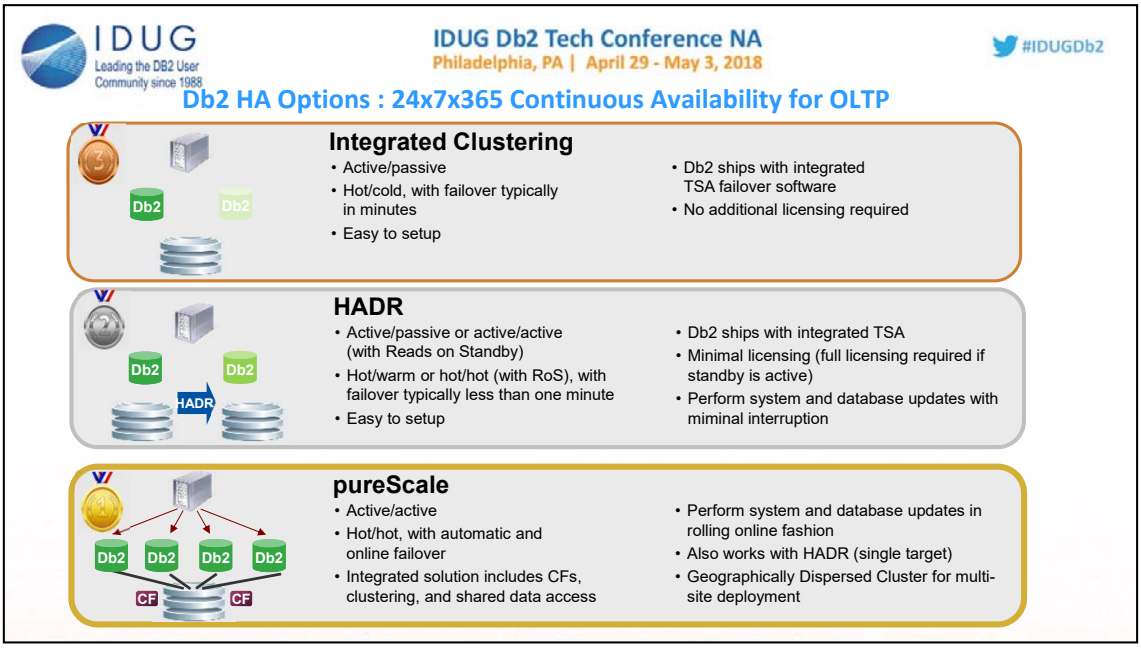

IDUG
Leading the DB2 User Community since 1988
IDUG Db2 Tech Conference NA
Philadelphia, PA | April 29 - May 3, 2018
#IDUGDb2
HADR Multiple Standby Deep Dive
Dale McInnis
IBM Canada Ltd.
Session code: D12
Wed May 2: 02:20 PM - 03:20 PM
Platform: Linux, UNIX and Windows

Agenda

- Comparison to other DB2 HA offerings
- HADR Review
- HADR Sync modes
- HADR Log Spooling
- HADR Time Delay
- HADR Multiple Standby
- HADR Configuration
- HADR Monitoring
- Automatic Client Reroute
- Futures



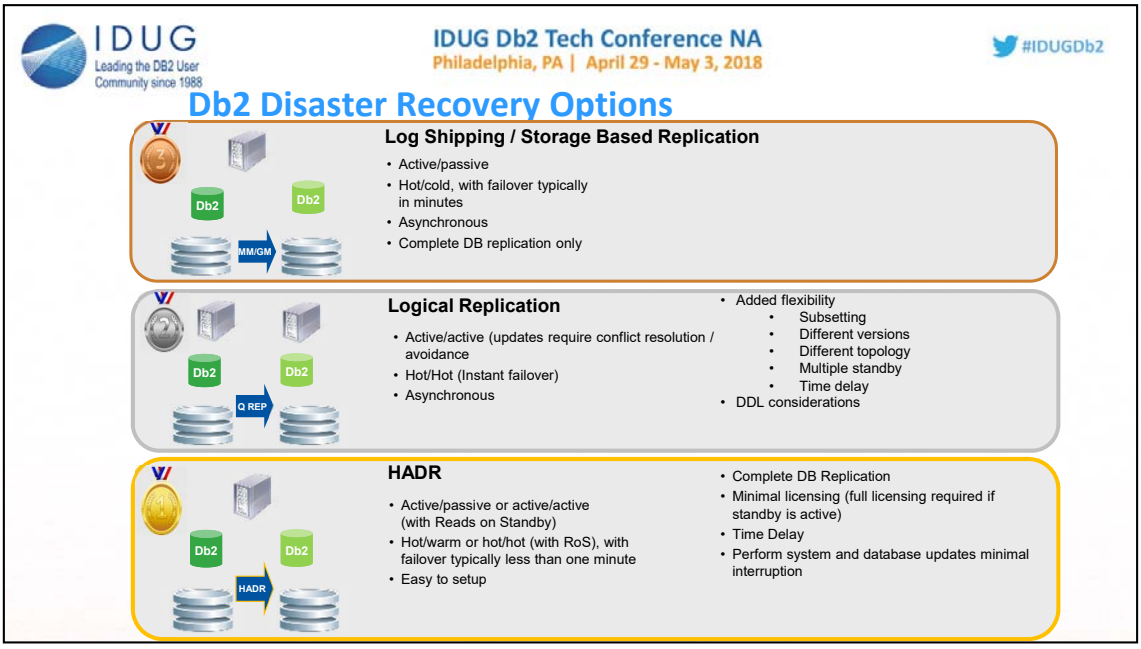
This slide shows the different high availability (HA) options available with Db2 (although not categorized as such here, you may sometimes hear them talked about as "bronze", "silver", and "gold" levels of availability – going from top to bottom here on the slide). Each option is a winning configuration, but each one provides a different level of availability with different pros and cons. Typically, when total costs are factored in (hardware, software, implementation resources), higher availability typically comes with a higher price tag.

It should be pointed out that not every database requires the highest levels of availability. It might not be a big deal to an organization if a particular departmental database is offline for 20 minutes, or an hour, or even the entire day. But there are certainly some databases that are considered "tier 1" that do require the highest availability possible. Therefore, it is important to understand the availability requirements and the costs associated with the different solutions and choose one based on those considerations.

The first solution shown here is integrated clustering. This is a hot/cold type of environment where there is a primary active server and a passive standby server. The storage is either shared between the servers or the storage can be replicated from the primary system to the standby system. Db2 has Tivoli Systems Automation (TSA) packaged in with it to detect failures and automate the failover process. Failover times are typically in the 1-2 minute range.

The next solution shown here is HADR. HADR is a solution for both HA and DR but we're talking about it in the context of HA exclusively here. This configuration allows for a primary database and up to three standby databases (as of Db2 10.1). Transaction log records are shipped from the primary to the standby and the standby is kept in sync with the primary (based on whatever HADR sync mode is chosen). It can be run in a hot/warm type of configuration (no work done on standby, but replay is constantly being done, buffer pools are primed, etc.) or in a hot/semi-hot type of configuration (with reads being allowed on the standby). Integrated TSA allows for automated failover of a primary database to it's principle standby database. Failover times are typically less than a minute.

The solution with the highest levels of availability is pureScale. It's an active/active solution that provides the utmost in availability. If a node fails, the rest of the cluster remains online with most of the data in the database remaining available with no down time. Db2's pureScale solution, simply put, is not high availability, it is continuous availability. With support for 2 CFs (caching facility) for redundancy and up to 128 members, there is no single point of failure, be it software, OS, or hardware. Need to perform patching/updates – do this without impacting database and thus application availability in a rolling update manner. Not a single second of downtime, always maintaining 100% access to all the data.



