PD and Best practices for HADR Performance

Hao Yan, IBM

Db2 LUW

Tue, Sept 19, 2023 (10:00 AM -10:50 PM Session LUW-07)

Agenda

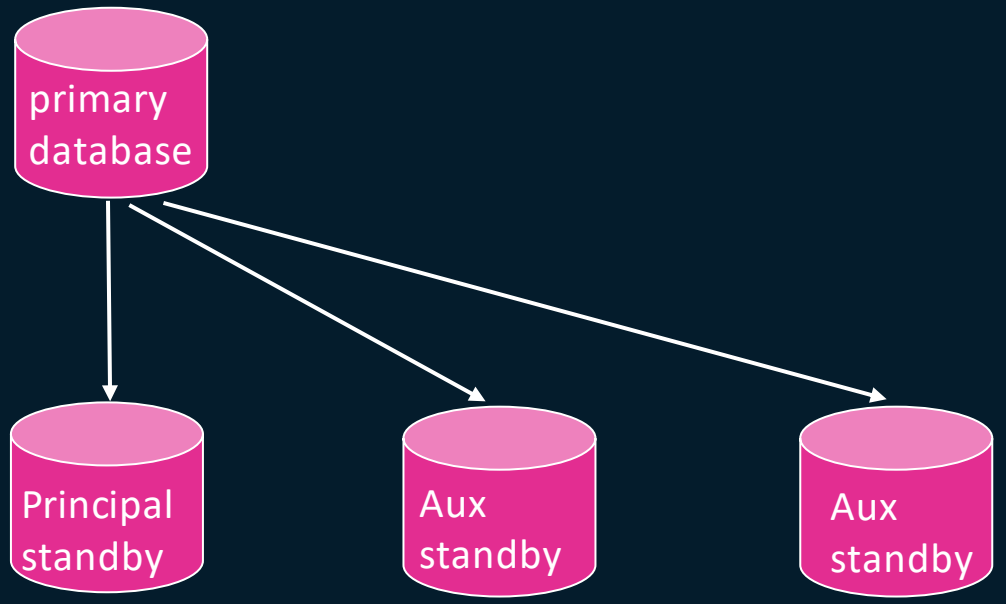
- Monitoring HADR
  - Persistent data collection
- Problem determination examples
- HADR Performance Tuning
  - Tools
  - HADR configs
  - Reads on Standby considerations
  - Takeover considerations
- HADR Best Practices

Monitoring HADR – Persistent Data Collection

- Monitoring interfaces
  - db2pd -hadr
  - Table function MON_GET_HADR
- Monitor the health of HADR system, every 60s, persistently collect
  - HADR Role, State, Status, Flags
  - Primary and Standby Log positions, calculate running averages of logging speed and replaying speed
- Save db2diag.log from all hosts

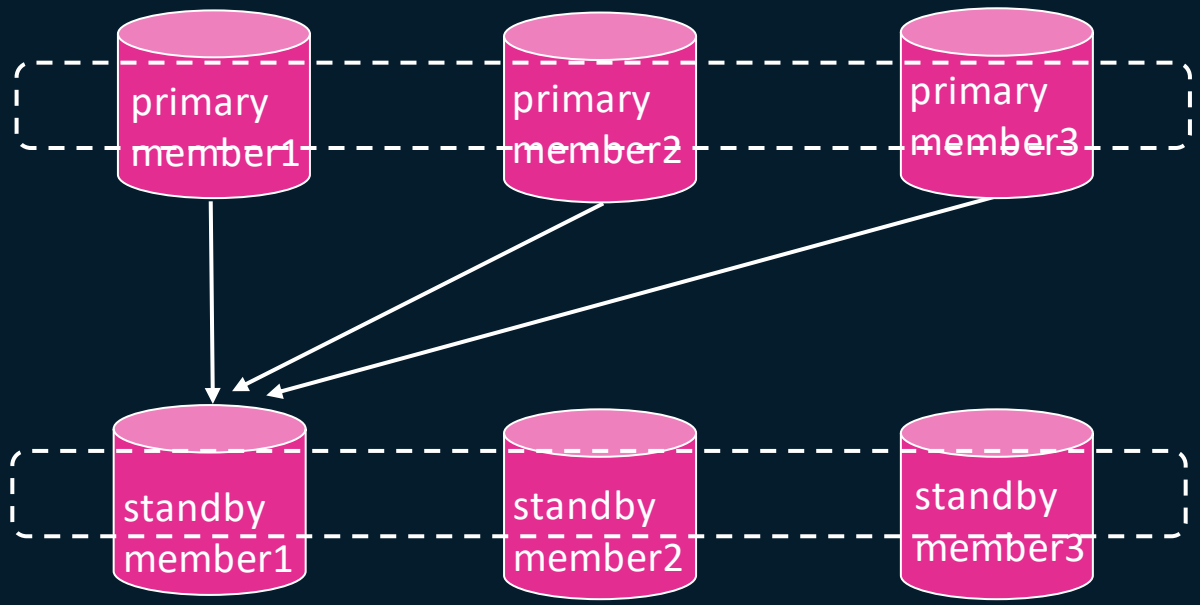
Monitoring HADR – Multiple Standby Considerations

