IDUG
Leading the DB2 User Community since 1988

IDUG Db2 Tech Conference NA
Philadelphia, PA | April 29 - May 3, 2018

#IDUGDb2

Deep Dive into Locks and Logs of Db2 v11.1 pureScale

Mohankumar Saraswatipura
Robert (Kent) Collins

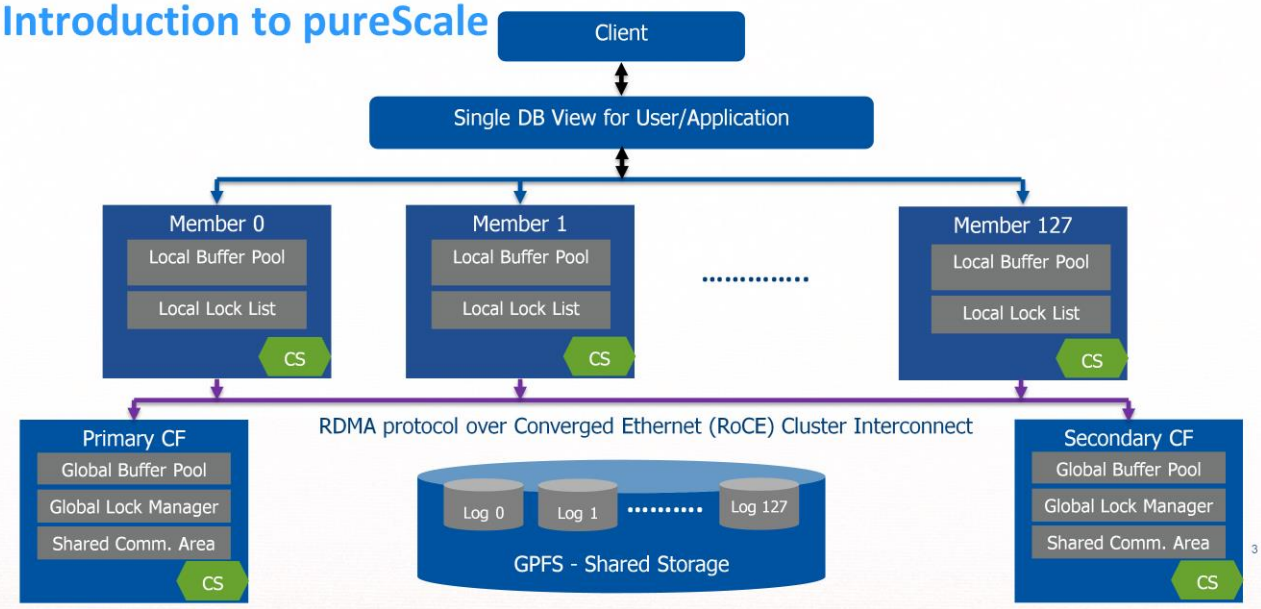
Session code:

1

Agenda

- Introduction to pureScale architecture
- Understanding the lock management in pureScale
- Understanding Page Reclaims and EHL
- Understanding pureScale transaction logging
- Understanding LSN and LFS numbers in pureScale
- Understanding the log stream merge operations
- Understanding the member and group crash recovery operations

Introduction to pureScale



What is a member? (1/3)

A Db2 member is the core processing engine within the pureScale cluster.

Similar to single partition Db2, it contains: a Db2 system controller process (db2sysc) and a watch dog process (db2wdog), Local buffer pools, lock list, database heap, log buffer, sort heap and application heap etc.

New Engine Dis-patchable Units (EDU's) within pureScale are:

db2castructevent	db2LLMn1	db2LLMn2
db2LLMng	db2LLMrl	db2LLMrc

What is a member? (2/3)

- db2castructevent - Reads the state of the CF links from two files

  (1) The pgrp

  (2) The "isOnline"

These two files gets the update from db2clstrRscMon EDU

- db2LLMn1 - Processes the information sent by the global lock manager; there are two of these EDUs on each member, one for the primary CF, and another for the secondary CF
- db2LLMn2 - Processes the information sent by the global lock manager for a special type of lock that is used during database activation and deactivation processing; there are two of these EDUs on each member, one for the primary CF, and another for the secondary CF

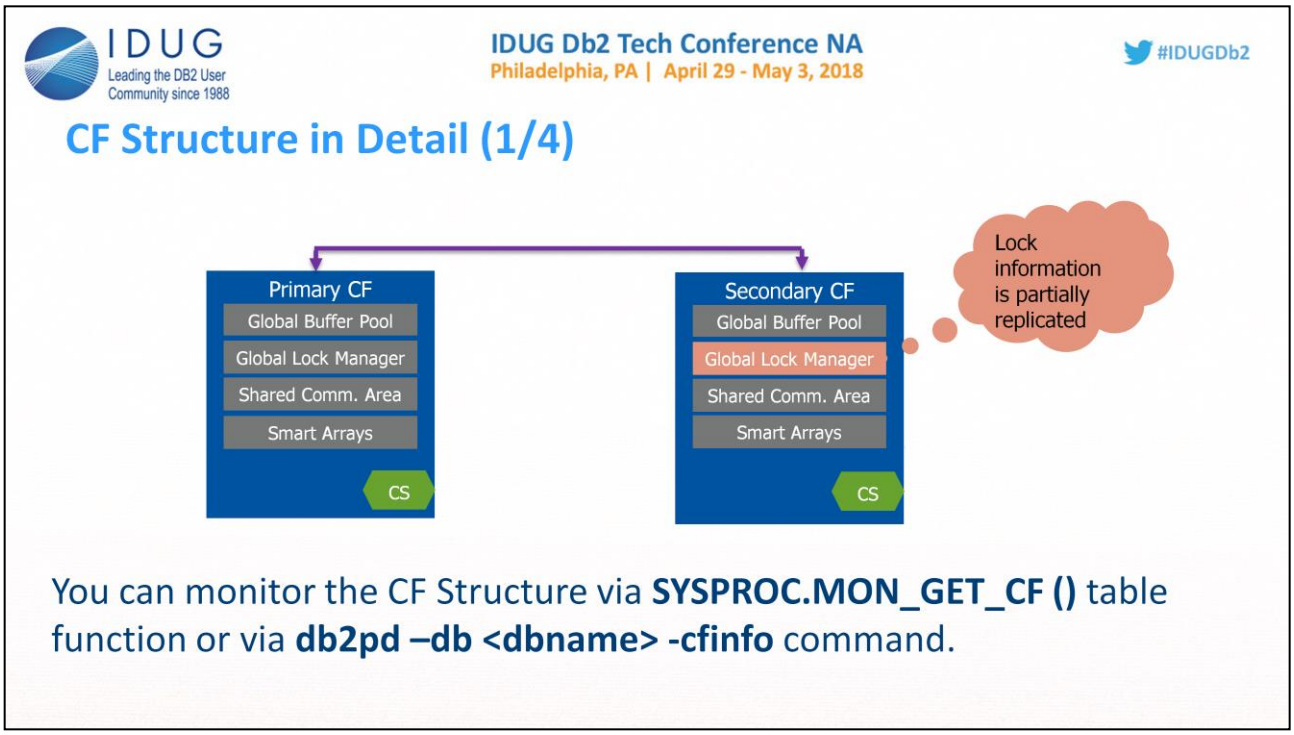
What is a member? (3/3)

- db2LLMng - Ensures that the locks held by this member are released in a timely manner when other members are waiting for these locks
- db2LLMrl - Processes the release of locks to the global lock manager
- db2LLMrc - Processing that occurs during database recovery operations and during CF recovery

What is a CF?

Software technology that assists in global buffer pool management and global locking. The services includes:

- Group Buffer pool Management (GBP)
- Global Lock Management (GLM)
- Shared Communication Area (SCA)



Information about any modified pages will be sent to primary and the secondary CF's in the cluster.

The Shared Communication Area (SCA) is a per database entity and it contains control block information for tables, indexes, table spaces and system catalogs.

Global Lock Manager (centralized locking mechanism) will process the locking requests by checking any lock conflicts across the members. The primary CF will only replicate the granted update locks to the secondary and read locks will not be replicated due to performance optimization reasons. In case of any role switch between the CF's, the read locks information will be regenerated on the new primary CF by information sent to it by members.

Smart array is actually one per table which facilitates direct updates by members and it is used to identify the lock avoidance opportunities.