This slide shows the different high availability (HA) options available with Db2 (although not categorized as such here, you may sometimes hear them talked about as "bronze", "silver", and "gold" levels of availability – going from top to bottom here on the slide). Each option is a winning configuration, but each one provides a different level of availability with different pros and cons. Typically, when total costs are factored in (hardware, software, implementation resources), higher availability typically comes with a higher price tag.

It should be pointed out that not every database requires the highest levels of availability. It might not be a big deal to an organization if a particular departmental database is offline for 20 minutes, or an hour, or even the entire day. But there are certainly some databases that are considered "tier 1" that do require the highest availability possible. Therefore, it is important to understand the availability requirements and the costs associated with the different solutions and choose one based on those considerations.

The first solution shown here is integrated clustering. This is a hot/cold type of environment where there is a primary active server and a passive standby server. The storage is either shared between the servers or the storage can be replicated from the primary system to the standby system. Db2 has Tivoli Systems Automation (TSA) packaged in with it to detect failures and automate the failover process. Failover times are typically in the 1-2 minute range.

The next solution shown here is HADR. HADR is a solution for both HA and DR but we're talking about it in the context of HA exclusively here. This configuration allows for a primary database and up to three standby databases (as of Db2 10.1). Transaction log records are shipped from the primary to the standby and the standby is kept in sync with the primary (based on whatever HADR sync mode is chosen). It can be run in a hot/warm type of configuration (no work done on standby, but replay is constantly being done, buffer pools are primed, etc.) or in a hot/semi-hot type of configuration (with reads being allowed on the standby). Integrated TSA allows for automated failover of a primary database to it's principle standby database. Failover times are typically less than a minute.

The solution with the highest levels of availability is pureScale. It's an active/active solution that provides the utmost in availability. If a node fails, the rest of the cluster remains online with most of the data in the database remaining available with no down time. Db2's pureScale solution, simply put, is not high availability, it is continuous availability. With support for 2 CFs (caching facility) for redundancy and up to 128 members, there is no single point of failure, be it software, OS, or hardware. Need to perform patching/updates – do this without impacting database and thus application availability in a rolling update manner. Not a single second of downtime, always maintaining 100% access to all the data.

DB2 Disaster Recovery Options : Continued

GDPC

- Active / active (fully coherent)
- Hot / hot (**online** failover)
- Synchronous
- Complete DB replication
- Continuous testing of DR site
- Distance limitations

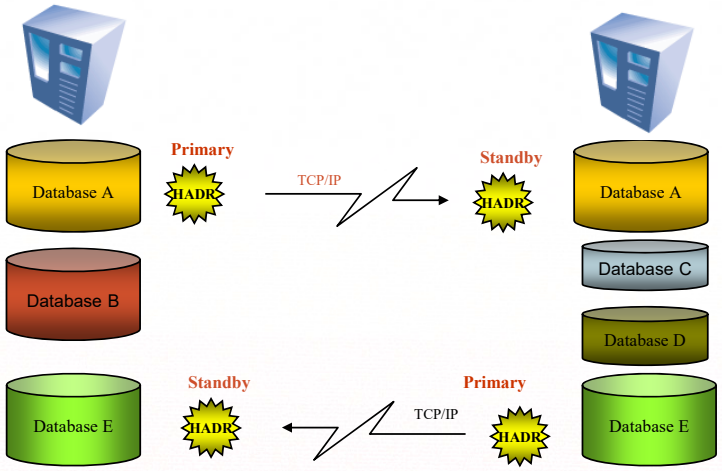
Situational Platinum

Agenda

- Comparison to other DB2 HA offerings
- HADR Review
- HADR Sync Modes
- HADR Log Spooling
- HADR Time Delay
- HADR Multiple Standby
- HADR Configuration
- HADR Monitoring
- Automatic Client Reroute
- Futures

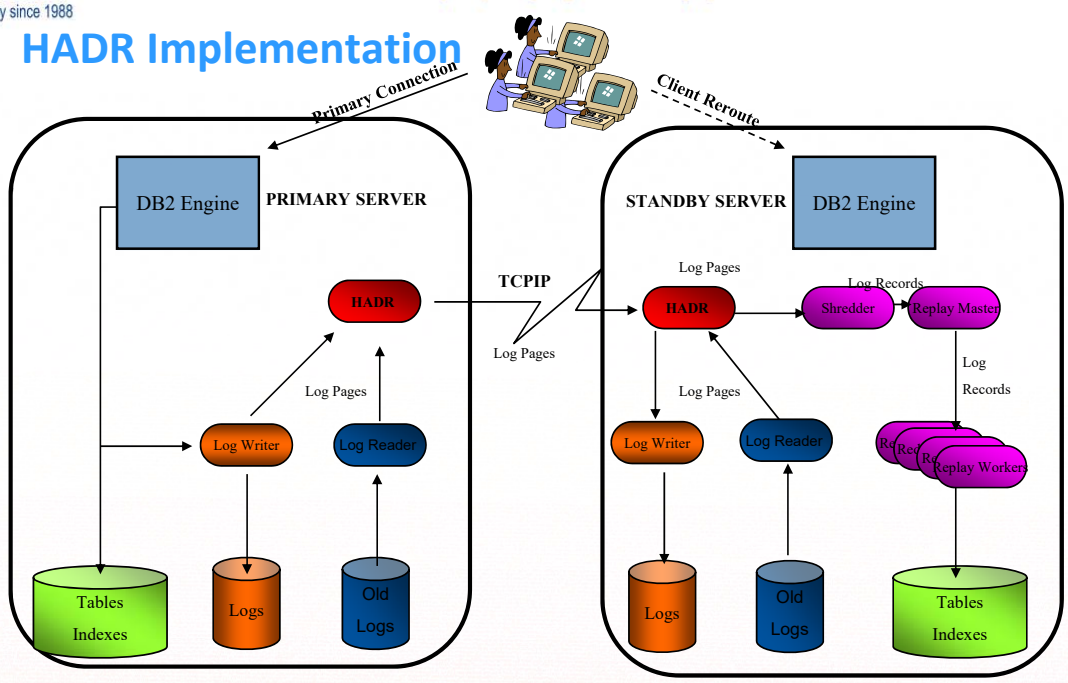
Scope of Action

HADR replication takes place at the database level.



The standby server can have multiple database from multiple primaries on it

HADR Implementation

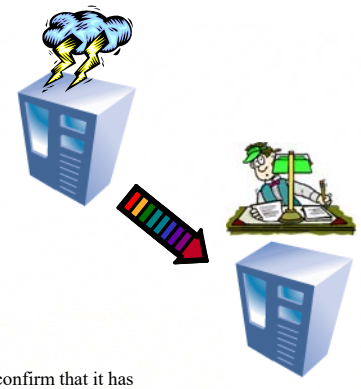


What's replicated, what's not?