- Treat as multiple primary-standby pairs
  - Each standby only reports on itself and the primary
  - The primary reports on itself and all standbys
- Watch out for
  - Network bottleneck on primary
  - Logging device/archive bottleneck on primary - Remote catchup reads from log device/archive



Monitoring HADR – PureScale Considerations

- Treat as multiple primary-standby pairs
- Watch out for
  - Network bottleneck on standby replay member
  - Standby replay speed



Monitoring HADR – Questions?

PD – Standby Blocked in PEER

- Has STANDBY_RECV_BLOCKED Flag
- Standby is unable to receive more logs because buffer/spool are full.
- Calculate the recent logging speed on primary and replaying speed on standby. Compare the results to the average speeds and determine the bottleneck.
  - In case of logging/shipping rate spike, adjust HADR configs to absorb the spike via bigger buffer/spool
  - In case of temporary slow replay, study workload
  - In case of consistent problem, use HADR tools to do a more thorough study

PD – Standby cannot catchup

- In REMOTE_CATCHUP state
- Calculate the recent logging speed on primary and replaying speed on standby. Compare the results to the average speeds and determine the bottleneck.
  - If there is log retrieve on primary, check db2diag.log retrieve messages to see if log retrieve is a bottleneck.
  - Log reading on primary could be a bottleneck, compared to in PEER state. Check if there are other log reading operations on primary.
  - If using shared archive, and Standby is far behind, consider reactivate standby to retrieve logs from archive

PD – Temporary Standby Replay Slow

- Most likely due to workload being replayed
- Is Standby replaying operations that are known to be slow, such as Offline Reorg?
  - Calculate the time to fully replay the operation, it may be faster to re-initialize standby with a new backup
- Optimize workloads
- Collect db2 stacks, db2pd -dpsprcb, and traces over 10 minutes and resolve via collaboration
- Workaround temporary slow replay with bigger buffer/spool

PD – Consistent Replay Slow

- IO performance
  - Compare iostat results between primary and standby
  - Alternatively use HADR simulator to test disk write on both primary and standby
- Bufferpool performance
  - Compare size of bufferpool between primary and standby using db2pd -bufferpool
  - Self Tuning Memory (STMM) may increase bufferpool size on primary but not on standby

PD – Network Congested

- Has CONGESTED Status
  - Primary is unable to send log records.
  - Sometimes caused by Standby receive being blocked.
  - If not, determine possible cause based on how long Congested Status lasts and how often it triggers.
    - It’s normal to occasionally have a short period of CONGESTED Status during network fluctuations.
    - CONGESTED lasting more than a few seconds means the logging rate might be consistently higher than network capacity to send.

