The Nuts and Bolts of IMS Lock Management

Kevin Stewart
IMS Technical Support

kevstew@us.ibm.com

IBM Systems Technical Events
ibm.com/training/events

- Central Canada DB2 Users Group

Please note

- IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.
- 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 is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including 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 results similar to those stated here.

Agenda

- IRLM vs PI
- Locking Basics
- IRLM Flow Overview
- Deadlock and Timeout
- Measurements

IMS Lock Manager Choices

- Program Isolation (PI)
  - Part of IMS
    - No separate address space
    - No cross memory access required
    - No CF structures required
  - Lock implemented in 31-bit storage
  - Matrix Algorithm
    - Limits maximum waiters to 63 for any lock
      - U2478 abend if exceeded
    - Instantaneous deadlock detection
- No support for Data Sharing
- No support for Lock Timeout

IMS Lock Manager Choices

- IRLM
  - Independent Component
    - Included with IMS DB
    - Separate address space
    - Accessed cross memory (PC instruction)
    - No CF structures required unless Data Sharing
  - Lock implemented in:
    - 64-bit IRLM private storage
    - CF Lock Structure (if SCOPE=GLOBAL)
    - No limit on maximum waiters for any lock
    - Cyclical deadlock detection
- Support for Data Sharing
- Support for Inter-System Notify/Messaging
- Support for Lock Timeout

IRLM vs PI

- Without Datasharing
  - IRLM has advantages
    - Timeout support
    - No limit on waiters for lock (U2478 abend)
    - Many more total locks (U0775, U3301)
    - Separate address space (RAS)
    - Cross Memory access (PC instruction)
    - No CF structures required unless Data Sharing
  - Locks implemented in:
    - 64-bit IRLM private storage
    - CF Lock Structure (if SCOPE=GLOBAL)
    - No limit on maximum waiters for any lock
    - Cyclical deadlock detection
- Required for Datasharing

IRLM vs PI

- Without Datasharing
  - PI has advantages
    - Slightly less CPU
    - Instant Deadlock Detection, vs Cyclical
    - Lock Granularity
      - Multiple concurrent access with single DB record
        - » Update different child segments of same root concurrently from different dependent regions
        - » This probably rare?

IRLM vs PI

- Recommendation – use IRLM
  - IRLM is Industrial Strength Lock Manager
    - Even if not sharing – CF structure not required
    - Eliminate U2478 abends
    - Allow order of magnitude more locks
      - 64-bit storage
    - Lock timeout support
      - SMF 79.15 records always produced
        - » More later
      - Optional waiter timeout (like DB2)
    - RAS
      - Locks kept in separate address space
      - Improved servicability - CTRACE

Conversion Experience

- Many customers have converted to IRLM on road to Data Sharing
  - Very few have any application issues
  - Deadlocks
    - Few customers have any issues
    - IRLM Deadlock cycle time can be set very low
    - Cost is CPU
    - Deadlocks are already very expensive
      - Message Region TCB abend/collapse/reattach
      - Reload Preload list
      - If Deadlocks are reason for PI, better to fix the deadlocks

What is a lock?

- Arbitrary byte string that represents a resource
  - Usually, a DB record or DB data block/CI
  - DB identifier is Global DMB Number assigned by DBRC
    - Not shared / not DBRC registered – Local DMB number
  - Additional data:
    - Lock Type
    - Dataset with Database (usually relative number)
      - Partition, DEDB Area, DS Group
    - RBA or Key
    - Other
    - Example
      - » 09C943CFA7800101D700000000000000
  - IRLM doesn't know or care what this string represents
  - Variable length, first byte is total length

What is a Lock - Levels

- IRLM lock compatibility matrix

Lock Level	2	3	4	6	8
2 – read	Y	Y	Y	Y	N
3 – erase	Y	Y	N	N	N
4 – share	Y	N	Y	N	N
6 – update	Y	N	N	N	N
8 - exclusive	N	N	N	N	N