You can use the below listed command to understand the allocations

db2pd -db <DBNAME> -cfinfo

```
CF Server (128) Information
  Hostname                    = CF1_hostname
  Management Port             = 56000
  Role                        = Secondary
  CF Host Information
    Virtual Memory Used       = 59903660032
    Virtual Memory Available  = 74582503424
    Real Memory Used          = 59751178240
    Real Memory Available     = 6552379392
    Swap File Used            = 152481792
```

```
    Swap File Available              = 68030124032
    CPU Usage                        = 0
  Total structure memory (4KB)       = 10239232
  Free structure memory (4KB)        = 1263104
  Memory frame Size (4KB)            = 256
  Configured Size (4KB)              = 10239232

  Group Bufferpool Information
    Structure Name                   = db2.db2inst1.<DBNAME>.gbp
    SID                              = 316
    Current Size (4KB)               = 104335
    Target Size (4KB)                = 6080000
    Configured Size (4KB)            = 6080000


  Shared Communication Area Information
    Structure Name                   = db2.db2inst1.<DBNAME>.sca
    SID                              = 314
    Current Size (4KB)               = 15360
    Target Size (4KB)                = 16640
    Configured Size (4KB)            = 16640


  List Structure / Smart Array Information
    Structure Name                   = db2.db2inst1.<DBNAME>.list
    SID                              = 3
    Current Size (4KB)               = 31599
    Target Size (4KB)                = 32000
    Configured Size (4KB)            = 32000


  Lock Structure Information
    Structure Name                   = db2.db2inst1.<DBNAME>.lock
    SID                              = 5
    Current Size (4KB)               = 194623
    Target Size (4KB)                = 1144320
    Configured Size (4KB)            = 1144320

CF Server (129) Information
  Hostname                           = CF2_hostname
  Management Port                    = 56000
  Role                               = Primary
  CF Host Information
    Virtual Memory Used              = 60169306112
    Virtual Memory Available         = 74316857344
    Real Memory Used                 = 60008017920
    Real Memory Available            = 6295539712
    Swap File Used                   = 161288192
```

```
    Swap File Available              = 68021317632
    CPU Usage                        = 0
  Total structure memory (4KB)       = 10247936
  Free structure memory (4KB)        = 1269760
  Memory frame Size (4KB)            = 256
  Configured Size (4KB)              = 10247936

  Group Bufferpool Information
    Structure Name                   = db2.db2inst1.<DBNAME>.gbp
    SID                              = 316
    Current Size (4KB)               = 560263
    Target Size (4KB)                = 6078976
    Configured Size (4KB)            = 6080000


  Shared Communication Area Information
    Structure Name                   = db2.db2inst1.<DBNAME>.sca
    SID                              = 314
    Current Size (4KB)               = 15360
    Target Size (4KB)                = 16640
    Configured Size (4KB)            = 16640


  List Structure / Smart Array Information
    Structure Name                   = db2.db2inst1.<DBNAME>.list
    SID                              = 2
    Current Size (4KB)               = 31599
    Target Size (4KB)                = 32000
    Configured Size (4KB)            = 32000


  Lock Structure Information
    Structure Name                   = db2.db2inst1.<DBNAME>.lock
    SID                              = 4
    Current Size (4KB)               = 202374
    Target Size (4KB)                = 1144320
    Configured Size (4KB)            = 1144320
```

GBP Information:

```
SELECT          VARCHAR(HOST_NAME,40) AS CF_HOSTNAME,
                VARCHAR(DB_NAME,8) AS DBNAME,
                CURRENT_CF_GBP_SIZE,
                CONFIGURED_CF_GBP_SIZE,
                TARGET_CF_GBP_SIZE
FROM TABLE( MON_GET_CF( cast(NULL as integer) ) ) AS CF_GBP_METRICS;

CF_HOSTNAME              DBNAME   CURRENT_CF_GBP_SIZE CONFIGURED_CF_GBP_SIZE TARGET_CF_GBP_SIZE
```

```
---------------------------- -------- ------------------- ---------------------- -------------------
CF1_hostname          <DBNAME>        104401          6080000        6080000
CF2_hostname          <DBNAME>        561602          6080000        6078976
```

GLM Information:

```
SELECT          VARCHAR(HOST_NAME,40) AS CF_HOSTNAME,
                VARCHAR(DB_NAME,8) AS DBNAME,
                CURRENT_CF_LOCK_SIZE,
                CONFIGURED_CF_LOCK_SIZE,
                TARGET_CF_LOCK_SIZE
FROM TABLE( MON_GET_CF( cast(NULL as integer) ) ) AS CF_GLM_METRICS;

CF_HOSTNAME           DBNAME   CURRENT_CF_LOCK_SIZE CONFIGURED_CF_LOCK_SIZE TARGET_CF_LOCK_SIZE
---------------------------- -------- ------------------- ---------------------- -------------------
CF1_hostname          <DBNAME>        194331          1144320        1144320
CF2_hostname          <DBNAME>        201794          1144320        1144320
```

SCA Information:

```
SELECT          VARCHAR(HOST_NAME,40) AS CF_HOSTNAME,
                VARCHAR(DB_NAME,8) AS DBNAME,
                CURRENT_CF_SCA_SIZE,
                CONFIGURED_CF_SCA_SIZE,
                TARGET_CF_SCA_SIZE
FROM TABLE( MON_GET_CF( cast(NULL as integer) ) ) AS CF_SCA_METRICS;

CF_HOSTNAME           DBNAME   CURRENT_CF_SCA_SIZE CONFIGURED_CF_SCA_SIZE TARGET_CF_SCA_SIZE
---------------------------- -------- ------------------- ---------------------- -------------------
CF1_hostname          <DBNAME>        47032           48640          48640
CF2_hostname          <DBNAME>        47032           48640          48640
```

CF Memory Allocation Information:

```
SELECT          VARCHAR(HOST_NAME,40) AS CF_HOSTNAME,
                VARCHAR(DB_NAME,8) AS DBNAME,
                CURRENT_CF_MEM_SIZE,
                CONFIGURED_CF_MEM_SIZE
FROM TABLE( MON_GET_CF( cast(NULL as integer) ) ) AS CF_MEM_METRICS;

CF_HOSTNAME           DBNAME   CURRENT_CF_MEM_SIZE CONFIGURED_CF_MEM_SIZE
---------------------------- -------- ------------------- ----------------------
CF1_hostname          <DBNAME>        8976128         10239232
CF2_hostname          <DBNAME>        8978176         10247936
```

CF Structure in Detail (2/4)

```
Hostname                                  = CF1_hostname
    Management Port                           = 56000
     Role                                      = Primary
    CF Host Information
       Virtual Memory Used                    = 233277587456
       Virtual Memory Available               = 68443865088
       Real Memory Used                       = 197005500416
       Real Memory Available                  = 562995200
       Swap File Used                         = 36272087040
       Swap File Available                    = 67880869888
       CPU Usage                              = 1
    Total structure memory (4KB)              = 45816832
    Free structure memory (4KB)               = 35537920
    Memory frame Size (4KB)                   = 256
    Configured Size (4KB)                     = 45816832
```

CF Structure in Detail (3/4)

```
Group Bufferpool Information
    Structure Name                      = db2.db2inst1.<DBNAME>.gbp
    SID                                 = 316
    Current Size (4KB)                  = 104335
    Target Size (4KB)                   = 6080000
    Configured Size (4KB)               = 6080000
Shared Communication Area Information
    Structure Name                      = db2.db2inst1.<DBNAME>.sca
    SID                                 = 314
    Current Size (4KB)                  = 15360
    Target Size (4KB)                   = 16640
    Configured Size (4KB)               = 16640
```

CF Structure in Detail (4/4)

```
List Structure / Smart Array Information
        Structure Name                      = db2.db2inst1.<DBNAME>.list
        SID                                 = 3
        Current Size (4KB)                  = 31599
        Target Size (4KB)                   = 32000
        Configured Size (4KB)               = 32000
    Lock Structure Information
        Structure Name                      = db2.db2inst1.<DBNAME>.lock
        SID                                 = 5
        Current Size (4KB)                  = 194623
        Target Size (4KB)                   = 1144320
        Configured Size (4KB)               = 1144320
```

Difference between Lock and Latch (1/2)

Lock	Latch
Protects transactional data consistency	Lightweight synchronization object that protects the data consistency at internal data structure in memory i.e. buffer pool
Locks are applied based on different isolation levels: • Uncommitted Read (UR) • Cursor Stability (CS) • Read Stability (RS) • Repeatable Read (RR) Lock statuses are: G (granted), W (waiting) and C (converting state)	Latches being applied at EDU (Engine Dis-patchable Unit) level and the status would generally be: H -> EDU holding a latch W -> EDU waiting on a latch C -> Latches are contested (just like lock-waits)
Locks can be monitored using the **SYSPROC.MON_GET_LOCKS ()** table function. The table function **SYSPROC.MON_GET_APPL_LOCKWAIT ()** can be used to find the information about lock for which an application is waiting.	Latches can be monitored using **SYSPROC.MON_GET_LATCH ()** table function. **SYSPROC.MON_GET_EXTENDED_LATCH_WAIT ()** table function can be used to find the latch waits between the members and the time it spends in waiting.