- Logged operations are replicated
  - Example: DDL, DML, table space and buffer pool creation/deletion.
- Not logged operations are not replicated.
  - Example: database configuration parameter. not logged initially table, UDF libraries.
- Index pages are not replicated unless LOGINDEXBUILD is enabled
  - Ensure logsecond is maxed out as index rebuild is a single transaction
- How do I prevent non-logged operations?
  - Enable BLOCKNONLOGGED db cfg parameter

Failing Over : Simple "TAKEOVER" Command

- Normal TAKEOVER
  - Primary and standby switch roles as follows:
    1. Standby tells primary that it is taking over.
    2. Primary forces off all client connections and refuses new connections.
    3. Primary rolls back any open transactions and ships remaining log, up to the end of log, to standby.
    4. Standby replays received log, up to end of the log.
    5. Primary becomes new standby.
    6. Standby becomes new primary



- Emergency TAKEOVER (aka 'Forced' TAKEOVER)
  - The standby sends a notice asking the primary to shut itself down.
  - The standby does NOT wait for any acknowledgement from the primary to confirm that it has received the takeover notification or that it has shut down
  - The standby stops receiving logs from the primary, finishes replaying the logs it has already received, and then becomes a primary.

```
TAKEOVER HADR ON DATABASE <dbname>
        <USER <username> [USING <password>]] [BY FORCE]
```

Primary Reintegration

- After primary failure and takeover, allow old primary to reintegrate as a standby with the new primary (saves user from having to reinitialize standby from scratch)
- Differentiating feature for DB2 HADR – competitors do not support this
- Reintegration possible if old primary can be made consistent with new primary
- Some conditions to satisfy, e.g. old primary crashed in peer state and had no disk updates that were not logged on old standby; some other details.
- Successful reintegration is most likely in SYNC mode, least likely in ASYNC and SUPERASYNC modes
- Synchronization with tail of the log file

HADR Setup Fits on One Slide

Primary Setup

db2 backup db hadr_db to backup_dir

db2 update db cfg for hadr_db using
HADR_LOCAL_HOST host_a
HADR_LOCAL_SVC svc_a
HADR_REMOTE_INST inst_b
HADR_TARGET_LIST host_b:svc_b
HADR_TIMEOUT 120
HADR_SYNCMODE ASYNC

db2 start hadr on database hadr_db as primary

Standby Setup

db2 restore db hadr_db from backup_dir

db2 update db cfg for hadr_db using
HADR_LOCAL_HOST host_b
HADR_LOCAL_SVC svc_b
HADR_REMOTE_INST inst_a
HADR_TARGET_LIST host_a:svc_a
HADR_TIMEOUT 120
HADR_SYNCMODE ASYNC

db2 start hadr on database hadr_db as standby

HADR Timeout

- While connected, the Primary and Standby exchange heartbeat messages
- The user can configure a timeout value using the database configuration parameter HADR_TIMEOUT
- If no message is received for the duration of HADR_TIMEOUT seconds, the TCP connection will be closed
  - A standby will then attempt to re-establish the connection by sending a handshake message to the primary
  - A primary will send a redirection message to the standby to probe it to start the handshake protocol
- Heartbeat interval is the minimum of the following:
  - 1/4 of HADR_TIMEOUT
  - 1/4 of HADR_PEER_WINDOW
  - 30 seconds
  - Find the exact heartbeat interval using the monitor element HEARTBEAT_INTERVAL

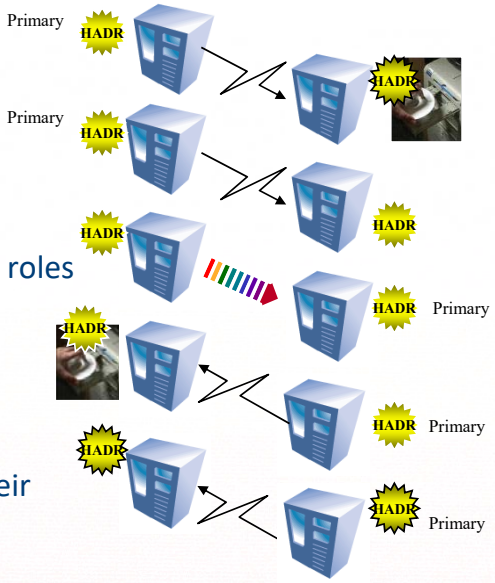
HADR Timeout

- If HADR_TIMEOUT is too long, it will slow detection of a lost connection or a failed standby
  - This may end up blocking transactions on primary
- It HADR_TIMEOUT is too short, HADR may get too many false alarms
  - Resulting in breaking the connection more often than necessary

- BEST PRACTICES:
  - The recommended HADR_TIMEOUT is at least 60 seconds
  - The default is 120 seconds
  - Some customers set HADR_TIMEOUT to very low values in order to avoid ever blocking the primary, at the cost of a disconnection every time the network hiccups

Software upgrades on the fly

1. HADR in peer state
2. Deactivate HADR on the Standby
3. Upgrade the standby
4. Start the standby again
   - Let it catch-up with primary
5. Issue a normal TAKEOVER
   - The primary and standby change roles
6. Deactivate the new standby
7. Upgrade the new standby
8. Reactivate the new standby
   - Let it catch-up with primary
9. Optionally, TAKEOVER again
   - The primary and standby play their original roles



The above sequence of events allow you to apply a DB2 fixpack, an OS patch or a hardware change with minimal impact on the application (likely less than 5 seconds in most cases).

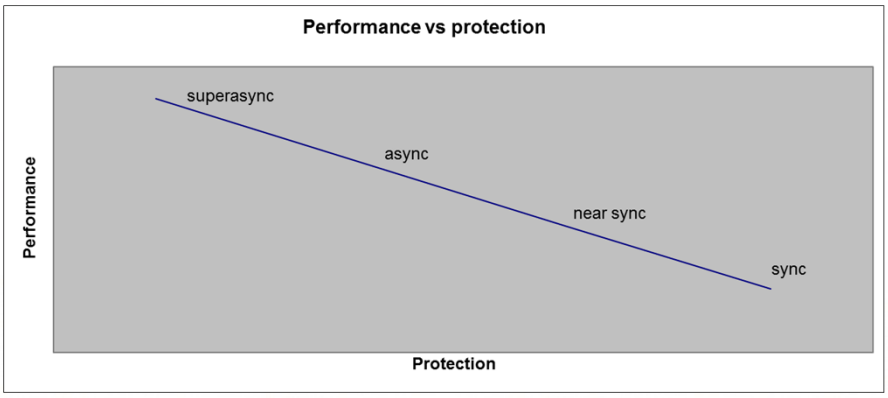
Diagnostics

- db2diag.log is the most important diagnostic tool.
  - Look for hdrSetHdrState() trace points. All HADR state transition goes through this function.
  - You can also search for all messages produced by the HADR EDU.
  - If there are multiple HADR databases in the instances, be sure to distinguish messages from different databases.
  - What about the “This operation is not allowed on a standby database” message?
    - It indicates that a client (possibly an admin tool) is trying to connect to the standby database.

Agenda

- Comparison to other DB2 HA offerings
- HADR Review
- HADR Sync Modes
- HADR Log Spooling
- HADR Time Delay
- HADR Multiple Standby
- HADR Configuration
- HADR Monitoring
- Automatic Client Reroute
- Futures

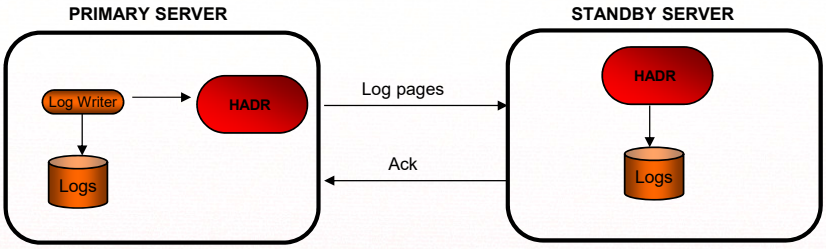
Performance Overhead varies with sync mode



- HADR overhead on primary database logging varies depending on the synchronization mode
  - Stronger sync mode provides more HA and DR protection
  - Weaker sync mode has less impact on the primary database