PD – Slow Takeover

- Use db2pd -hadr output to determine if takeover is still progressing.
  - Monitor TAKEOVER_APP_REMAINING_PRIMARY and TAKEOVER_APP_REMAINING_STANDBY fields for the progress of forcing applications off the primary and standby.
- If takeover is stuck at forcing off applications:
  - Use db2pd -applications to view details of applications, kill them manually if needed.
  - Try takeover by force as last resort.

PD – Slow Takeover

- When takeover times out, db2 automatically collect stacks
- If takeover is stuck in redo phase shipping and replaying log records:
  - Check common replay slow problems.
  - Collect db2pd -recovery output.
- If takeover is stuck in undo phase compensating open transactions:
  - Collect db2pd -transactions, find out if any transaction hangs.

PD – Standby Active Log File Piling Up

- By default, Standby only reclaims log file that has been archived on primary
- Check FirstArchNum on Primary and Head Extent on Standby
  - See if Standby is prevented from reclaiming old log files by these values
- Check db2diag.log for db2logalloc messages
  - Ensure the extent number being deleted/renamed make sense
  - If Head Extent suddenly moves up, it could take some time to delete large log files

PD – Standby goes into log shipping when there are log files to retrieve

- Standby retrieves in REMOTE_CACTHUP_PENDING state
  - Standby retrieve is unbounded
  - If Standby replay is far behind, it’s possible for retrieve to hit DISK FULL. Upon error, Standby will stop retrieving and enters REMOTE_CATCHUP state
- For faster catchup, it’s recommended to allow Standby to retrieve as much as possible
  - Log retrieving performs better than log shipping
  - Once retrieved log files are replayed, re-activate standby to let standby retrieve log files again

PD – Standby Table Space Error

- STANDBY_TABLESPACE_ERROR Flag
  - One or more table spaces on standby are in an invalid or error state, the replay of transactions on such table spaces will stop, while the replay of other transactions will continue.
  - Use *db2pd -tablespaces* on standby to identify the table spaces that are in error (non-zero) states; or check db2diag.log to find the exact messages puts the table spaces in error states.
  - Need to restore tablespace before takeover on the standby, otherwise the tablespace will not be available.

PD – Replay Only Windows

- During replay only window, ROS connections are forced out
  - Use DB2_HADR_ROS_AVOID_REPLAY_ONLY_WINDOW regvar (default to ON in 11.5) to avoid most common replay only windows.
- Use db2fmtlog -replayonlywindow tool to check replay only window
  - DDLs are often cause of replay only window, turn database config LOG_DDL_STMTS on to also display DDL statements in the output.
  - Use DDL statements to understand why replay only windows happen.
- If stuck in replay only window, check if ROS connections are forced off

PD – Questions?

HADR Performance Tuning

- Log Shipping
  - Network Speed
  - Db2 HADR Configs – Sync modes, Buffer size, TCP Socket buffer size, Peer Window, Peer Wait Limit, Shared archive
- Log Replay
  - Overall recovery performance
  - Some workloads cause replay to be exceedingly slow
  - Some workloads cause replay only window

HADR Performance Tuning – Tools

- Starting from 11.5, tools are part of installation
- HADR simulator – sqllib/bin/simhadr
  - Estimate HADR performance of current system with different configs
- Db2 Log Scanner – sqllib/bin/db2logscan
  - Scan actual log files to measure logging performance
- HADR calculator – sqllib/samples/perl/hadrCalculator.pl
  - Use db2logscan results to estimate HADR performance with pre-determinted network and disk speed

HADR Performance Tuning – Running HADR Simulator

- Needs to run simhar on both primary and standby hosts
  - Example: `simhadr_linux -lhost portland.ibm.com -lport 4000 -rhost sanfrancisco.ibm.com -rport 5000 -role primary -syncmode ASYNC -t 60`
  - On S: `simhadr_linux -lhost sanfrancisco.ibm.com -lport 5000 -rhost portland.ibm.com -rport 4000 -role standby`
  - Estimate current HADR performance with specified syncmode
- Multiple Standbys – for each standby, needs to run two commands above
- PureScale HADR – for each member, needs to run two commands above