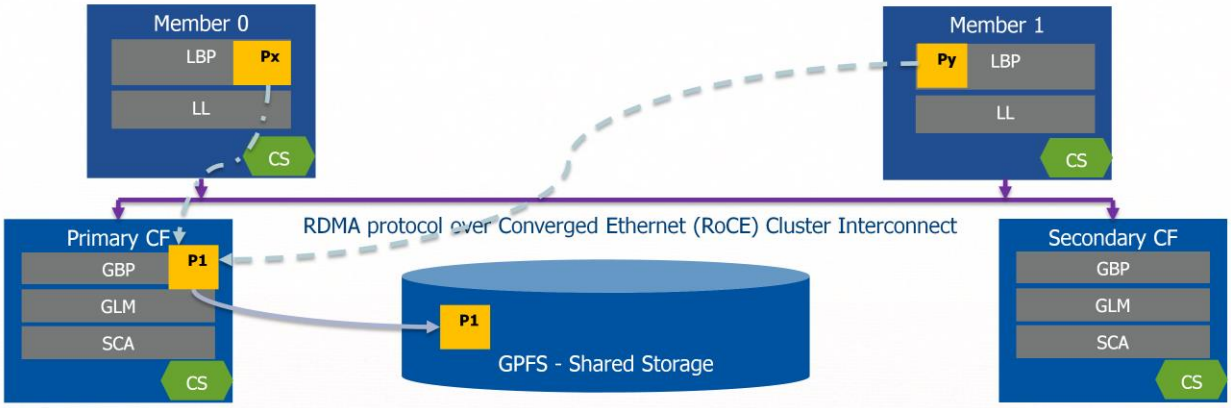
Difference between Lock and Latch (2/2)

```
SELECT  VARCHAR(LATCH_NAME,40) AS LATCH_NAME, VARCHAR(MEMORY_ADDRESS,20) AS
MEMORY_ADDRESS, EDU_ID, SUBSTR(EDU_NAME,1,20) EDU_NAME, APPLICATION_HANDLE,
MEMBER, LATCH_STATUS, LATCH_WAIT_TIME FROM TABLE ( MON_GET_LATCH( NULL, -2 ) )
ORDER BY LATCH_NAME, LATCH_STATUS

LATCH_NAME                                MEMORY_ADDRESS       EDU_ID
EDU_NAME             APPLICATION_HANDLE   MEMBER LATCH_STATUS LATCH_WAIT_TIME
---------------------------------------- -------------------- --------------------
- -------------------- -------------------- ------ ------------ -----------------
---
SQLO_LT_SQLE_KRCB__EDUChainLatch          0x000000020083F262
20 db2agent (SAMPLE) 0                      7      0 H
-
SQLO_LT_sqeWLDispatcher__m_tunerLatch     0x0000000202240470
14 db2wlmt 0                                -      0 H
-

  2 record(s) selected.
```

p-Lock in pureScale (1/2)



p-Lock in pureScale (2/2)

- Db2 pureScale uses a special page level lock to manage the changes to the buffer pool pages across multiple members in the cluster.
- This is necessary to make sure a member makes a change to the most recent copy of the data.
- CF maintains the shared and exclusive page lock information globally to the cluster
- When a member makes a change to a page in LBP and the changed page will be sent to GBP upon a commit. Then the p-lock will be released or downgraded, for example – from X to NS lock.

Lock Management in pureScale (1/2)

Local Lock Manager (LLM)

- Runs on each member and manages the lock requests made by applications just like an ESE database
- Records information about locks held by each application and transactions

Global Lock Manager (GLM)

- Runs on CF and co-ordinates the lock requests made by the LLM's
- Records information about locks held by each member

Lock Request and Negotiation

When a member needs a lock, the LLM coordinates with GLM to acquire it. There are set of CF commands and can be viewed via **MON_GET_CF_CMD ()**

- Set Lock State Command – Single lock request from LLM to GLM
- Set Lock State Multiple Command – Multiple lock requests from LLM to GLM

16

```
SELECT
VARCHAR(CF_CMD_NAME,40) AS CF_CMD_NAME,
TOTAL_CF_REQUESTS AS REQUESTS
FROM TABLE( MON_GET_CF_CMD(129) ) AS CF_REQ_METRICS;

CF_CMD_NAME                              REQUESTS
---------------------------------------- --------------------
TestPageValidity                                    0
ReadAndRegister                            38604286913
WriteAndRegister                            7191197986
WriteAndRegisterMultiple                    1718730842
ReadCastoutClass                              10315991
ReadCCInfo                                    61328184
ReadForCastout                                      0
ReadForCastoutMultipleList                          0
ReadForCastoutMultiple                        513056951
ReleaseCastoutLocks                           235757813
ReadCacheInfo                                      1202
AttachLocalCache                                    945
ReadSetLFS                                 6905487110
TryInstant                                 1018432811
SetLockStateCommands                       37346910671
RecordLockStateCommands                        8971006
ReadLocks                                         8726
ReadSA                                     51425173
```

WriteSA	40376365410
DiagPing	0
ProcessSetLockState	59438873585
WriteAndRegisterMultipleSubOperation	13848176799
CrossInvalidate	30577576465
MessageResponseBlock	58913126584
ExtendedMessageResponseBlock	22724086942
MessageResponseBlockAsync	0
ExtendedMessageResponseBlockAsync	0
LockNotification	4205777402
AllocationUnitRecovery	36681
ProcessSetLockStateSingleLock	0
ProcessSetLockStateNewLock	0
ProcessSetLockStateExistingLock	0
ProcessSetLockStateNewClient	0
ProcessSetLockStateExistingClient	0

34 record(s) selected.

Lock Management in pureScale (2/2)

```
SELECT VARCHAR(CF_CMD_NAME,40) AS CF_CMD_NAME,
TOTAL_CF_REQUESTS AS REQUESTS FROM TABLE( MON_GET_CF_CMD(129) ) AS
CF_REQ_METRICS;
CF_CMD_NAME                                REQUESTS
---------------------------------------- --------------------
ReadAndRegister                                    38604286913
WriteAndRegister                                    7191197986
WriteAndRegisterMultiple                            1718730842
SetLockStateCommands                               37346910671
RecordLockStateCommands                                8971006
ProcessSetLockState                                59438873585
WriteAndRegisterMultipleSubOperation               13848176799
CrossInvalidate                                    30577576465
```

Lock Parameters

Local Lock Manager Level

- **LOCKLIST:** This parameter indicates the amount of storage that is allocated to the lock list in 4K pages. This is allocated per member and per database in a pureScale cluster.
- **MAXLOCKS:** This parameter defines a percentage of the lock list held by an application that must be filled before the database performs lock escalation

Global Lock Manager Level

- **CF_LOCK_SZ:** This parameter determines the memory size in 4K pages used by the CF for global lock management between members.

Lock Monitoring (1/2)

Local Lock-waits

```
SELECT  VARCHAR(HOST_NAME,30) AS HOST,
        VARCHAR(DB_NAME,8) AS DBNAME,
        CURRENT_CF_LOCK_SIZE,
        CONFIGURED_CF_LOCK_SIZE,
        TARGET_CF_LOCK_SIZE
FROM TABLE( MON_GET_CF( 128 ) ) AS GLM_METRICS;
```

HOST	DBNAME	CURRENT_CF_LOCK_SIZE	CONFIGURED_CF_LOCK_SIZE	TARGET_CF_LOCK_SIZE
CF1	SAMPLE	117751	820480	820480

Lock Monitoring (2/2)

Local Lock-waits

```
SELECT DISTINCT
        B.APPLICATION_HANDLE,
        A.HLD_APPLICATION_HANDLE,
        A.LOCK_NAME, A.HLD_MEMBER,
        A.LOCK_STATUS FROM TABLE (MON_GET_APPL_LOCKWAIT(NULL, -2)) A JOIN
  TABLE (MON_GET_LOCKS(CLOB('<lock_name>'||A.LOCK_NAME||'</lock_name>'),-2)) B
  ON A.LOCK_NAME=B.LOCK_NAME
  WHERE A.HLD_APPLICATION_HANDLE IS NOT NULL;
```

GLM Lock Information

```
db2pd –db <dbname> -cfdump struct=glm
```

pureScale Lock Wait Monitoring Elements

lock_waits: The total number of times that applications waited for locks

lock_waits_global: The number of lock waits due to the application holding the lock being on a remote member

lock_timeouts: The number of times that a request to lock an object timed out instead of being granted

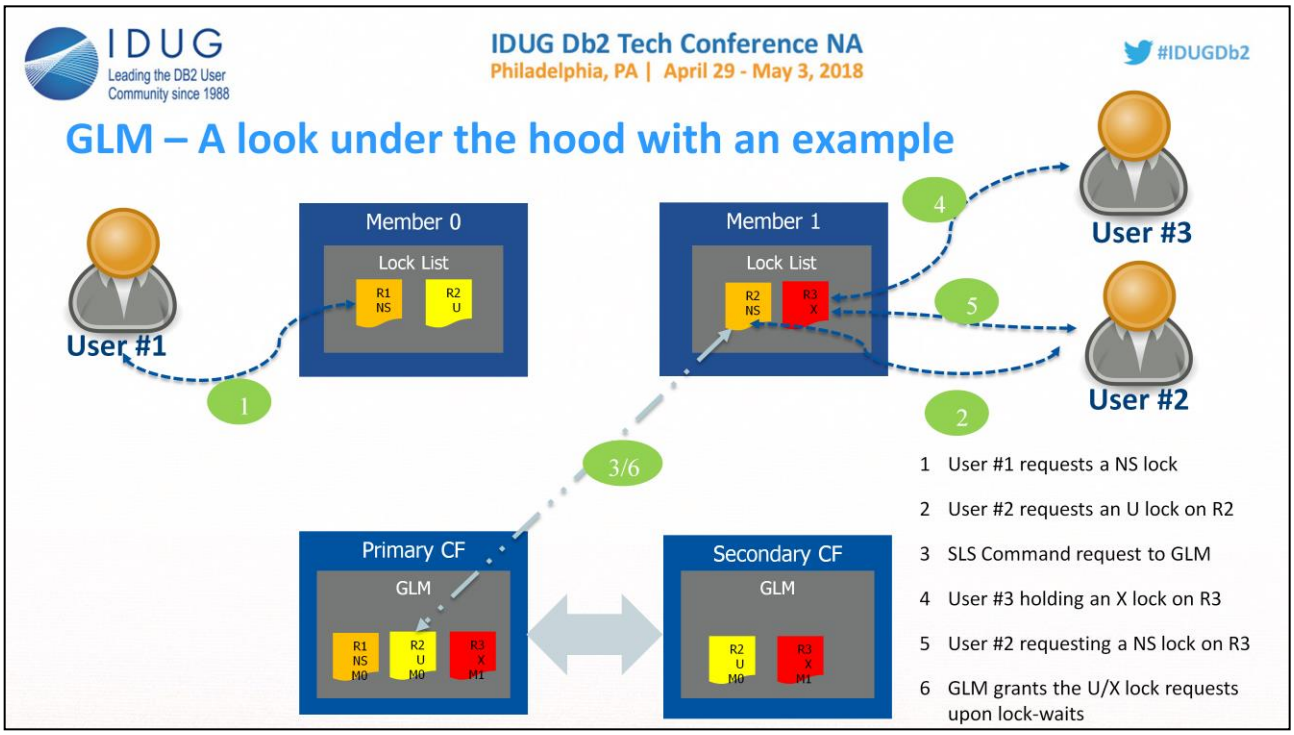
lock_timeouts_global: The number of lock timeouts where the application holding the lock was on a remote member