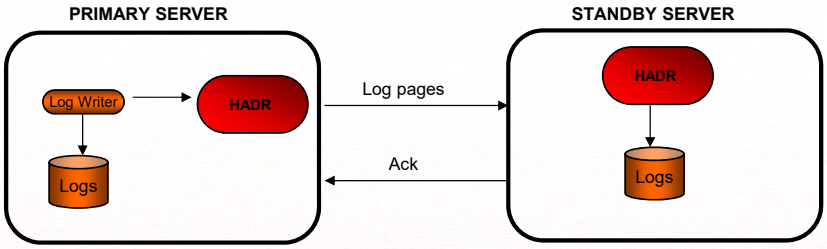
Synchronization Modes

- SYNC mode:
  - Logs are first written to the primary and are only then sent to standby (Serially)
  - Two on-disk copies of the log data are required for transaction commit
  - Total log write time = primary log write + log send + standby log write + ack message
  - Best data protection
  - But the cost of replication is higher than all other modes



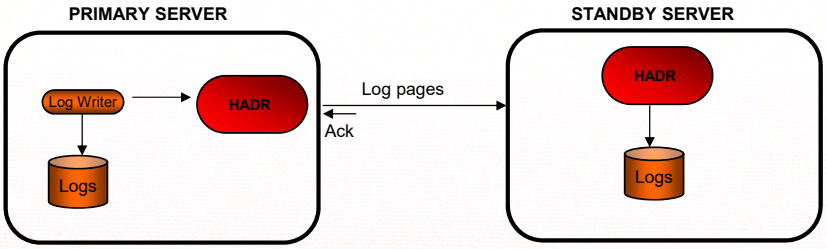
Synchronization Modes

- NEARSYNC mode:
  - Writing logs on the primary and sending logs to standby are done in parallel
  - Standby sends ack message as soon as it receives the logs
  - On a fast network, log replication results in little or no overhead to primary
  - Total log write time = Max (primary log write, log send + ack message)
  - Exposure to the relatively rare 'double failure' scenario
    - primary fails and the standby fails before it has a chance to write received logs to disk
  - Good choice for many applications
  - providing near synch protection at lower performance cost



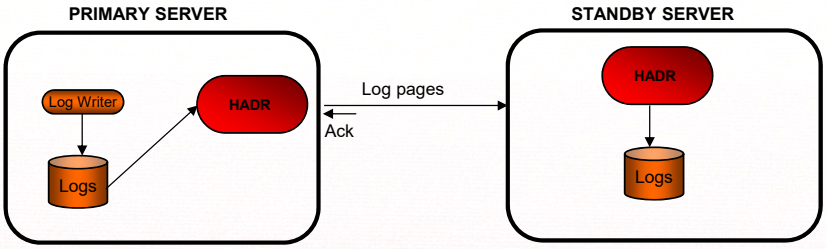
Synchronization Modes

- ASYNC mode
  - Writing logs on the primary and sending logs to standby are done in parallel
  - Does not wait for ack messages from the standby
    - Just the ack that the message has been sent
  - If the primary database fails, there is a higher chance that logs in transit are lost
  - Total log write time = Max (Primary log write rate, Submit log for sending)
  - Well suited for WAN application since network transmission delay does not impact performance



Synchronization Modes

- SUPERASYNC mode
  - Log writing and log shipping are completely independent
  - HADR remains in remote catchup state and never enters peer state
  - Zero impact on Log writing: Total log write time = Primary log write
  - But the log gap between the primary and the standby can grow
    - In a failover, data in the gap will be lost.
  - This mode has the least impact on primary, at the cost of the lowest data protection

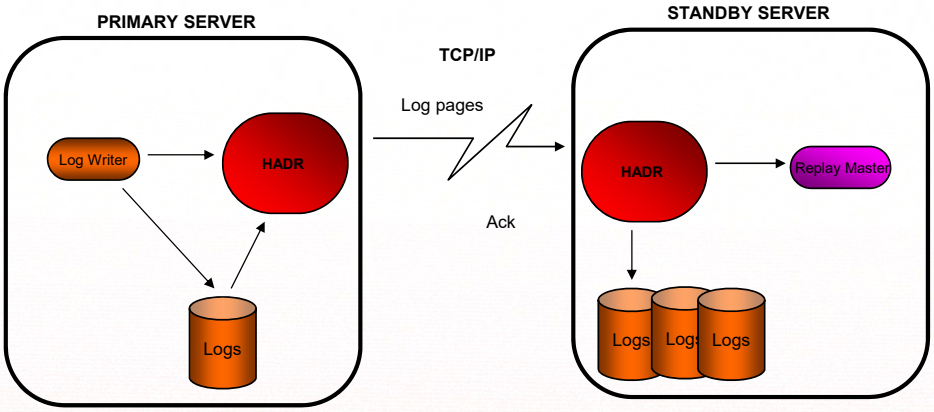


Agenda

- Comparison to other DB2 HA offerings
- HADR Review
- HADR Sync modes
- HADR Log Spooling
- HADR Time Delay
- HADR Multiple Standby
- HADR Configuration
- HADR Monitoring
- Automatic Client Reroute
- Futures

HADR Log Spooling

- All log records are written to the active log path on the standby
- Synchronization Mode is with respect to the log data shipped not with respect to the replay
- Size of spool controlled by the HADR_SPOOL_LIMIT configuration parameter
  - In v10.1 defaults to 0 (i.e. spooling disabled)
  - In v10.5 defaults to AUTOMATIC which is (LOGPRIMARY + LOGSECOND) log files, computed at HADR startup



The operations on the critical path are

- Sending log records
- Receiving log records
- Writing log records to disk
- Replaying log records

Agenda

- Comparison to other DB2 HA offerings
- HADR Review
- HADR Sync Modes
- HADR Log Spooling
- HADR Time Delay
- HADR Multiple Standby
- HADR Configuration
- HADR Monitoring
- Automatic Client Reroute
- Futures

HADR TIME DELAY

New configuration parameter which will control how far behind the standby will remain at all times to prevent data loss due to errant transactions

hadr_replay_delay :

- This parameter specifies the time that must have passed from when the data is changed on primary before these changes would be reflected on the standby database. The time is specified in number of seconds

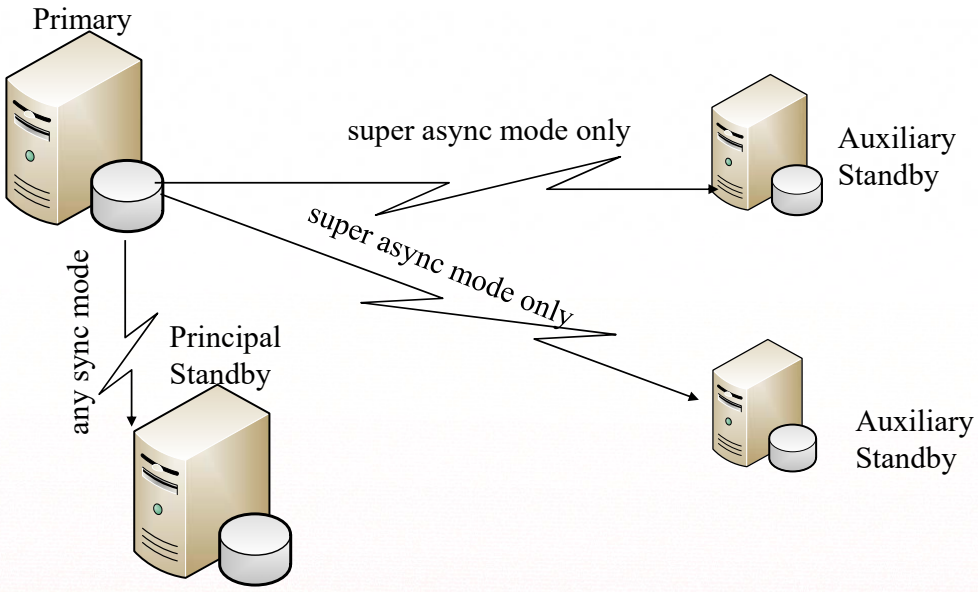
HADR TIME DELAY Restrictions

- A TAKEOVER command on a standby with replay delay enabled will fail.
  - You must first set the hadr_replay_delay configuration parameter to 0 and then deactivate and reactivate the standby to pick up the new value, and then issue the TAKEOVER command.
- The delayed replay feature is supported only in SUPERASYNC mode

