REBIND
- EXPLAIN PACKAGE COPY *copy-id*
  - CURRENT: the copy ID in SYSPACKAGE.COPYID column (0 or non-0)
  - Omit *COPY*: copies CURRENT, PREVIOUS, ORIGINAL

# REBIND SWITCH Phase-In

- Threads are executing CURRENT copy ID (n)
- REBIND SWITCH:
  - Obtain SIX lock* followed by U lock on package
  - Copy PREVIOUS or ORIGINAL to new CURRENT copy ID (n+1)
  - Copy n becomes the phased-out copy
  - Replicate phased-out copy (n) into PREVIOUS and ORIGINAL if needed
- New threads can execute new CURRENT (n+1)

# REBIND Phase-In Eligibility

- Requires FL 505 or above -> **an Always-On feature**
- Initial support:
  - PLANMGMT(EXTENDED) APREUSE(NO)
  - PLANMGMT(EXTENDED) APREUSE(YES) APREUSESOURCE(CURRENT)
- Limitation:
  - Package for native SQL routine & advanced trigger

# Phased-out Copy cleanup

- Phased-out copy clean up on subsequent REBIND
  - Compare current thread's allocation time to phased-out copy's time
  - Query package lock holders
  - DSNT500I message with new reason code 00E30307 (max number of copy IDs has been reached)
  - New IFCID 393 to identify the oldest thread which prevents the phased-out copies from being deleted
    (long running thread with RELEASE(DEALLOCATE), uncommitted thread, etc)

# IFCID 393

- Which thread to recycle?

| | | |
|---|---|---|
| QW0393 | DSECT | IFCID(QWHS0393) |
| QW0393COLLID_Off | DS H | Offset from QW0393 to collection ID |
| * | | |
| QW0393PK_Off | DS H | Offset from QW0393 to package ID |
| * | | |
| QW0393CONTKN | DS CL8 | Package consistency token |
| QW0393THDTKN | DS F | Thread token of the thread that prevents phased-out copies from being freed |
| * | | |
| QW0393MEMBER | DS CL8 | Member name where thread is |
| QW0393THDTS | DS CL13 | Thread's package allocation timestamp |
| | DS CL3 | Available |
| QW0393THDCT | DS D | (S) |

# Rebind Phase-In

- Which copy does a thread execute?
- Package account trace

```
QPAC                    DSECT
QPACPKNM                DS    0CL60          /*PACKAGE NAME                    */
QPACLOCN                DS    CL16           /* %U LOCATION NAME               */
*                                            /*   Truncated if QPACLOCN_Off¬=0  */
QPACCOLN                DS    CL18           /* %U PACKAGE COLLECTION ID       */
*                                            /*   Truncated if QPACCOLN_Off¬=0  */
QPACPKID                DS    CL18           /* %U PROGRAM NAME                */
*                                            /*   Truncated if QPACPKID_Off¬=0  */
QPACCONT                DS    CL8            /*CONSISTENCY TOKEN - 64 BIT      */
*                                            /*UNSIGNED BINARY INTEGER         */
*                                            .
QPACA313                EQU *
QPAC_PIPE_WAIT          DS     XL8           /* accumulated wait time for a pipe while  */
*                                            /*  executing this package                 */
QPAC_PIPEWAIT_COUNT DS F                     /* number of wait trace events processed  */
*                                            /* for waits for a pipe while executing   */
*                                            /* this package                           */
QPAC_COPYID             DS  F                /* Package copy ID                        */
QPACEND                 DS     0C
```

# Catalog Query

```
SELECT NAME, COPYID, VALID, TIMESTAMP, BINDTIME, PLANMGMT
        FROM SYSIBM.SYSPACKAGE
         WHERE NAME='xx';

SELECT NAME, COPYID, VALID, TIMESTAMP, BINDTIME
        FROM SYSIBM.SYSPACKCOPY
         WHERE NAME='xx';

SELECT BNAME, BTYPE, DTYPE, COPYID
        FROM SYSIBM.SYSPACKDEP
         WHERE DNAME='xx';

SELECT HEX(SPTRESV) AS COPYID, HEX(SPTSEC) AS SECTION,
            HEX(SPTSEQ) AS SEQUENCE
        FROM SYSIBM.SPTR
         WHERE SPTNAME='xx';
```