lock_wait_time: The total elapsed time in millisecond spent waiting for locks at local and global level

lock_wait_time_global: The total elapsed time in millisecond spent waiting for global locks

lock_wait_time_top: The high watermark for lock wait times in milliseconds of any request in a workload

lock_wait_time_global_top: The longest lock wait reported in milliseconds that has occurred for a lock that is held on another member



1. User #1 requests NS Lock on row R1 on member 0.
   - Row R1 is registered in GLM already and the lock holding member is 0,
   - No re-grant is necessary from GLM
2. User #2 requests an update lock on row R2 on member 1 however row R2 lock has been held by member 0
3. Member 1 sends a request to GLM via set lock state command and waits for the lock
   - This lock-wait is GLOBAL in nature because of multiple members participation in the lock request
4. User #3 holding an exclusive lock on row R3
5. User #2 requesting a NS lock on row R3. Since R3 lock is already been with user #2 on the same member, this will be a local lock-wait due to the fact that lock request and the holder are from the same member
6. GLM sending acknowledgement

Local and Global Lock Escalations

The lock escalation can occur both at local member level and global level.

At Local Member Level: Lock escalations are based on ***maxlocks*** setting and the ***locklist*** memory availability for a new lock request.

At Global Level: When a member requests for an additional lock reaches a threshold of GLM memory ***cf_lock_sz***. The ***maxlocks*** has no effect on the global level lock escalation.

Monitoring Local and Global Lock Escalations

```
SELECT
        MEMBER,
        LOCK_WAITS,
        LOCK_WAIT_TIME,
        LOCK_TIMEOUTS,
        LOCK_ESCALS,
        LOCK_WAITS_GLOBAL,
        LOCK_WAIT_TIME_GLOBAL,
        LOCK_TIMEOUTS_GLOBAL,
        LOCK_ESCALS_GLOBAL
FROM TABLE(MON_GET_WORKLOAD('SYSDEFAULTUSERWORKLOAD',-2))  ORDER BY MEMBER DESC;
```

MEMBER	LOCK_WAITS	LOCK_WAIT_TIME	LOCK_TIMEOUTS	LOCK_ESCALS	LOCK_WAITS_GLOBAL	LOCK_WAIT_TIME_GLOBAL	LOCK_TIMEOUTS_GLOBAL	LOCK_ESCALS_GLOBAL
0	15859	3541387	58	0	291	3526836	58	0
1	1186	6403124	102	0	915	6397849	102	0
2	1073	110667	1	0	248	88642	1	0

LOCK_WAITS: The total number of times that applications or connections waited for locks.

LOCK_WAIT_TIME: The total elapsed time spent waiting for locks. The value is given in milliseconds.

LOCK_TIMEOUTS: The number of times that a request to lock an object timed out instead of being granted.

LOCK_ESCALS: The number of times that locks have been escalated from several row locks to a table lock.

LOCK_WAITS_GLOBAL: Number of lock waits due to the application holding the lock being on a remote member.

LOCK_WAIT_TIME_GLOBAL: Time spent on global lock waits. The unit of measurement for time is in milliseconds.

LOCK_TIMEOUTS_GLOBAL: Number of lock timeouts where the application holding the lock was on a remote member.

LOCK_ESCALS_GLOBAL: Number of lock escalations on a global lock due to global lock memory usage reaching the limit specified in the ***cf_lock_sz*** database configuration parameter.

LOCK_WAIT_TIME = (Local lock-wait time + Global lock-wait time) in millisecond

GLOBAL_WAIT_TIME_GLOBAL = This is just the time spent on lock-waits between members

A Word About Memory Capacity Planning

- CF_GBP_SZ = 25%~40% of SUM of LBP's
  - 25% for heavy read workload
  - 40% for heavy write and read workload
- CF_DB_MEM_SZ = CF_GBP_SZ + 25% of (CF_GBP_SZ)
- CF_LOCK_ZS = 15% of CF_GBP_SZ
- CF_SCA_SZ = 8% of CF_GBP_SZ

CF_DB_MEM_SZ
CF_GBP_SZ
CF_LOCK_SZ
CF_SCA_SZ

If you want Db2 to manage the CF memory settings automatically, set:

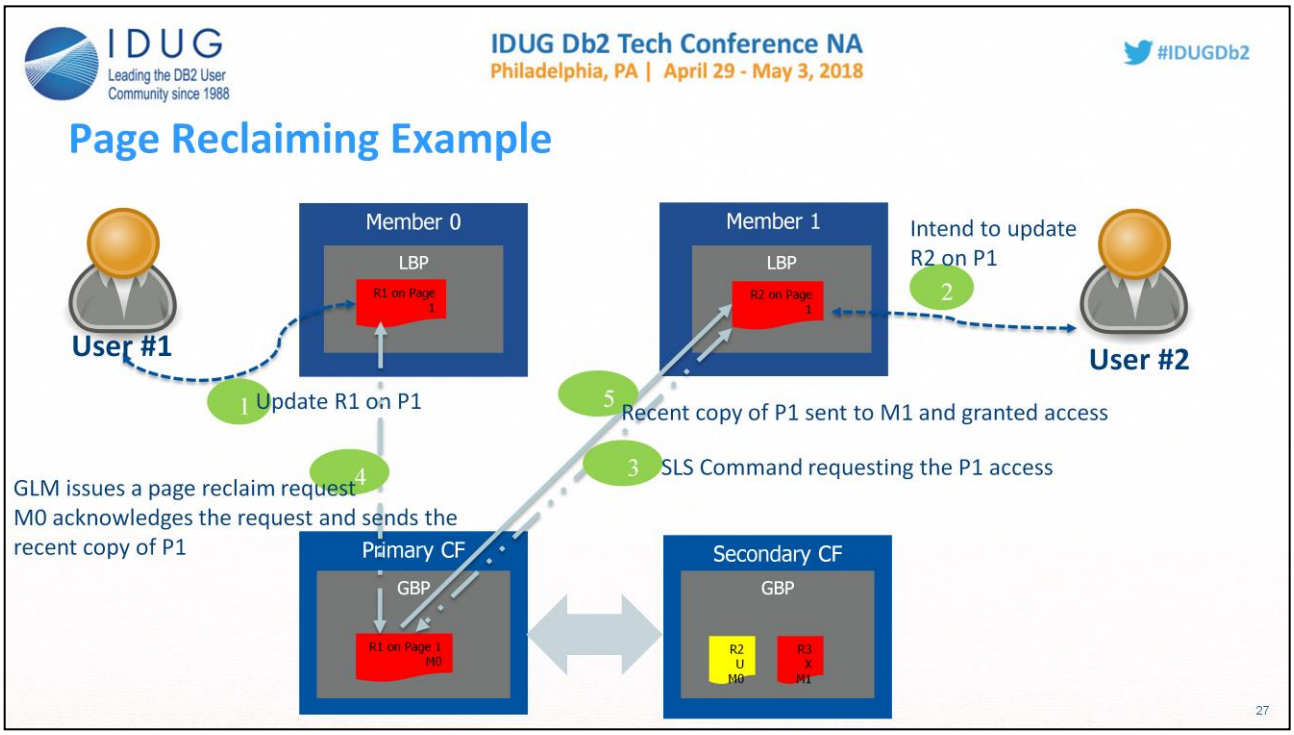
db2set **DB2_DATABASE_CF_MEMORY=AUTO**

But we will discuss the caveats in the next presentation

Page Reclaims (Negotiations)

When an application on a member needs to make a change to a page which is currently held exclusively by another member, the CF can request the p-lock is released early. This process of releasing a page early is called *page reclaiming*.

This help eliminating the page lock waits across members, but these reclaims are expensive and this can be monitored via **SYSPROC.MON_GET_PAGE_ACCESS_INFO ()** table function.