HADR Performance Tuning – HADR Simulator Options

- -rcu Simulates REMOTE_CATCHUP state log shipping, instead of PEER for default.
- -flushSize Specifies size of log flushes on Primary.
- -hadrBufSize Specifies Standby receive buffer size.
- -sockSndBuf, -sockRcvBuf set socket send or recv buffer size.
- -write Tests disk write performance
- -read Tests disk read performance

HADR Performance Tuning – HADR Simulator Outputs

> simhadr -role primary -lhost brant -lport 46000 -rhost auklet -rport 46000 -sockSndBuf 64000 -sockRcvBuf 64000 -flushSize 32

NEARSYNC: Total 410255360 bytes in 4.000821 seconds, 102.542793 MBytes/sec
Total 3130 flushes, 0.001278 sec/flush, 32 pages (131072 bytes)/flush

Total 410255360 bytes sent in 4.000821 seconds. 102.542793 MBytes/sec
Total 10893 send calls, 37.662 KBytes/send,
Total 3136 congestions, 0.848714 seconds, 0.000270 second/congestion

Total 150240 bytes recv in 4.000821 seconds. 0.037552 MBytes/sec
Total 3130 recv calls, 0.048 KBytes/recv

Distribution of log write size (unit is byte):
Total 3130 numbers, Sum 410255360, Min 131072, Max 131072, Avg 131072
Exactly 131072 3130 numbers

Distribution of log shipping time (unit is microsecond):
Total 3130 numbers, Sum 3989623, Min 1262, Max 1622, Avg 1274
From 1024 to 2047 3130 numbers

Distribution of congestion duration (unit is microsecond):
Total 3136 numbers, Sum 848714, Min 208, Max 824, Avg 270
From 128 to 255 120 numbers
From 256 to 511 3011 numbers
From 512 to 1023 5 numbers

Distribution of send size (unit is byte):
Total 10893 numbers, Sum 410255360, Min 752, Max 86880, Avg 37662
From 512 to 1023 192 numbers
From 2048 to 4095 3 numbers
From 4096 to 8191 2661 numbers
From 8192 to 16383 1777 numbers
From 16384 to 32767 2913 numbers
From 32768 to 65535 217 numbers
From 65536 to 131071 3130 numbers

Distribution of recv size (unit is byte):
Total 3130 numbers, Sum 150240, Min 48, Max 48, Avg 48
Exactly 48 3130 numbers

HADR Performance Tuning – db2logscan

```
2023-02-08 23:56:27  0.113 MB/s, 60 sec,   4.3 pg/f, 0.150000 sec/f,   1.0 pg/tr, 0.034558 sec/tr, 0.032931 sec/cmt, nOpenTrans   0.7
2023-02-08 23:57:27  0.097 MB/s, 60 sec,   4.9 pg/f, 0.197368 sec/f,   1.1 pg/tr, 0.045356 sec/tr, 0.043197 sec/cmt, nOpenTrans   0.7
2023-02-08 23:58:27  0.792 MB/s, 60 sec,  17.7 pg/f, 0.087083 sec/f,   6.6 pg/tr, 0.038671 sec/tr, 0.032680 sec/cmt, nOpenTrans   0.9
2023-02-08 23:59:27  0.415 MB/s, 60 sec,  11.2 pg/f, 0.105448 sec/f,   3.2 pg/tr, 0.035371 sec/tr, 0.030318 sec/cmt, nOpenTrans   0.7
...
2023-02-09 01:22:30  2.596 MB/s, 60 sec,   5.8 pg/f, 0.008786 sec/f,  29.4 pg/tr, 0.100666 sec/tr, 0.044412 sec/cmt, nOpenTrans  38.3
2023-02-09 01:23:30  1.709 MB/s, 60 sec,  10.0 pg/f, 0.022945 sec/f,  14.8 pg/tr, 2.731225 sec/tr, 0.033879 sec/cmt, nOpenTrans  28.6
2023-02-09 01:24:30  0.684 MB/s, 60 sec,   5.7 pg/f, 0.032662 sec/f,  11.9 pg/tr, 0.656532 sec/tr, 0.067568 sec/cmt, nOpenTrans   5.1
2023-02-09 01:25:30  0.499 MB/s, 60 sec,   6.4 pg/f, 0.050125 sec/f,   8.6 pg/tr, 0.150000 sec/tr, 0.068182 sec/cmt, nOpenTrans   2.3

transTable: nOpenTran=48 openTranSize=5910673 nBuckets=1024 nAdds=166489 nCollisions=1243 collisionRate=0%

Distribution of log write rate (unit is MB/s):
...

Distribution of flush size (unit is page):
Total 68369 numbers, Sum 499984, Min 1, Max 370, Avg 7.3
From 1 to 1              18615 numbers  27%
From 2 to 3              17843 numbers  26%
From 4 to 7              15782 numbers  23%
From 8 to 15              9621 numbers  14%
From 16 to 31             3847 numbers   5%
From 32 to 63             1728 numbers   2%
From 64 to 127             753 numbers   1%
From 128 to 255            166 numbers   0%
From 256 to 511             14 numbers   0%
...

3906.125 MB scanned in 5.300 seconds, scanning rate 736.941 MB/second
```