# Catalog Query

SYSPACKAGE

| COLLID | NAME | COPYID | APPLCOMPAT | Note |
|---|---|---|---|---|
| NULLID | SYSLH100 | 0 | V12R1M500 | Current copy |

After REBIND APPLCOMPAT(V12R1M501)

SYSPACKAGE

| COLLID | NAME | COPYID | APPLCOMPAT | Note |
|---|---|---|---|---|
| NULLID | SYSLH100 | 4 | V12R1M501 | Current copy |

SYSPACKCOPY

| COLLID | NAME | COPYID | APPLCOMPAT | Note |
|---|---|---|---|---|
| NULLID | SYSLH100 | 0 | V12R1M500 | Phased-out copy |
| NULLID | SYSLH100 | 1 | V12R1M500 | Previous copy |
| NULLID | SYSLH100 | 2 | V12R1M500 | Original copy |

# Catalog Query

## After REBIND SWITCH(ORIGINAL)

SYSPACKAGE

| COLLID | NAME | COPYID | APPLCOMPAT | Note |
|---|---|---|---|---|
| NULLID | SYSLH100 | 5 | V12R1M500 | Current copy |

SYSPACKCOPY

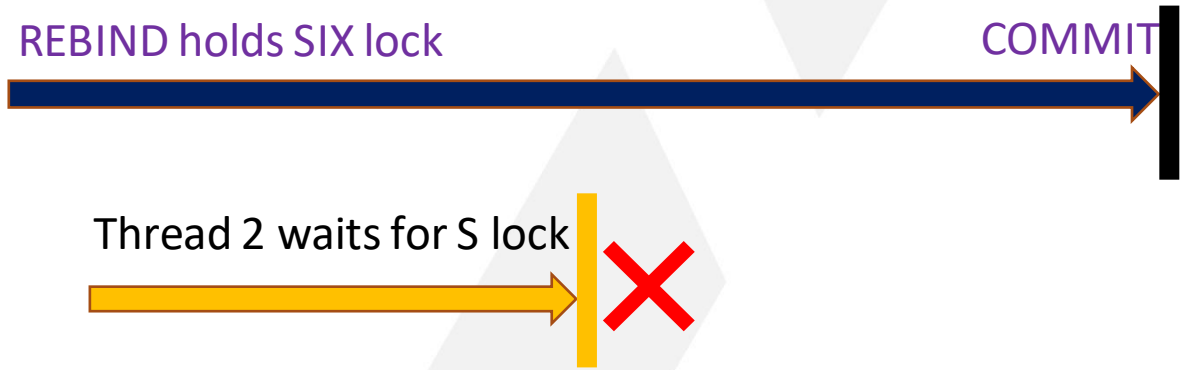
| COLLID | NAME | COPYID | APPLCOMPAT | Note |
|---|---|---|---|---|
| NULLID | SYSLH100 | 4 | V12R1M501 | Phased-out copy |
| NULLID | SYSLH100 | 1 | V12R1M501 | Previous copy |
| NULLID | SYSLH100 | 2 | V12R1M500 | Original copy |