1. User #1 updates a record R1 in page P1 on member 0
2. User #2 requests an update lock on record R2 in page P1 on member 1
3. Member 1 has to initiate a GLM SLS command request due to the fact that row R2 is on exact same page where row R1 is and page P1 access has been granted to member 0
4. CF sends an early page release request to member 0 and member 0 honors the request and sends the recent copy of the page back to CF's group buffer pool (GBP)
5. CF sends the recent page P1 to member 1 and registers the handle.

Please note, during this page reclaim window, member 0 will still held on to other locks (if any).

Monitoring Page Reclaims

```
SELECT
        VARCHAR(TABNAME,30) AS NAME,
        VARCHAR(OBJTYPE,10) AS TYPE,
        PAGE_RECLAIMS_X AS PGRCX,
        PAGE_RECLAIMS_S AS PGRCS,
        SPACEMAPPAGE_PAGE_RECLAIMS_X AS SMPPGRCX,
        SPACEMAPPAGE_PAGE_RECLAIMS_S AS SMPPGRCS,
        RECLAIM_WAIT_TIME
FROM TABLE( MON_GET_PAGE_ACCESS_INFO('DEMO','EQP_CST',NULL)  ) AS RECLAIMMETRICS
ORDER BY RECLAIM_WAIT_TIME DESC;
```

TYPE	PGRCX	PGRCS	SMPPGRCX	SMPPGRCS	RECLAIM_WAIT_TIME
INDEX	1944526	1027	14112	0	16925526
TABLE	315	6	290	6	14

PAGE_RECLAIMS_X: The number of times a page related to the object was reclaimed by another member before its planned release, where the member that reclaimed the page required exclusive access.
PAGE_RECLAIMS_S: The number of times a page related to the object was reclaimed by another member before its planned release, where the member that reclaimed the page required shared access.
SPACEMAPPAGE_PAGE_RECLAIMS_X: The number of times a page related to a space map page was reclaimed by another member before its planned release. The member that reclaimed the page required exclusive access to the space map page. Generally space mappage holds the information about the free space available within the allocated extents. This space map information is important when write operation is performed.
SPACEMAPPAGE_PAGE_RECLAIMS_S: The number of times a page related to a space map page was reclaimed by another before its planned release. The member that reclaimed the page required shared access to the space map page.
RECLAIM_WAIT_TIME: This element represents the amount of time in milliseconds spent waiting on page locks, where the lock request caused a page to be reclaimed.

How to Reduce the Page Reclaims?

- Smaller page size reduces false sharing conflicts and help reduces reclaims on tables and indexes
- Increase **PCTFREE** for tiny hot tables
- Adding **CURRENT MEMBER** or **CURRENT NODE** default column to the table and partition it on period and member. For Example:

```
CREATE TABLE DEMO.EQP_CST (
   ...
   PSCL_CLSTR_MBR_NBR INTEGER NOT NULL WITH DEFAULT CURRENT NODE,
   PSCL_CLSTR_SKEY DATE NOT NULL WITH DEFAULT CURRENT DATE)
PARTITION BY RANGE(PSCL_CLSTR_MBR_NBR, PSCL_CLSTR_SKEY) (....);
```

TYPE	PGRCX	PGRCS	SMPPGRCX	SMPPGRCS	RECLAIM_WAIT_TIME
INDEX	7423	0	0	0	97109
INDEX	6414	0	0	0	81105
INDEX	6261	0	0	0	75940

Random Key Indexes

- Workloads that are inserting into an index which is ordered on a monotonically incrementing value such as timestamp, identity column, or sequence numbers, can possibly experience high index key page contention. In pureScale environment this will result in an index leaf page reclaim.
- The fix is to create a new **RANDOM** order index. These index key values will be sorted in random order instead of sorting it in strict ascending or descending order and the placement of index keys occur randomly across all of the leaf index pages, avoiding the contention.

```
CREATE INDEX DEMO.IDX1 ON DEMO.EQCMPNT_MEAS (PSCL_CLSTR_MBR_NBR, PSCL_CLSTR_SKEY
RANDOM)
```

This approach CAN'T be used on tables where you have range queries running but equality predicate query tables are the ideal candidates!

Explicit Hierarchical Locking (EHL)

- Eliminates data sharing (CF communication) cost for tables, range partitioned tables and partitioned indexes that are accessed only by a single member.

```
UPDATE DB CFG FOR <DBNAME> USING opt_direct_wrkld YES;
```

- This parameter can be set to AUTOMATIC, and when it is AUTOMATIC, EHL will be activated when both ***cf_gbp_sz*** and ***cf_lock_sz*** are set to AUTOMATIC.

EHL States

- **SHARED:** Full shared across all the members in the cluster
- **BECOMING_NOT_SHARED:** Transitioning from SHARED to NOT_SHARED
- **NOT_SHARED:** All access is limited to one member in the cluster
- **BECOMING_SHARED:** Transitioning from NOT_SHARED to SHARED

EHL State Changes

- Entering into **NOT_SHARED** state
  - An INSERT, UPDATE, DELETE and SELECT operation on a table
  - A data partition in a range partition table can enter NOT_SHARED state while the base table is unaffected
  - Index can enter this state when a non-partitioned index is created
- Exiting **NOT_SHARED** state
  - Table access from other member or members
  - Drop table or database deactivation
  - Partition ATTACH/DETACH operation on a partitioned table

Locking Behavior During NOT_SHARED to SHARED State Change

The table lock is held in super exclusive mode (Z) until all the page locks and row locks are registered in Global Lock Manager (GLM) and all the changed pages are written to Group Buffer pool (GBP).

Monitoring Elements:

data_sharing_state – We have already discussed

data_sharing_state_change_time – Timestamp of the last state change

data_sharing_remote_lockwait_count – Number of times application waited on a table transition from NOT_SHARED to SHARED

data_sharing_remote_lockwait_time – Application wait time in milliseconds on a table during the transition from NOT_SHARED to SHARED

EHL Monitoring Table Functions

- SYSPROC.MON_GET_TABLE ()
- SYSPROC.MON_GET_DATABASE()
- SYSPROC.MON_GET_APPL_LOCKWAIT()

```
SELECT   VARCHAR(TABNAME,40) AS TABNAME,
         MEMBER,
         DATA_SHARING_STATE AS DS_STATE,
         DATA_SHARING_STATE_CHANGE_TIME AS DSC_TIME,
         DATA_SHARING_REMOTE_LOCKWAIT_COUNT AS DSRL_COUNT,
         DATA_SHARING_REMOTE_LOCKWAIT_TIME AS DSRL_TIME_MS
FROM
TABLE(MON_GET_TABLE('DEMO','EQCMPNT_MEAS',-2)) ORDER BY MEMBER ASC;
```

TABNAME	MEMBER	DS_STATE	DSC_TIME	DSRL_COUNT	DSRL_TIME_MS
EQCMPNT_MEAS	0	SHARED	2018-04-13-15.44.33.794552	5	485
EQCMPNT_MEAS	1	SHARED	2018-04-15-00.05.42.018350	12	865
EQCMPNT_MEAS	2	NOT_SHARED	2018-04-15-18.14.56.995229	68	6434

Transaction Logging in pureScale

- Active transaction logs must always be on GPFS
- One log stream per member

```
32768 Apr 17 14:51 LOGSTREAM0004/
32768 Apr 17 14:53 LOGSTREAM0002/
32768 Apr 17 14:54 LOGSTREAM0000/
32768 Apr 17 15:01 LOGSTREAM0003/
32768 Apr 17 15:01 LOGSTREAM0001/
```

- No additional configuration parameters for pureScale logging

LOGBUFSZ	LOGFILSIZ	LOGPRIMARY
LOGSECOND	NEWLOGPATH	OVERFLOWLOGPATH
MIRRORLOGPATH	BLK_LOG_DSK_FUL	BLOCKNONLOGGED
MAX_LOG	NUM_LOG_SPAN	PAGE_AGE_TRGT_MCR
PAGE_AGE_TRGT_GCR	LOGARCHMETH1	FAILARCHPATH
LOGARCHCOMPR1	SOFTMAX (deprecated)	LOGINDEXBUILD

LOGBUFSZ: Amount of the database heap to use as a buffer for log records before writing these records to disk

LOGFILSIZ: Defines the size of each primary and secondary log file

LOGPRIMARY: Specifies the number of primary log files to be pre-allocated

LOGSECOND: Specifies the number of secondary log files that are created and used for recovery log files

NEWLOGPATH: Specifies the location for active log files

OVERFLOWLOGPATH: Specifies a location to find log files needed for a roll-forward operation, as well as where to store active log files retrieved from the archive

MIRRORLOGPATH: Specifies a path to store the mirror logs

BLK_LOG_DSK_FUL: Prevents disk full errors from being generated when database cannot create a new log file in the active log path

BLOCKNONLOGGED: Specifies whether the database manager will allow tables to have the NOT LOGGED or NOT LOGGED INITIALLY attributes activated

MAX_LOG: Specifies if there is a limit to the percentage of the primary log space that a transaction can consume, and what that limit is

NUM_LOG_SPAN: Specifies whether there is a limit to how many log files one transaction can span, and what that limit is