Agenda

- Comparison to other DB2 HA offerings
- HADR Review
- HADR Sync Modes
- HADR Log Spooling
- HADR Time Delay
- HADR Multiple Standby
- HADR Configuration
- HADR Monitoring
- Automatic Client Reroute
- Futures

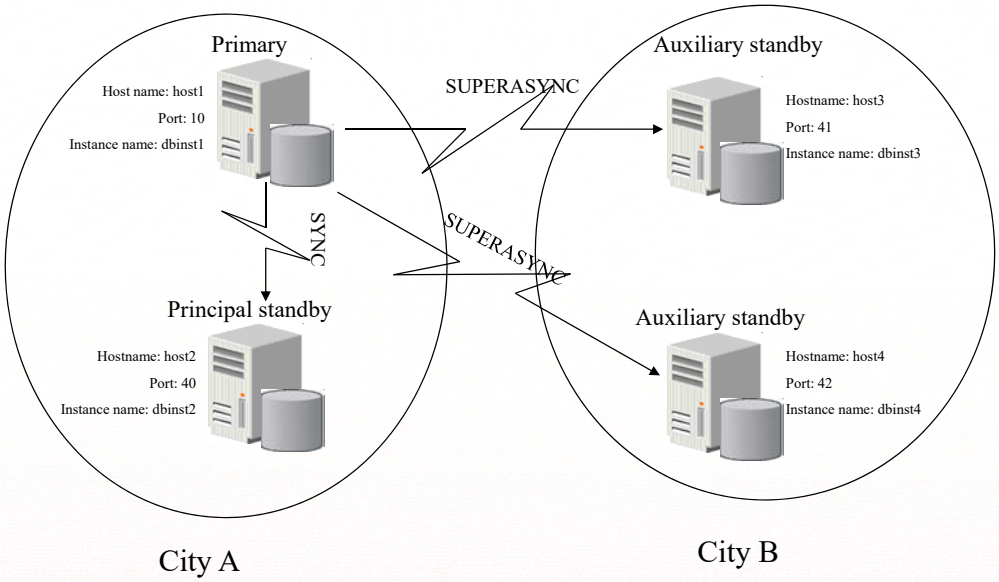
HADR Multiple Standby Overview



HADR Multiple Standby Features

- Principal Standby (PS) equivalent to standby today
  - PS supports any sync mode
  - Can automate takeover using integrated TSA
- Support for up to two(2) Auxiliary Standbys (AS)
  - AS supports super async mode only
  - No automated takeover supported
  - Always feed from the current primary
  - Can be added dynamically

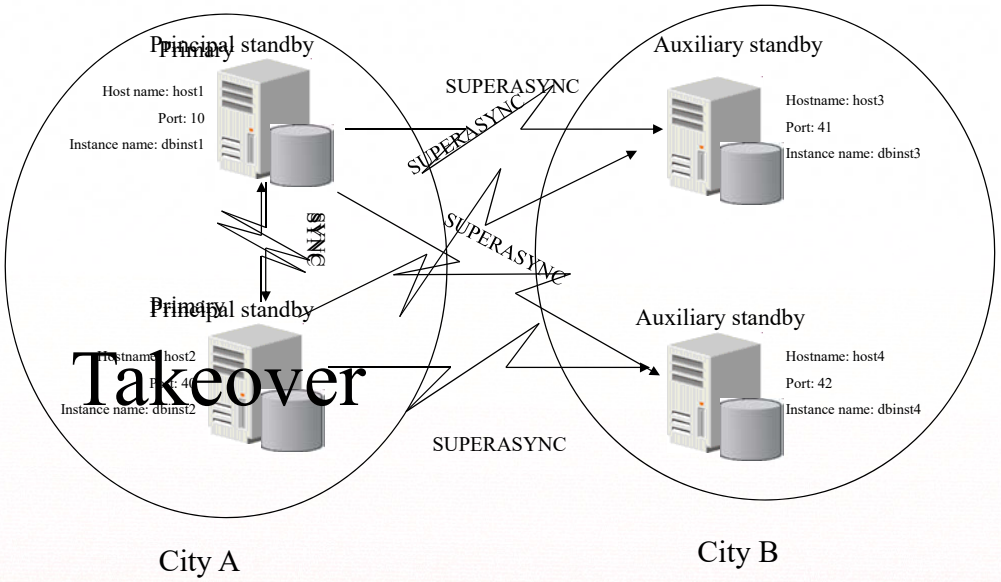
HADR Multiple Standby Example



Configuration values for each host

Configuration parameter	Host1 (Primary)	Host2 (Principal Standby)	Host3 (Aux. Standby)	Host4 (Aux. Standby)
Hadr_target_list	host2:40 | host3:41 | host4:42	host1:10 | host3:41 | host4:42	Host4:40 | host1:10 | host2:42	host3:41 | host1:10 | host2:40
Hadr_remote_host	host2	host1	host1	host1
Hadr_remote_svc	40	10	10	10
Hadr_remote_inst	dbinst2	dbinst1	dbinst1	dbinst1
Hadr_local_host	host1	host2	host3	host4
Hadr_local_svc	10	40	41	42
Operational Hadr_syncmode	sync	sync	Near sync	Async
Effective Hadr_syncmode	N/A	sync	superasync	superasync

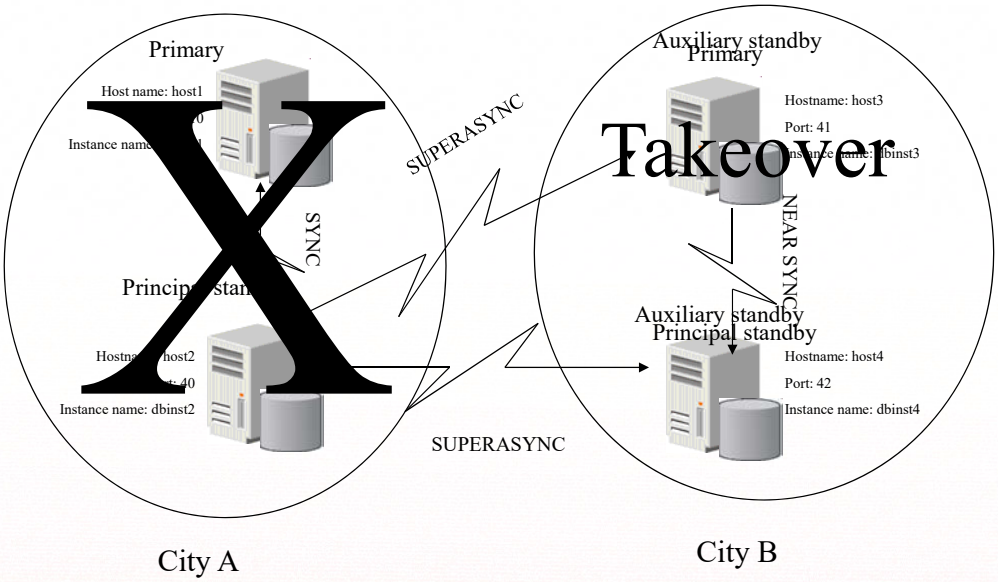
HADR Multiple Standby Role Reversal Example