HADR Performance Tuning – HADR calculator

```
> hadrCalculator.pl -network 10 .1 -disk 200 .001 <inputFile>

hadrCalculator.pl: Network speed 10 MB/s, .1 second round trip time
hadrCalculator.pl: Disk speed 200 MB/s, .001 second overhead per write

2013-02-26 15:39:05  3.280 MB/s, 56 sec,   6.7 pg/f, 0.007984 sec/f,  15.1 pg/tr, 0.102178 sec/tr, 0.017937 sec/cmt, nOpenTrans  45.0
actual               3.280 MB/s@   6 pg/f 0.007984 s/f
SYNC         ??   3.283 MB/s@ 134 pg/f 0.159649 s/f, min   0.250 MB/s@   6 pg/f 0.104879 s/f, max   6.737 MB/s@ 679 pg/f 0.393973 s/f
NEARSYNC     ??   3.280 MB/s@ 124 pg/f 0.148815 s/f, min   0.255 MB/s@   6 pg/f 0.102617 s/f, max   7.263 MB/s@ 679 pg/f 0.365430 s/f
ASYNC            10.000 MB/s@   6 pg/f 0.002617 s/f, min  10.000 MB/s@   6 pg/f 0.002617 s/f, max  10.000 MB/s@ 679 pg/f 0.265430 s/f
...

Distribution of log write rate (unit is MB/s):
Total 8 numbers, Sum 32.764, Min 1.362, Max 7.306, Avg 4.096
Average rate       4.096 MB/s
REMOTE CATCHUP     10.000 MB/s@  16 pg/f 0.006250 s/f
Exactly     1.362        1 numbers  12%
Exactly     1.987        1 numbers  12%
...

SYNC      Max flush size: predicted 360 pages, workload max 1126 pages
NEARSYNC  Max flush size: predicted 360 pages, workload max 1126 pages
ASYNC     Max flush size: predicted  17 pages, workload max 1126 pages
```

- InputFile is db2logscan output
- Need to provide sync mode, disk speed and network speed

HADR Performance Tuning – Peer Window

- HADR_PEER_WINDOW (database cfg variable) – primary block writing peer window to protect against data loss in SYNC and NEARSYNC modes.
- Cause primary to wait for the duration when standby disconnects
  - In this peer window, HADR is in DISCONNECTED_PEER state
  - No data loss during Peer window for failover to standby
- Works well when Standby rarely disconnects

HADR Performance Tuning – Peer Wait Limit

- DB2_HADR_PEER_WAIT_LIMIT (registry variable) - wait limit for peer state log replication.
  - Default is 0, means no limit.
  - When the limit is reached, break HADR connection.
- Apply for any kind of wait, including the case when Standby blocks receiving due to buffer or spool full
  - HADR_TIMEOUT works currently with DB2_HADR_PEER_WAIT_LIMIT for log write blocking caused by network