PAGE_AGE_TRGT_MCR: Specifies the target duration (in seconds) for changed pages to be kept in the local buffer pool before they are persisted to table space storage, or to the group buffer pool

PAGE_AGE_TRGT_GCR: This configuration parameter specifies the target duration (in seconds) for changed pages to be kept in the group buffer pool before the pages are persisted to table space storage

LOGARCHMETH1: Specify the media type of the primary destination for logs that are archived from the current log path

FAILARCHPATH: Specifies a path to which database will try to archive log files if the log files cannot be archived to either the primary or the secondary (if set) archive destinations because of a media problem affecting those destination

LOGARCHCOMPR1: Specifies whether the log files written to the primary archive destination for logs are compressed

Log Sequence Number (LSN) in pureScale

- Determine the order of the operations that generated the log records
- The LSN numbers are unique within a log stream, however it is not unique across the log streams (across the members)
- There will be a LNS sync whenever you create a table or table space object. For example:

```
Member 00x00000007D095F6B6
Member 10x00000007D095F6B2
Member 20x00000007D095F6A6
CREATE TABLE DEMO.SAMPLE (ID INT);  on member 0
Member 00x00000007D095F748
Member 10x00000007D095F74E
Member 20x00000007D095F750
```

db2pd –db <dbname> -logs

- The LSN on rest of the members will actually sync (greater LSN number than that of the table creation member LSN) with the table creation member LSN
- Whenever a page is changed, a respective member's LSN for the log record is placed into the page header

On a 3 member pureScale cluster:

Member 0:

```
Logs:
Current Log Number          21149
Pages Written               20069
Cur Commit Disk Log Reads   0
Cur Commit Total Log Reads  1720
Method 1 Archive Status     Success
Method 1 Next Log to Archive  21149
Method 1 First Failure      n/a
Method 2 Archive Status     n/a
Method 2 Next Log to Archive  n/a
Method 2 First Failure      n/a
Log Chain ID                4
Current LSO                 2832035376797
Current LSN                 0x00000007D095F6B6

Address          StartLSN        StartLSO          State     Size     Pages     Filename
0x0A00030096884CD8 00000007D0868D38 2831953572833      0x00000000 32768     32768     S0021149.LOG
0x0A00030097912718 0000000000000000 2832087135201      0x00000000 32768     32768     S0021150.LOG
0x0A000300979899B8 0000000000000000 2832220697569      0x00000000 32768     32768     S0021151.LOG
0x0A00030099D60658 0000000000000000 2832354259937      0x00000000 32768     32768     S0021152.LOG
0x0A0003008D2181D8 0000000000000000 2832487822305      0x00000000 32768     32768     S0021153.LOG
```

0x0A000300962C7638 0000000000000000 2832621384673 0x00000000 32768 32768 S0021154.LOG
0x0A0003008CEEC958 0000000000000000 2832754947041 0x00000000 32768 32768 S0021155.LOG
0x0A0003008D21C8B8 0000000000000000 2832888509409 0x00000000 32768 32768 S0021156.LOG
0x0A000300978F3358 0000000000000000 2833022071777 0x00000000 32768 32768 S0021157.LOG
0x0A000300951AE9F8 0000000000000000 2833155634145 0x00000000 32768 32768 S0021158.LOG
0x0A00030096885858 0000000000000000 2833289196513 0x00000000 32768 32768 S0021159.LOG
0x0A00030094A46838 0000000000000000 2833422758881 0x00000000 32768 32768 S0021160.LOG
0x0A0003008D299558 0000000000000000 2833556321249 0x00000000 32768 32768 S0021161.LOG
0x0A0003008D0055F8 0000000000000000 2833689883617 0x00000000 32768 32768 S0021162.LOG
0x0A000300962EC658 0000000000000000 2833823445985 0x00000000 32768 32768 S0021163.LOG
0x0A000300968057B8 0000000000000000 2833957008353 0x00000000 32768 32768 S0021164.LOG
0x0A00030097522918 0000000000000000 2834090570721 0x00000000 32768 32768 S0021165.LOG
0x0A0003008D1287D8 0000000000000000 2834224133089 0x00000000 32768 32768 S0021166.LOG
0x0A00030094DBF458 0000000000000000 2834357695457 0x00000000 32768 32768 S0021167.LOG
0x0A0003008CE88BB8 0000000000000000 2834491257825 0x00000000 32768 32768 S0021168.LOG

Member 1:

Logs:
Current Log Number 21567
Pages Written 28945
Cur Commit Disk Log Reads 0
Cur Commit Total Log Reads 837
Method 1 Archive Status Success
Method 1 Next Log to Archive 21567
Method 1 First Failure n/a
Method 2 Archive Status n/a
Method 2 Next Log to Archive n/a
Method 2 First Failure n/a
Log Chain ID 4
Current LSO 2880710559129
Current LSN 0x00000007D095F6B2

Address StartLSN StartLSO State Size Pages Filename
0x0A0003006BA02398 00000007D01DA8EE 2880592578657 0x00000000 32768 32768 S0021567.LOG
0x0A000300031749D8 0000000000000000 2880726141025 0x00000000 32768 32768 S0021568.LOG
0x0A0003007A682458 0000000000000000 2880859703393 0x00000000 32768 32768 S0021569.LOG
0x0A000300766B46F8 0000000000000000 2880993265761 0x00000000 32768 32768 S0021570.LOG
0x0A000300766C4658 0000000000000000 2881126828129 0x00000000 32768 32768 S0021571.LOG
0x0A0003007E733718 0000000000000000 2881260390497 0x00000000 32768 32768 S0021572.LOG
0x0A0003007B5ECB58 0000000000000000 2881393952865 0x00000000 32768 32768 S0021573.LOG
0x0A0003007B544B98 0000000000000000 2881527515233 0x00000000 32768 32768 S0021574.LOG
0x0A00030077BA07B8 0000000000000000 2881661077601 0x00000000 32768 32768 S0021575.LOG
0x0A00030072AF79D8 0000000000000000 2881794639969 0x00000000 32768 32768 S0021576.LOG
0x0A0003007DEE3A38 0000000000000000 2881928202337 0x00000000 32768 32768 S0021577.LOG
0x0A00030085DD35B8 0000000000000000 2882061764705 0x00000000 32768 32768 S0021578.LOG
0x0A0003007B20F878 0000000000000000 2882195327073 0x00000000 32768 32768 S0021579.LOG

0x0A0003007E46AD98	0000000000000000	2882328889441	0x00000000	32768	32768	S0021580.LOG
0x0A0003007E141978	0000000000000000	2882462451809	0x00000000	32768	32768	S0021581.LOG
0x0A0003007A8CBA98	0000000000000000	2882596014177	0x00000000	32768	32768	S0021582.LOG
0x0A00030085DD8D98	0000000000000000	2882729576545	0x00000000	32768	32768	S0021583.LOG
0x0A0003008A3EED78	0000000000000000	2882863138913	0x00000000	32768	32768	S0021584.LOG
0x0A00030085E23F98	0000000000000000	2882996701281	0x00000000	32768	32768	S0021585.LOG
0x0A00030075574F78	0000000000000000	2883130263649	0x00000000	32768	32768	S0021586.LOG

Member 2:

Logs:

Current Log Number 1040
Pages Written 26603
Cur Commit Disk Log Reads 12
Cur Commit Total Log Reads 1477
Method 1 Archive Status Success
Method 1 Next Log to Archive 1040
Method 1 First Failure n/a
Method 2 Archive Status n/a
Method 2 Next Log to Archive n/a
Method 2 First Failure n/a
Log Chain ID 4
Current LSO 139013299248
Current LSN 0x00000007D095F6A6

Address	StartLSN	StartLSO	State	Size	Pages	Filename
0x0A0003008C1F1058	00000007D04EEB87	138904862721	0x00000000	32768	32768	S0001040.LOG
0x0A0003008BD7D598	0000000000000000	139038425089	0x00000000	32768	32768	S0001041.LOG
0x0A0003008B76DE18	0000000000000000	139171987457	0x00000000	32768	32768	S0001042.LOG
0x0A0003008BBA66D8	0000000000000000	139305549825	0x00000000	32768	32768	S0001043.LOG
0x0A00030093885258	0000000000000000	139439112193	0x00000000	32768	32768	S0001044.LOG
0x0A0003008BD18AD8	0000000000000000	139572674561	0x00000000	32768	32768	S0001045.LOG
0x0A0003009501A9B8	0000000000000000	139706236929	0x00000000	32768	32768	S0001046.LOG
0x0A00030093E295F8	0000000000000000	139839799297	0x00000000	32768	32768	S0001047.LOG
0x0A000300944F6378	0000000000000000	139973361665	0x00000000	32768	32768	S0001048.LOG
0x0A00030094857358	0000000000000000	140106924033	0x00000000	32768	32768	S0001049.LOG
0x0A00030094C34058	0000000000000000	140240486401	0x00000000	32768	32768	S0001050.LOG
0x0A0003009482BF58	0000000000000000	140374048769	0x00000000	32768	32768	S0001051.LOG
0x0A000300944E2ED8	0000000000000000	140507611137	0x00000000	32768	32768	S0001052.LOG
0x0A00030094CE7B58	0000000000000000	140641173505	0x00000000	32768	32768	S0001053.LOG
0x0A000300942D68D8	0000000000000000	140774735873	0x00000000	32768	32768	S0001054.LOG
0x0A000300939901B8	0000000000000000	140908298241	0x00000000	32768	32768	S0001055.LOG
0x0A0003008BB2F2D8	0000000000000000	141041860609	0x00000000	32768	32768	S0001056.LOG
0x0A0003008BEAF4B8	0000000000000000	141175422977	0x00000000	32768	32768	S0001057.LOG
0x0A0003008C226B18	0000000000000000	141308985345	0x00000000	32768	32768	S0001058.LOG
0x0A0003008BF59B38	0000000000000000	141442547713	0x00000000	32768	32768	S0001059.LOG