After issuing takeover on host2 (auto reconfigured)

Configuration parameter	Host1 (Principal Standby)	Host2 (Primary)	Host3 (Aux. Standby)	Host4 (Aux Standby)
Hadr_target_list	host2:40 | host3:41 | host4:42	host1:10 | host3:41 | host4:42	Host4:40 | host1:10 | host2:42	host3:41 | host1:10 | host2:40
Hadr_remote_host	host2	host1	host2	host2
Hadr_remote_svc	40	10	40	40
Hadr_remote_inst	dbinst2	dbinst1	dbinst2	dbinst2
Hadr_local_host	host1	host2	host3	host4
Hadr_local_svc	10	40	41	42
Operational Hadr_syncmode	sync	sync	Near sync	Async
Effective Hadr_syncmode	sync	N/A	superasync	superasync

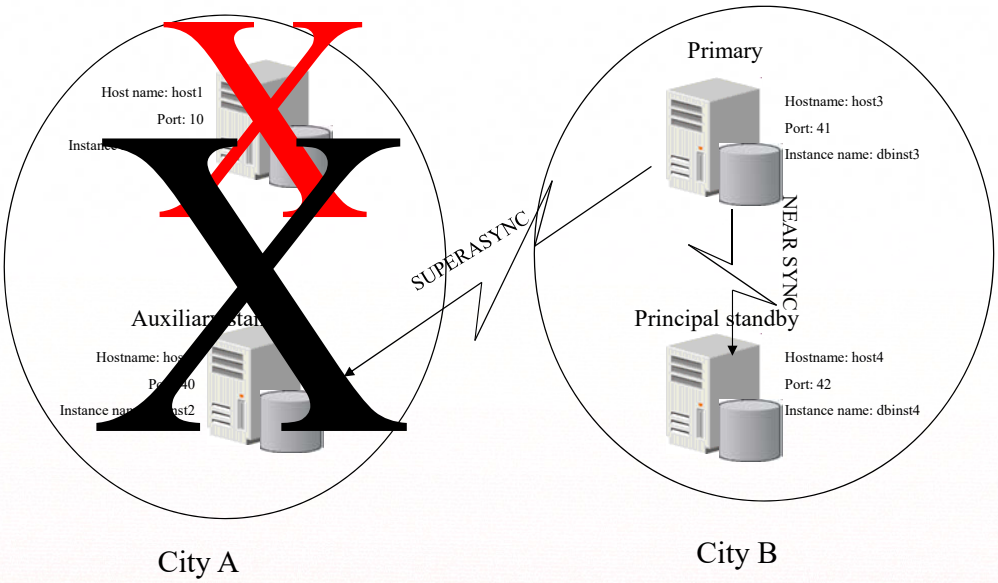
DC #1 goes down - Issue takeover on Host 3



After issuing takeover on host3 (host 1+2 are down)

Configuration parameter	Host1 (Offline)	Host 2 (Offline)	Host3 (Primary)	Host4 (Principal Standby)
Hadr_target_list	host2:40 | host4:42	host1:10 | host4:42	host4:40 | host1:10 | host2:42	host3:41 | host1:10 | host2:40
Hadr_remote_host	host2	host1	host4	host3
Hadr_remote_svc	40	10	42	41
Hadr_remote_inst	dbinst2	dbinst1	dbinst4	dbinst3
Hadr_local_host	host1	host2	host3	host4
Hadr_local_svc	10	40	41	42
Operational Hadr_syncmode	sync	sync	Near sync	Async
Effective Hadr_syncmode	N/A	sync	n/a	Near sync

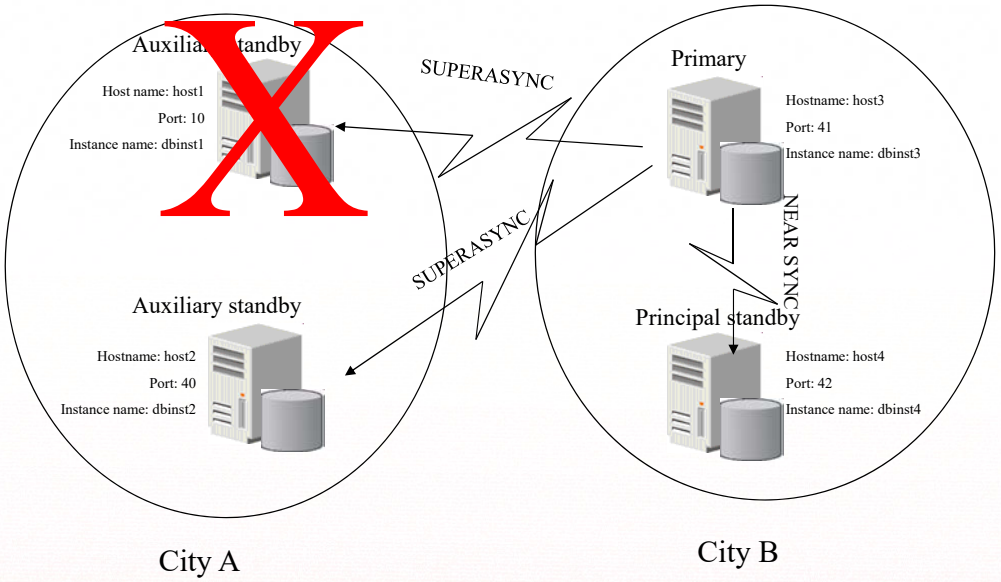
Host 2 now back online



Host 2 now back online

Configuration parameter	Host1 (Offline)	Host2 (Aux. Standby)	Host3 (Primary)	Host4 (Principal Standby)
Hadr_target_list	host2:40 | host3:4 host4:42	host1:10 | host3:41 | host4:42	host4:40 | host1:10 | host2:42	host3:41 | host1:10 | host2:40
Hadr_remote_host	host2	host3	host4	host3
Hadr_remote_svc	40	41	42	41
Hadr_remote_inst	dbinst2	dbinst3	dbinst4	dbinst3
Hadr_local_host	host1	host2	host3	host4
Hadr_local_svc	10	40	41	42
Operational Hadr_syncmode	sync	sync	Near sync	Async
Effective Hadr_syncmode	N/A	Super Async	n/a	Near Sync