IRLM Lock Request

- Input
  - Lock Name
  - Lock Hash Value
    - DBMS (IMS) must hash Lock Name to 4-byte value
  - Lock Level
  - Owning Subsystem (DBMS)
  - Owning Work Unit
    - Usually, PST address
  - Request Info
    - Conditional or Unconditional

IRLM Lock Request

- Output
  - Lock Token – 4-byte value that represents granted lock
    - Required to Change/Alter/Unlock
    - DBMS must preserve for future use
  - Return Code
    - Reason(s) request failed

Lock Manager Request Types

- LOCK
- UNLOCK
- RALL
  - Release all locks held by OWU
- CHANGE
  - Transfer Lock Ownership
- NOTIFY (IRLM only)
  - Messaging
  - Partners obtain SHR lock on same lock name
  - NOTIFY message uses lock token as input, is sent to any DBMS with interest in lockname
  - Used for dataset extend notifications, Fast Path control block sync, SVSO, utility processing
  - Also used for GLOBAL commands (such as DBR or STA)

IRLM Lock Structure

- Lock Table Entries
  - One entry per held lock (2 bytes)
  - HASH value on lock request used to determine LTE entry
- Record Table Entries
  - One entry per Modify Lock (about 540 bytes)
    - Lock that represents altered data
    - Lock that must be RETAINed if DBMS fails
    - Lock that must persist if IRLM fails
- Division of structure space set by LTE= in IRLM procedure
  - Use CFSIZER tool to initially size Lock Structure
- IRLM keeps all locks in 64-bit private storage
  - Modify and non-Modify

IRLM Lock Processing

- HASH value used to access LTE slot
  - If slot is free, lock can be immediately granted
  - If slot is not free
    - IRLMs must exchange actual lock name via XCF
    - Lock actually held – suspend requestor (or bad RC)
    - Lock not held – false contention, grant lock
      - False contention can be:
        - » Too small Lock Table
        - » Poor DBMS Hash routine
        - » Note, it's not easy or safe to change a DBMS hash routine

IMS – IRLM Interface

- At IRLM Identify Time, IMS passes:
  - Address of Suspend Routine
  - Address of Resume Routine
- Lock Request
  - PC to IRLM
    - Lock can be granted, return token
    - Suspend required, call Suspend Routine under IMS's TCB
      - ISERWAIT
- Lock Granted
  - Drive IMS Resume Exit in SRB mode
  - Exit will IPOST the PST
  - PST redispatched in Suspend Routine, return to IRLM, which returns back to IMS request

Deadlocks and Long Locks

- At IRLM Identify Time, IMS passes:
  - Address of Deadlock Routine (DFSDLKX0)
  - Address of Long Lock Routine (DFSLLKX0)
  - Timeout Value
    - If LOCKTIME is coded: smaller of value coded for
      - BMP(xx) or MSG(xx)
    - If LOCKTIME is not coded
      - 300 seconds

Deadlocks and Long Locks

- Deadlock Cycle
  - DEADLOK=(x,y) on IRLM Procedure
    - x is local deadlock interval
      - x >0<99, value is seconds
      - x > 99, value is milliseconds
    - y is number of local intervals per global interval
      - No matter what is coded, 1 is used
  - At the specified interval, IRLM checks for deadlocks
    - Deadlock found, call DBMS Deadlock Exit
      - Exit is passed information on Locks involved in Deadlock
    - DBMS exit instructs IRLM on action to take
      - Reject lock request (IMS always returns this)
      - Do nothing

Deadlocks and Long Locks

- IMS Deadlock Processing
  - DFSDLKX0 called
    - Obtains storage to save deadlock data
    - Save address of this storage in PST
    - Saves deadlock data into the storage
    - Instructs IRLM to reject request, and returns to IRLM
  - IRLM returns to lock requestor with deadlock rejection return code
    - DFSLMGR0, requestor
      - Sets up U0777 abend
      - Returns to call handler
    - Call handler issues U0777
    - ABTERM processing uses the saved deadlock data
      - Produce 67FF log records to document deadlock
      - If message driven flow, input message will be requeued for retry