On Member 0:

CREATE TABLE DEMO.SAMPLE (ID INT);

Member 0:

Logs:
Current Log Number 21149
Pages Written 20072
Cur Commit Disk Log Reads 0
Cur Commit Total Log Reads 1720
Method 1 Archive Status Success
Method 1 Next Log to Archive 21149
Method 1 First Failure n/a
Method 2 Archive Status n/a
Method 2 Next Log to Archive n/a
Method 2 First Failure n/a
Log Chain ID 4
Current LSO 2832035389497
Current LSN 0x00000007D095F748

Address	StartLSN	StartLSO	State	Size	Pages	Filename
0x0A00030096884CD8	00000007D0868D38	2831953572833	0x00000000	32768	32768	S0021149.LOG
0x0A00030097912718	0000000000000000	2832087135201	0x00000000	32768	32768	S0021150.LOG
0x0A000300979899B8	0000000000000000	2832220697569	0x00000000	32768	32768	S0021151.LOG
0x0A00030099D60658	0000000000000000	2832354259937	0x00000000	32768	32768	S0021152.LOG
0x0A0003008D2181D8	0000000000000000	2832487822305	0x00000000	32768	32768	S0021153.LOG
0x0A000300962C7638	0000000000000000	2832621384673	0x00000000	32768	32768	S0021154.LOG
0x0A0003008CEEC958	0000000000000000	2832754947041	0x00000000	32768	32768	S0021155.LOG
0x0A0003008D21C8B8	0000000000000000	2832888509409	0x00000000	32768	32768	S0021156.LOG
0x0A000300978F3358	0000000000000000	2833022071777	0x00000000	32768	32768	S0021157.LOG
0x0A000300951AE9F8	0000000000000000	2833155634145	0x00000000	32768	32768	S0021158.LOG
0x0A00030096885858	0000000000000000	2833289196513	0x00000000	32768	32768	S0021159.LOG
0x0A00030094A46838	0000000000000000	2833422758881	0x00000000	32768	32768	S0021160.LOG
0x0A0003008D299558	0000000000000000	2833556321249	0x00000000	32768	32768	S0021161.LOG
0x0A0003008D0055F8	0000000000000000	2833689883617	0x00000000	32768	32768	S0021162.LOG
0x0A000300962EC658	0000000000000000	2833823445985	0x00000000	32768	32768	S0021163.LOG
0x0A000300968057B8	0000000000000000	2833957008353	0x00000000	32768	32768	S0021164.LOG
0x0A00030097522918	0000000000000000	2834090570721	0x00000000	32768	32768	S0021165.LOG
0x0A0003008D1287D8	0000000000000000	2834224133089	0x00000000	32768	32768	S0021166.LOG
0x0A00030094DBF458	0000000000000000	2834357695457	0x00000000	32768	32768	S0021167.LOG
0x0A0003008CE88BB8	0000000000000000	2834491257825	0x00000000	32768	32768	S0021168.LOG

Member 1:

Logs:

Current Log Number 21567
Pages Written 28945
Cur Commit Disk Log Reads 0
Cur Commit Total Log Reads 837
Method 1 Archive Status Success
Method 1 Next Log to Archive 21567
Method 1 First Failure n/a
Method 2 Archive Status n/a
Method 2 Next Log to Archive n/a
Method 2 First Failure n/a
Log Chain ID 4
Current LSO 2880710559500
Current LSN 0x00000007D095F74E

Address	StartLSN	StartLSO	State	Size	Pages	Filename
0x0A0003006BA02398	00000007D01DA8EE	2880592578657	0x00000000	32768	32768	S0021567.LOG
0x0A000300031749D8	0000000000000000	2880726141025	0x00000000	32768	32768	S0021568.LOG
0x0A0003007A682458	0000000000000000	2880859703393	0x00000000	32768	32768	S0021569.LOG
0x0A000300766B46F8	0000000000000000	2880993265761	0x00000000	32768	32768	S0021570.LOG
0x0A000300766C4658	0000000000000000	2881126828129	0x00000000	32768	32768	S0021571.LOG
0x0A0003007E733718	0000000000000000	2881260390497	0x00000000	32768	32768	S0021572.LOG
0x0A0003007B5ECB58	0000000000000000	2881393952865	0x00000000	32768	32768	S0021573.LOG
0x0A0003007B544B98	0000000000000000	2881527515233	0x00000000	32768	32768	S0021574.LOG
0x0A00030077BA07B8	0000000000000000	2881661077601	0x00000000	32768	32768	S0021575.LOG
0x0A00030072AF79D8	0000000000000000	2881794639969	0x00000000	32768	32768	S0021576.LOG
0x0A0003007DEE3A38	0000000000000000	2881928202337	0x00000000	32768	32768	S0021577.LOG
0x0A00030085DD35B8	0000000000000000	2882061764705	0x00000000	32768	32768	S0021578.LOG
0x0A0003007B20F878	0000000000000000	2882195327073	0x00000000	32768	32768	S0021579.LOG
0x0A0003007E46AD98	0000000000000000	2882328889441	0x00000000	32768	32768	S0021580.LOG
0x0A0003007E141978	0000000000000000	2882462451809	0x00000000	32768	32768	S0021581.LOG
0x0A0003007A8CBA98	0000000000000000	2882596014177	0x00000000	32768	32768	S0021582.LOG
0x0A00030085DD8D98	0000000000000000	2882729576545	0x00000000	32768	32768	S0021583.LOG
0x0A0003008A3EED78	0000000000000000	2882863138913	0x00000000	32768	32768	S0021584.LOG
0x0A00030085E23F98	0000000000000000	2882996701281	0x00000000	32768	32768	S0021585.LOG
0x0A00030075574F78	0000000000000000	2883130263649	0x00000000	32768	32768	S0021586.LOG

Member 2:

Logs:
Current Log Number 1040
Pages Written 26603
Cur Commit Disk Log Reads 12
Cur Commit Total Log Reads 1477
Method 1 Archive Status Success
Method 1 Next Log to Archive 1040
Method 1 First Failure n/a

Method 2 Archive Status n/a
Method 2 Next Log to Archive n/a
Method 2 First Failure n/a
Log Chain ID 4
Current LSO 139013300591
Current LSN 0x00000007D095F750

Address	StartLSN	StartLSO	State	Size	Pages	Filename
0x0A0003008C1F1058	00000007D04EEB87	138904862721	0x00000000	32768	32768	S0001040.LOG
0x0A0003008BD7D598	0000000000000000	139038425089	0x00000000	32768	32768	S0001041.LOG
0x0A0003008B76DE18	0000000000000000	139171987457	0x00000000	32768	32768	S0001042.LOG
0x0A0003008BBA66D8	0000000000000000	139305549825	0x00000000	32768	32768	S0001043.LOG
0x0A00030093885258	0000000000000000	139439112193	0x00000000	32768	32768	S0001044.LOG
0x0A0003008BD18AD8	0000000000000000	139572674561	0x00000000	32768	32768	S0001045.LOG
0x0A0003009501A9B8	0000000000000000	139706236929	0x00000000	32768	32768	S0001046.LOG
0x0A00030093E295F8	0000000000000000	139839799297	0x00000000	32768	32768	S0001047.LOG
0x0A000300944F6378	0000000000000000	139973361665	0x00000000	32768	32768	S0001048.LOG
0x0A00030094857358	0000000000000000	140106924033	0x00000000	32768	32768	S0001049.LOG
0x0A00030094C34058	0000000000000000	140240486401	0x00000000	32768	32768	S0001050.LOG
0x0A0003009482BF58	0000000000000000	140374048769	0x00000000	32768	32768	S0001051.LOG
0x0A000300944E2ED8	0000000000000000	140507611137	0x00000000	32768	32768	S0001052.LOG
0x0A00030094CE7B58	0000000000000000	140641173505	0x00000000	32768	32768	S0001053.LOG
0x0A000300942D68D8	0000000000000000	140774735873	0x00000000	32768	32768	S0001054.LOG
0x0A000300939901B8	0000000000000000	140908298241	0x00000000	32768	32768	S0001055.LOG
0x0A0003008BB2F2D8	0000000000000000	141041860609	0x00000000	32768	32768	S0001056.LOG
0x0A0003008BEAF4B8	0000000000000000	141175422977	0x00000000	32768	32768	S0001057.LOG
0x0A0003008C226B18	0000000000000000	141308985345	0x00000000	32768	32768	S0001058.LOG
0x0A0003008BF59B38	0000000000000000	141442547713	0x00000000	32768	32768	S0001059.LOG