# Rebind Phase-In Recommendations

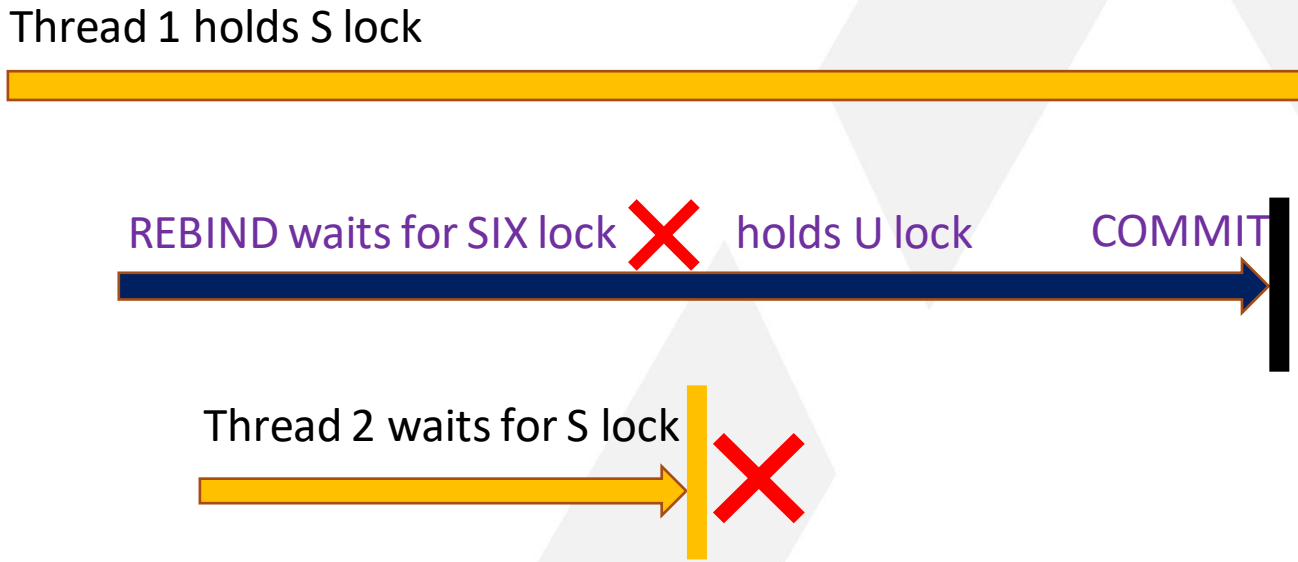
- Discourage use of –F DDF,PKGREL(BINDPOOL) to prevent DBAT executes new copy then a phased-out copy
- High performance DBAT honors RELEASE(DEALLOCATE) and is recycled after 200 UOW's or 120 seconds (subsystem parameter POOLINAC)
- CICS protected threads should set REUSELIMIT = 1000 (default) to pick up new copy
- IMS Fast Path:
  - "true" Wait For Input (WFI) region type: threads can stay for weeks
  - Use Pseudo WFI region type instead (2-3 reuses before recycle)

- PTF UI73874 (APAR PH28693): Rebind always creates a new copy to ensure true concurrency with subsequent executing threads

