DB2 for z/OS:
Thread Reuse and BIND option RELEASE(DEALLOCATE)

John Campbell
Distinguished Engineer
IBM DB2 for z/OS Development
campbelj@uk.ibm.com
Disclaimer:

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance Disclaimer:

This document contains performance information based on measurements done in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the numbers stated here.
Agenda

• Primer on thread reuse with RELEASE bind option

• Considerations and limitations
“Resources”: Static SQL

- Static statements
  - Packages and statements
  - Parent locks
  - Index look-aside buffer
  - Dynamic prefetch tracking via sequential detection
RELEASE - BIND and REBIND Option

- Determines when to release the resources that a program uses
  - RELEASE(COMMIT) - Releases resources at commit point
  - RELEASE(DEALLOCATE) - Releases resources when thread terminates (child page/row locks still released at commit)
  - RELEASE(INHERITFROMPLAN) – Support added by APAR PM07087

- Default behavior
  - BIND PLAN - COMMIT
  - BIND PACKAGE - plan value
  - REBIND PLAN/PACKAGE existing value
  - DB2Binder Utility for JDBC and SQLJ
    - DEALLOCATE is default in DB2 10
    - COMMIT is default in DB2 9 and earlier releases
Program A
Stmt 1: Insert into table1
Stmt 2: Insert into table2

EDM_SKELETON_POOL
SKCT/SKPT

SPT01
I/O

V10 Thread storage above the bar
V9 EDM thread pool below the bar

CT
PT
SQL Stmt1
SQL Stmt2

Program A:
“Insert into table 1”

Thread allocation
Sign on, authorization checking
Locate SKCT and SKPT
Allocate CT/PT and STMT1
Start Insert STMT1
  Lock table space 1
  Lock data page
End STMT1

“insert into table 2”
Allocate STMT2
Start Insert STMT2
  Lock table space 2
  Lock data page
End STMT2
COMMIT ;
Package Allocation and Commit

Program A
Stmt 1: Insert into table1
Stmt 2: Insert into table2
Commit ;

EDM_SKELETON_POOL
SKCT/SKPT

SPT01

I/O

Expensive Operation!

CT
PT
SQL Stmt1
SQL Stmt2

V10 Thread storage above the bar
V9 EDM thread pool below the bar

RELEASE PT, statements
Unlock data pages
Unlock TS1, TS2

COMMIT

Lock TS 1
Lock table1 data page

Lock TS 2
Lock table2 data page
Thread Reuse and RELEASE(DEALLOCATE)

- Thread reuse eliminates CPU cost of DB2 thread allocation and deallocation
  - CICS
    - Protected ENTRYs
    - Organic reuse of ENTRYs (or POOL)
  - IMS/TM
    - Fast Path (IFP) regions
    - Wait-For-Input (WFI) regions
    - Pseudo Wait-For-Input (PWFI) regions
  - DDF
    - High Performance DBATs
    - WebSphere Type 2 local connections
    - Batch with intermediate commits
- Use of RELEASE(DEALLOCATE) coupled with effective thread reuse i.e., thread persistence
  - Further reduces the CPU cost with potential for significant savings (up to 10% plus)
  - Resources are held until thread deallocation
  - Without thread reuse RELEASE(COMMIT) vs. RELEASE(DEALLOCATE) is a moot point for discussion
Limitations and Considerations

- Virtual and real storage
  - DB2 9 and earlier - DBM1 virtual storage below the 2G bar and real storage
    - Package information is stored in EDM pool below the bar
  - DB2 10 after REBIND - real storage usage
    - Package information is stored in thread storage above the bar in DB2 9 and earlier releases
  - Accumulated DB2 object control blocks
    - Virtual, real, potentially CPU cost for scanning the objects built up under the thread

- Recommendations
  - Design for thread reuse for
    - High volume simple transactions
    - Complex transactions at a reasonably high rate
  - Selectively use RELEASE(DEALLOCATE) on high use packages – use % of Total Acctg Class 7 CPU
    - DBM1 31-bit virtual storage constraint (DB2 9)
    - Real storage constraint (DB2 10)
  - Use CICS or WebSphere parameter to periodically clean up and rejuvenate threads (thread deallocation)
Considerations for Clean Up

• **REUSELIMIT (0-10,000) in CICS TS V4R2 - default 1000**
  - Number of times a thread can be reused before it is terminated
  - Use default and monitor DB2 storage usage and adjust the number if needed

• **WebSphere Type 2 connection Aged Timeout - default 0**
  - The interval in seconds before a physical connection is discarded
  - Consider setting WAS “aged timeout” to less than 5 min, recommend using 120 secs to reduce exposure of long living threads

• **DB2 10 High Performance DBATs (threads)**
  - Thread will go inactive every 200 commits
  - No user control for this value
Considerations - Concurrency

- More persistent threads with RELEASE(DEALLOCATE) is also trade off with concurrency
  - BIND/REBIND
  - SQL DDL
  - Online REORG which invalidates packages
- For RELEASE(DEALLOCATE) some locks are held beyond commit until thread termination
  - Mass delete locks (SQL DELETE without WHERE clause)
  - Gross level lock acquired on behalf of a SQL LOCK TABLE
  - Table space defined with LOCKSIZE TABLESPACE | TABLE
  - Note: no longer a problem for gross level lock acquired by lock escalation

- **DO YOUR HOMEWORK BEFORE USING PERSISTENT THREADS WITH BIND OPTION RELEASE(DEALLOCATE)**