LSN Distribution Across Multiple Members

Data Modification Description	Member 0 LSN	Member 1 LSN	Page 1 Header LSN	Page 2 Header LSN
Initial Values	90	80	70	60
Member 0 inserts a new row into Page 1	91	80	90	60
Member 1 inserts a new row into Page 2	91	81	90	80
Member 0 inserts another row into Page 1	92	80	91	80
Member 1 inserts another row into Page 2	92	82	91	81
Member 1 reads Page 1 from GBP	92	92	91	81
Member 1 inserts another row into Page 1	92	93	92	81

1. When member 0 inserts a new record into page 1, the LSN for page 1 is set to 90 and LSN for member 0 becomes 91 (increment member LSN by 1)
2. When member 1 inserts a new record into page 2, the LSN for page 2 is set to 80 and LSN for member 1 becomes 81 (increment member LSN by 1)
3. When member 0 inserts another record into page 1, the LSN for page 1 is set to 91 and LSN for member 0 becomes 92 (increment member LSN by 1)
4. When member 1 inserts a new record into page 2, the LSN for page 2 is set to 81 and LSN for member 1 becomes 82 (increment member LSN by 1)
5. When member 1 reads page 1 (which was inserted by member 0) from GBP, it detects page 1 LSN (91) and increases the member 1 LSN to 92
6. When member 1 inserts a records into page 1, the LSN for page 1 is set to 92 and member 1 LSN becomes 93

Log Flush Sequence (LFS) Numbers (1/2)

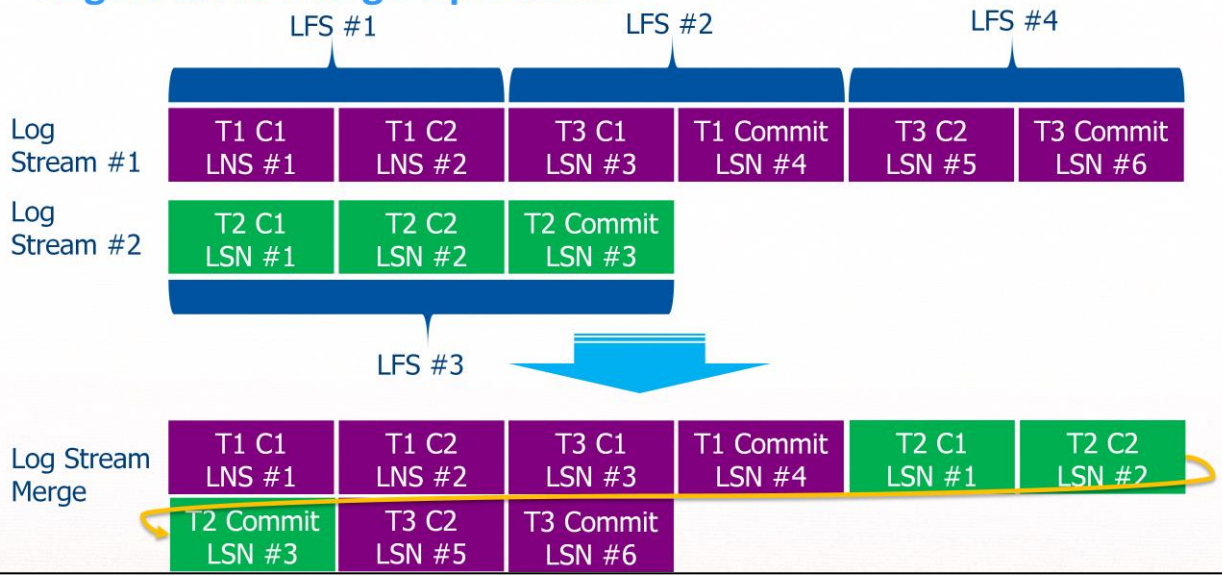
- Log Record Identifier (LRI) is used to uniquely identify log records across log streams for a database
- The LFS allows multiple log streams to be merged with an accurate time sequence
- The CF maintains a global LFS value and whenever log records are flushed to disk, respective member sends a request to the CF to get a next LFS value
- The CF increments its LFS counter and returns a value to the member
- The member places the LFS value into all of the log records being flushed
- You can monitor the number of requests sent to the CF via **SYSPROC. MON_GET_CF_WAIT_TIME ()** table function

Log Flush Sequence (LFS) Numbers (2/2)

```
SELECT,
ID,
TOTAL_CF_WAIT_TIME_MICRO/1000 AS WAIT_TIME_MS,
TOTAL_CF_REQUESTS AS CF_REQUESTS,
VARCHAR(CF_CMD_NAME,20) AS  CF_CMD_NAME
FROM TABLE (MON_GET_CF_WAIT_TIME( -1 ))
WHERE CF_CMD_NAME like '%LSN' or CF_CMD_NAME like '%LFS' ;
```

ID	TOTAL_CF_REQUESTS	TOTAL_CF_WAIT_TIME_MICRO	CF_CMD_NAME
128	0	0	GetAndIncLFS
128	40	258	GetLFS
128	0	0	GetLSN
128	455964	55200267	SetLFS
128	2334604	51309167	SetLSN
129	111948834	4327088481	GetAndIncLFS
129	797126	84115838	GetLFS
129	5667581	307508881	GetLSN
129	13	0	SetLFS
129	2351565	58576498	SetLSN

Log Streams Merge Operation



Log Record Identifier (LRI)

db2pd –db <dbname> -recovery

```
Recovery:
Recovery Status      0x41180881
Current Log          S0105550.LOG
Current LSN          000000C39619447B
Current LRI          0000000000000001000000712E115AA000000C39619447B
Current LSO          56390357315934
Job Type             ROLLFORWARD RECOVERY
Job ID               7
Job Start Time       (1522957045) Thu Apr  5 14:37:25 2018
Job Description      Database Rollforward Recovery
Invoker Type         User
Total Phases         2
Current Phase        1

Progress:
Address            PhaseNum   Description          StartTime           CompletedWork               TotalWork
0x078000000839F368 1          Forward              Thu Apr  5 14:37:25 7079648408833 bytes         Unknown
0x078000000839F4F0 2          Backward             NotStarted          0 bytes                     Unknown
```

Member Crash Recovery (MCR)

- If a member fails and has a working primary CF server, the recovery is performed on the failed member or on the restart light server using the failed member log stream
- GLM holds locks for in-flight transactions until the completion of MCR and are released at the end of the MCR
- Database will continue to service the applications except a few changes which are in inflight state

Group Crash Recovery (GCR)

A group crash recovery or a group restart is needed when:

- Only one CF and it fails
- Both the CF's fail
- Primary CF fails and the secondary CF has not reached the PEER state

During the GCR process, the logs are merged and replayed. Here only one member is used to perform the recovery.

Crash Recovery Monitoring

Both MCR and GCR's can be monitored using **LIST UTILITIES SHOW DETAIL** command or **db2pd –util** command. The command output will display the recovery type in "Description" section.

```
ID                    1
Type                  GROUP CRASH RECOVERY
Database Name         SAMPLE
Member Number         0
Description           Group Crash Recovery
```

Database Roll-forward Recovery

- ROLLFORWARD DATABASE command can be executed from any member. The member which is running the recovery will merge the logs from multiple streams and performs the recovery
- The Db2 pureScale can roll-forward the logs to END OF LOGS, any point in time or to END OF BACKUP. The point in time recovery operation stops when it encounters the first log record from any of the log streams whose timestamp is greater than the specified time stamp

A Word About GBP

- This keeps track of buffered pages in each LBP using a transaction page list off the transaction control block
- The Force-At-Commit protocol forces members to send the updated pages from LBP to GBP at commit. This information is used to synchronously invalidate any copies of such pages on other members.

A Word About Castout Operation

- Process of writing dirty pages from GBP into disk
- Minimizes the member requests to CF and GBP needing to wait for memory resources
- Limits the number of updates required during the group crash recovery
- What triggers the castout?
  The page age target group crash recovery configuration parameter (*page_age_trgt_gcr*) initiates a castout. This parameter specifies the target duration in seconds for changed pages to be kept in GBP before the pages are written to the disk.
  Recommended Setting: *page_age_trgt_gcr* to 240 seconds

Reference: Db2 v11.1 Knowledge Center:
https://www.ibm.com/support/knowledgecenter/en/SSEPGG_11.1.0/com.ibm.db2.luw.welcome.doc/doc/welcome.html

Mohankumar (Mohan) Saraswatipura works as a Database Solutions Architect at Kronsys, Inc., focusing on IBM Db2, Linux, UNIX,and Windows solutions on ESE, pureScale and BLU. Prior to his current position, he worked as a Database Solutions Architect at Reckitt Benckiser Group, plc (UK) focusing on IBM Smart Analytics System 5600, Siebel, SQL Server, and SAP HANA solutions. He is an IBM Champion (2010–2018) and a DB2's Got Talent 2013 winner. Mohan has written dozens of technical papers for IBM developerWorks and IBM Data Magazine. He is an IBM Certified DB2 Advanced Database Administrator, DB2 Application Developer, and DB2 Problem Determination Master. Mohan holds aMaster's of Technology (M Tech) degree in computer science and an Executive MBA (IT).