Long Lock Processing

- During each Deadlock Cycle, IRLM compares time lock request has been waiting, to the LOCKTIME passed by DBMS at Identify. If request has been waiting >= LOCKTIME:
  - Call DBMS Long Lock Exit
    - Pass information on Blocker(s)
  - DBMS Long Lock Exit will instruct IRLM on action
    - Do nothing (continue wait)
    - Reject Request with Timeout RC
- Therefor
  - LOCKTIME should be integer multiple of DEADLOK
    - LOCKTIME is seconds…DEADLOK may be milliseconds

IMS Long Lock Processing

- If LOCKTIME is not coded in DFSVSMxx
  - IMS passes 300 second to IRLM as LOCKTIME
  - LOCKTIME flags are not set in IMS internally
- If LOCKTIME is coded in DFSVSMxx
  - IMS passes smaller of BMP(xx) or MSG(xx) to IRLM as LOCKTIME
- Address of DFSLLKX0 passed as Long Lock Exit

IMS Long Lock Processing

- IRLM Deadlock Cycle Process detects lock request has waited longer than LOCKTIME
  - Call DBMS Long Lock Exit (DFSLLKX0)
- DFSLLKX0
  - If LOCKTIME not active in IMS Internally
    - Write SMF79.15 records documenting waiter, blocker
  - If LOCKTIME is active in IMS Internally
    - Determine waiter region type (BMP, MSG)
    - Compare BMP or MSG value to wait time
      - » If exceeded
        - Write SMF79.15
        - Optionally build DFS2291I message, save in PST
        - Set RC to IRLM to reject request as timeout

IMS Long Lock Processing

- IRLM returns to lock requestor with timeout RC
  - DFSLMGR0 (requestor)
    - Sets up U3310 Return Code or ‘BD’ status code
    - Optionally issues DFS2291I message
      - » DFSDFxx, MSG2291= ISSUE|SHORT
      - » Identifies Blocker
    - Returns to call handler
      - » Abend if U3310 setup
- It is the victim (waiter) that abends, not blocker
  - Can automate on DFS2291I to ABDUMP blocker
  - PI86409 adds SUBSYS to short form of DFS2291I

LOCKTIME Considerations

- There are two LOCKTIMEs:
  - IRLM's value
  - IMS's value(s)
  - Both can be changed dynamically
    - F IRLM,SUBSYS=iiii,TIMEOUT=sss
    - UPDATE IMS LOCKTIME
  - IRLM value should be integer multiple of IMS value(s)

LOCKTIME Considerations

- LOCKTIME not active in IMS
  - Can reduce IRLM value to collect SMF79.15 data to analyze performance issues at specific times (or during problem)
- LOCKTIME active in IMS
  - Reducing IRLM value isn't useful, as SMF79.15 produced only at IMS time exceeded

LOCKTIME Considerations

- Interpreting SMF 79.15
  - Real Time – RMF ILOCK ALL command
  - Post Incident
    - IMS Problem Investigator Tool
    - Other Vendor Tools
    - Own program (ERBSMF macro)
    - Print with IDCAMS and IEBEYEBALL
    - DFSSMF79 DSECT in source of DFSLLKX0

Other Locking Considerations

- LOCKMAX
  - IMS Parameter to limit number of locks a dependent region can obtain
  - Even IRLM can run out of lock space
    - CF structure space
  - Many locks can cause contention, other performance issues
  - Code on PSBGEN, or in dependent region JCL

Measurements

- Locking Statistics
  - IMS records highwater mark count of locks held in
    - X'37',x'38' ,x'5937',x'5938' log records
    - IMS PA reports
  - Locks may be released during a commit cycle
    - As PCB position moves for Full Function
    - Buffer Steal at NBA, for FastPath
  - HWM count doesn't reflect total lock requests
    - IMS V14 PTF UI47337 adds Lock Request counts to 56FA TLS log records
    - Requires TLS active
    - Both PI and IRLM supported
    - May help identify opportunities for PROCOPT=GO
    - GDPS – remote lock structure

Questions?

Notice and disclaimers

Notice and disclaimers continued

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular**, **purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.