HADR Performance Tuning – Recovery Settings

- DB2_RECOVERY_SETTINGS
  - New regvar to tune performance of log recovery operations, including HADR Standby replay
  - NUM_AGENTS – number of db2redow agents, larger -> more parallelism for processing log records, with diminishing returns and even worse performance after reaching a threshold
  - QSETSIZE – size of a queue set, larger -> more log records from a table space can be processed in parallel
  - NUM_QSETS – number of queue sets, larger -> log records from different table spaces are more likely to be put in different queue sets

HADR Performance Tuning – Takeover

- Graceful Takeover
  - Standby needs to finish receiving and replaying all current log files on primary. Try to takeover when standby log replay position is close to the primary log position.
  - Standby needs to undo open transactions on primary. Try to takeover when primary is not experiencing high workloads.
- Forced Takeover
  - Standby only needs to finish replay current log file on standby and undo any current open transactions.

HADR Performance Tuning – Questions?

HADR Best Practices – Optimizing Workloads

- Spread workload over space and time to avoid spikes
  - Use multiple tablespaces so workloads are separated to allow more parallelism in Standby replay
  - Avoid clustering operations that generate many log records or take a long time to replay, such as LOAD or Reorg
- Avoid inefficient operations such as using TRUNC_TABLE on relatively small tables

HADR Best Practices – Optimizing Index

- Reduce number of index
  - Review usefulness of index defined, fewer index -> Less logging overhead in Primary and faster replay on Standby
- Put index in a separate table spaces from data tables
  - Replaying index log records would not impact processing of data table log records
- Use largest page size for index
  - Less the frequency of index page splits log records -> faster replay on Standby

HADR Best Practices – Shared Archive

- HADR system can benefit from a shared archive to avoid network and primary log reading overheads
  - HADR Standby only retrieves log files in REMOTE_CATCHUP_PENDING state after activation
  - In REMOTE_CATCHUP state, primary needs to read log files from disk to send to standbys
- One central location for all archive log files. Only HADR primary can archive, but all the HADR standbys can retrieve.
  - No need to find archive log files after takeover

HADR Best Practices – Standby Log File Management

- HADR automatically write and manage log files on the standby in the same way as that in primary. The log files on standby is logically identical to those on primary.
  - Use same or slightly bigger log space / disk for standby
- A log record is only replayed when it’s written to the disk.
- With archiving, once a log file has been replayed on standby and archive on primary, it can be renamed to become a new log file.
  - If there are too many old log files, they will be deleted asynchronously

HADR Best Practices – Reorg

- Online Reorg – logging in fine granularity.
  - HADR Standby can replay normally without falling further behind than usual
  - Generate large amount of logs
  - If a takeover happens during reorg, new primary cannot continue reorg, only restart
- Offline Reorg
  - Generate a few logs
  - Replay would take a long time to go through one log record

HADR Best Practices – Load

- Only a successful LOAD COPY YES will be replicated to Standby
  - Standby needs to be able to access the load copy image
  - Network could impact the reading of load copy image on Standby, causing LOAD operation to be replayed slowly
- If a LOAD COPY NONRECOVERABLE is executed on Primary, or if a LOAD COPY YES failed to replicate to Standby, the table will be marked bad on Standby
  - Can be recovered with LOAD COPY YES REPLACE command on primary

HADR Best Practices – HADR Update and Upgrade

- Recommended to use rolling update and upgrade without standby reinitialization for HADR system
- Rolling update
  - Must be in PEER state (except SUPERASYNC)
  - Update Standby -> Takeover -> Update old Primary
- HADR Seamless Upgrade
  - Standby must replay all log files
  - db2ckupgrade must pass on all hosts
  - Upgrade Standby -> Upgrade Primary
  - Standby will replay logs for upgrade

Resources & Questions

- HADR Wiki
- Optimizing Reocvery performance
- Tuning and Monitoring Database System Performance
- Rolling Update
- HADR Upgrade without standby reinitialization