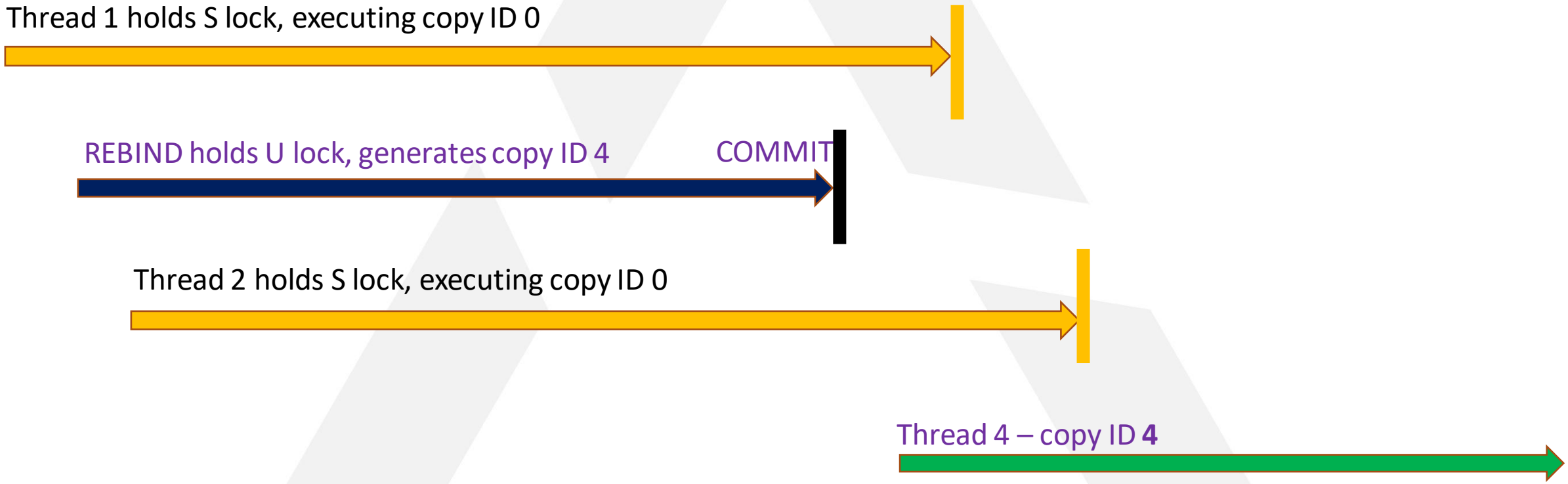
# Rebind Phase-in with SIX lock



# Rebind Phase-in with SIX lock



# Rebind Phase-in without SIX lock (APAR PH28693)



# Performance Study

| Workload | Description |
|---|---|
| OLTP+ Batch | • A batch job at the beginning of the run<br>• OLTP like workload<br>• Rebind while workloads are running. |

•Distributed Program (200 threads)- Perform unit of work (OLTP trans)

•Rebind / Rebind /Rebind/……..…/Rebind

•Distributed Program 2- Select-no Commit(OLTP+BATCH ONLY)

- Note: Rebind JCC package SYSLH200 only because the distributes application only touches this package.
- Package SYSLH200 is rebound every 45 seconds during the measurement period.

# Performance Observation

| Function Level | Elapsed Time (no Rebind vs Rebind) | CP Time (no Rebind vs Rebind) | Rollback (no Rebind vs Rebind) |
|---|---|---|---|
| FL500 | Increases from 51ms to 32s | Increases from 76µs to 118 µs | 1/3 transactions |
| FL505 | Increases from 51ms to 114ms | No CPU impact | 0.17% transactions |
| FL505+APAR PH28693 | No elapsed time impact | No CPU impact | No roll back |

# Technical Data- BATCH+OLTP Workload

| | V12R1M500 | | V12R1M505 | | V12R1M505+APAR | |
|---|---|---|---|---|---|---|
| | NO REBIND | REBIND | NO REBIND | REBIND | NO REBIND | REBIND |
| CL1ET | 0.050875 | 31.678713 | 0.050833 | 0.113982 | 0.050953 | 0.050955 |
| CL2ET | 0.000174 | 29.572873 | 0.000170 | 0.063355 | 0.000186 | 0.000187 |
| CL1CP | 0.000119 | 0.000169 | 0.000118 | 0.000125 | 0.000130 | 0.000130 |
| CL2CP | 0.000067 | 0.000112 | 0.000067 | 0.000067 | 0.000072 | 0.000073 |
| CL1SECP | 0.000169 | 0.000200 | 0.000168 | 0.000179 | 0.000187 | 0.000187 |
| CL2SECP | 0.000090 | 0.000128 | 0.000089 | 0.000090 | 0.000097 | 0.000097 |
| TotalCL1CP | 0.000139 | 0.000181 | 0.000138 | 0.000147 | 0.000153 | 0.000153 |
| TotalCP2CP | 0.000076 | 0.000118 | 0.000076 | 0.000076 | 0.000082 | 0.000083 |
| Rebind Success | NA | 0 | NA | 13 | NA | 13 |
| Rebind Error | NA | 7 | NA | 0 | NA | 0 |
| Transaction Commit | 2358744 | 2221 | 2360207 | 1052726 | 2352627 | 2352559 |
| Transaction Rollback | 0 | 1200 | 0 | 1819 | 0 | 0 |

# Question and Answer

- Contact Information
  - Tammie Dang, tammied@us.ibm.com

# THANK YOU