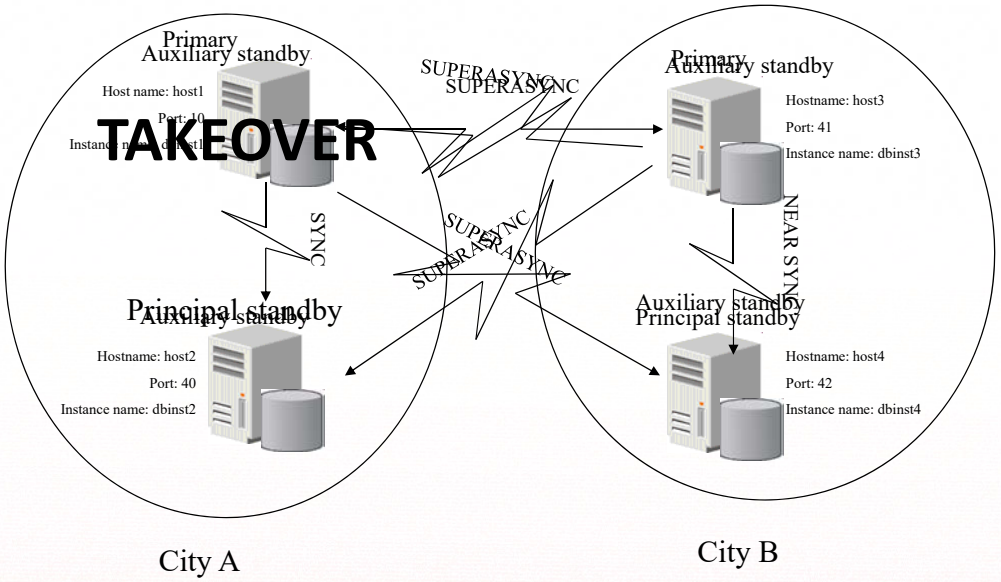
Host 1 now back online



Host 2 now back online

Configuration parameter	Host1 (Offline)	Host2 (Aux. Standby)	Host3 (Primary)	Host4 (Principal Standby)
Hadr_target_list	host2:40 | host3:41 | host4:42	host1:10 | host3:41 | host4:42	host4:40 | host1:10 | host2:42	host3:41 | host1:10 | host2:40
Hadr_remote_host	host3	host3	host4	host3
Hadr_remote_svc	41	41	42	41
Hadr_remote_inst	dbinst3	dbinst3	dbinst4	dbinst3
Hadr_local_host	host1	host2	host3	host4
Hadr_local_svc	10	40	41	42
Operational Hadr_syncmode	sync	sync	Near sync	Async
Effective Hadr_syncmode	Super Async	Super Async	n/a	Near Sync

Revert back to original DC

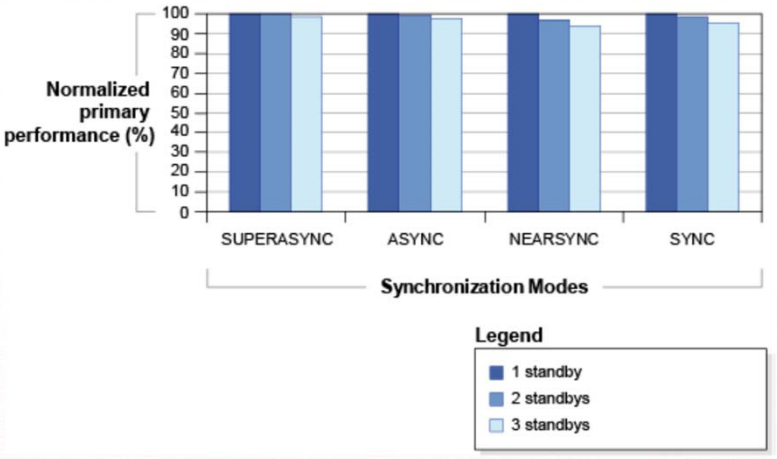


Configuration values for each host

Configuration parameter	Host1 (Primary)	Host2 (Principal Standby)	Host3 (Aux. Standby)	Host4 (Aux. Standby)
Hadr_target_list	host2:40 | host3:41 | host4:42	host1:10 | host3:41 | host4:42	Host4:40 | host1:10 | host2:42	host3:41 | host1:10 | host2:40
Hadr_remote_host	host2	host1	host1	host1
Hadr_remote_svc	40	10	10	10
Hadr_remote_inst	dbinst2	dbinst1	dbinst1	dbinst1
Hadr_local_host	host1	host2	host3	host4
Hadr_local_svc	10	40	41	42
Operational Hadr_syncmode	sync	sync	Near sync	Async
Effective Hadr_syncmode	N/A	sync	superasync	superasync

Multiple standby Performance impact

- The performance impact on the primary of adding additional standby targets is approximately 3%



For details on HADR with multiple standby target see http://www.ibm.com/developerworks/data/library/long/dm-1206hadrmultiplestandby/

What is different when running in a Multi-Standby environment

- It is even more important than ever that you have a shared log archive location and a shared load copy image location
- Reads on Standby is supported on ALL standbys
- Db2pd on the primary shows details on all members, db2pd on the standby only knows about itself.
- Keeps the systems as similar as possible
  - SSD versus HDD.
- Rolling updates – start with the standbys first and the primary is last

Agenda

- Comparison to other DB2 HA offerings
- HADR Review
- HADR Sync Modes
- HADR Log Spooling
- HADR Time Delay
- HADR Multiple Standby
- HADR Configuration
- HADR Monitoring
- Automatic Client Reroute
- Futures

HADR Configuration Parameters Updates

- you need only stop and start HADR for updates to some HADR configuration parameters for the primary database to take effect. You do not have to deactivate and reactivate the database. This dynamic capability affects only the primary database because stopping HADR deactivates any standby database.
- The affected configuration parameters are as follows:
  - **hadr_local_host**
  - **hadr_local_svc**
  - **hadr_peer_window**
  - **hadr_remote_host**
  - **hadr_remote_inst**
  - **hadr_remote_svc**
  - **hadr_replay_delay**
  - **hadr_spool_limit**
  - **hadr_syncmode**
  - **hadr_target_list**
  - **hadr_timeout**

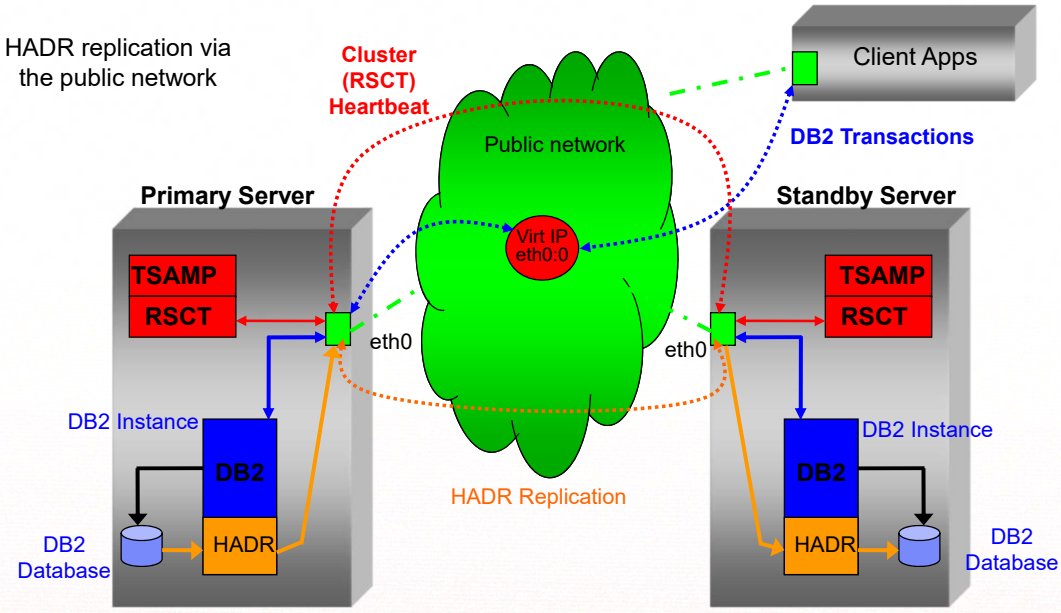
HADR Automation using TSA/MP

- Follow instructions in

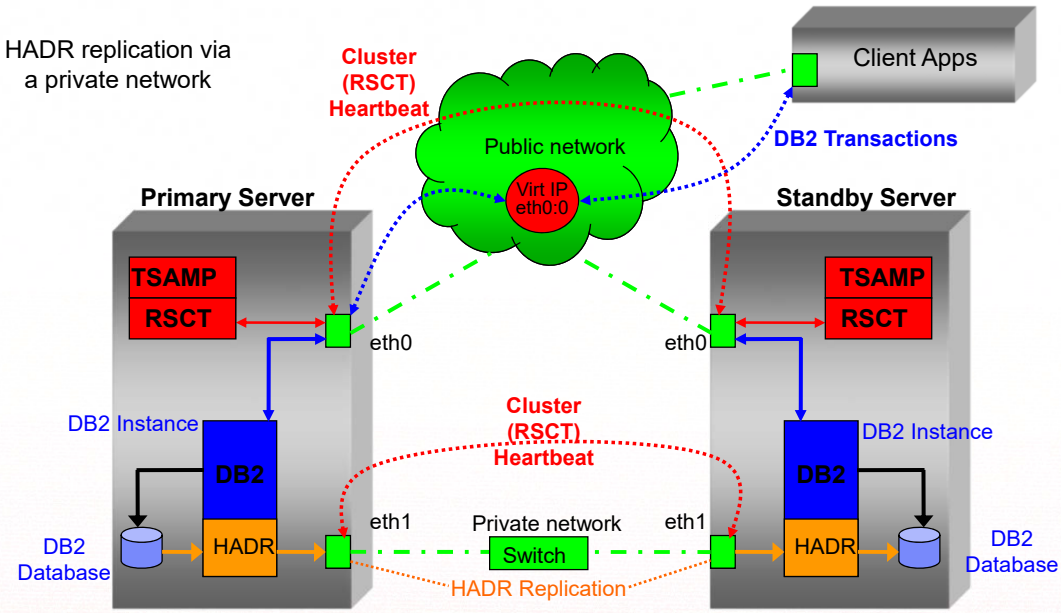
http://download.boulder.ibm.com/ibmdl/pub/software/dw/data/dm-0908hadrdb2haicu/HADR_db2haicu.pdf

- Items to watch out for (source: Ember Crooks)
  - Ensure that you have picked either the server's short name or the server's long name and are using it consistently in each of:
    - /etc/hosts
    - HADR configuration parameters in db cfg
    - db2nodes.cfg (in $HOME/sqllib)
    - Results of the 'hostname' command
    - Ensure the hostname case is the same in all location
  - Ensure that you successfully executed the preprpnode command on both hosts
  - Ensure that you successfully executed the db2cptsa command on both hosts

Example 1 of a DB2 HADR environment



Example 2 of a DB2 HADR environment



Use db2haicu to configure TSA for HADR automation

db2haicu [-f <*XML-input-file-name*>]
[-disable]
[-delete [dbpartitionnum <*db-partition-list*> | hadrdb <*database-name*>]]

- Default is to run in interactive mode
- Follow the instructions in https://www.ibm.com/developerworks/mydeveloperworks/blogs/nsharma/resource/HADR-db2haicu_v5.pdf

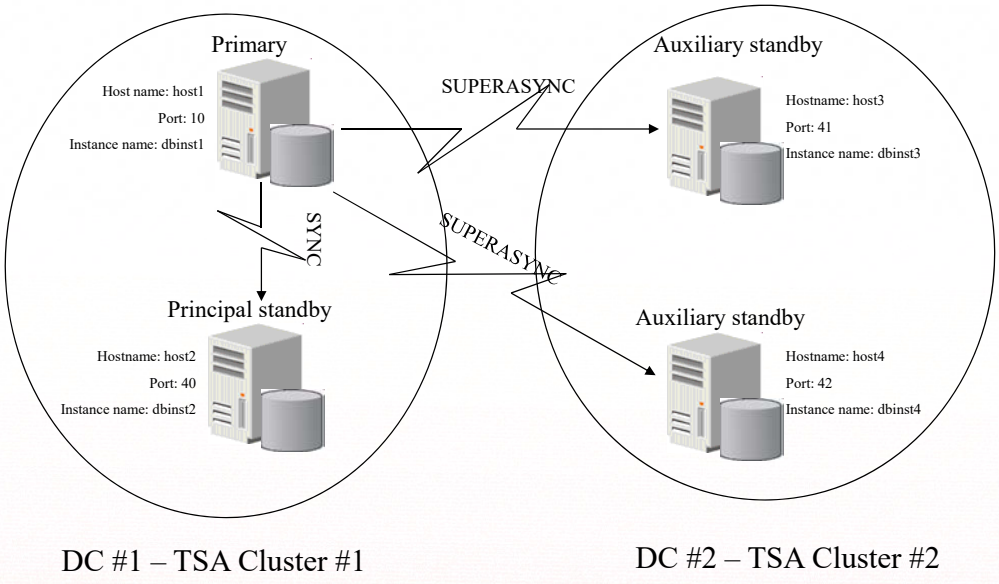
Table of contents

Part 1 : DB2 configuration..... 4
Part 2 : TSA Cluster setup 7
Part 3 : Miscellaneous tasks / Diagnostics..... 11
Part 4 : Remove TSA/HADR configuration 15
Part 5 : Automatic client reroute (ACR) 16

Setting up TSA on Two Data Centers Using HADR with Multiple Standbys

- Can we configure automation for HADR when using Multiple standbys across 2 data centers?
- What if we could, what would it look like?

HADR Dual Data Center Example



Configuring TSA on both sites

- The following procedure will setup TSA in both data centers to handle automated failover. Details of db2haicu usage can be found in the Automated cluster controlled HADR configuration setup using the IBM DB2 high availability instance configuration utility (db2haicu) white paper: https://www.ibm.com/developerworks/community/blogs/nsharma/resource/HADR-db2haicu_v5.pdf?lang=en.
- 1. Run db2haicu on host Host B to create domain containing Host A and B, defines tie breaker device, the network and DB2 instance resources.
- 2. Run db2haicu on host Host A to create the second DB2 instance resource and HADR resource (after this step, HADR failover automation has been enabled in DC1).
- 3. Issue manual graceful takeover on Host C (after this step, Host C is the primary, with Host D as its principal standby, Host A and Host B are the auxiliary standbys).
- 4. Run db2haicu on Host D to create domain containing Host C and D, defines tie breaker device, the network and DB2 instance resources.
- 5. Run db2haicu on Host C to create the second DB2 instance resource and HADR resource (after this step, HADR failover automation has been enabled in DC2).

Configuring TSA on both sites

- 6. Run db2haicu -disable on B.
- 7. Run db2haicu -disable on A.
- 8 a. Part of the initial configuration, the VIP should already be on A, but double check by running lssam on Host A and ensuring the db2ip_xxx_xxx_xxx_xxx-rs resource associated with the database is online on A. If it is instead online on B, then the takeover in step 8b must be run on B.
- 8b. Issue manual graceful takeover on Host A (after this step, Host A is the primary, with Host B as its principal standby, Host C and Host D are the auxiliary standbys).
- 9. Run db2haicu on Host B to re-enable automation.
- 10. Run db2haicu on Host A to re-enable automation.
- Note that the above setup creates two disjoint TSA domains, one in DC1 and the other in DC2. After the above setup, the HADR system is now enabled with the following: Automated failover in DC1, should there be an outage of A, Host B automatically becomes the new primary. This will result in Host B as primary, with Host A the principal standby, and Host C and Host D are still the auxiliary standbys.

Agenda

- Comparison to other DB2 HA offerings
- HADR Review
- HADR Sync Modes
- HADR Log Spooling
- HADR Time Delay
- HADR Multiple Standby
- HADR Configuration
- HADR Monitoring
- Automatic Client Reroute
- Futures

Monitoring HADR Performance

- Two interfaces:
  - MON_GET_HADR table function
    - Available on the primary
    - Available on the standby when reads on standby is enabled
  - db2pd –hadr
    - Available on both the primary and the standby
- The older snapshot interface has been deprecated
- Information for a remote database can be slightly out of date
  - For up to date information, query the source database directly

Changes to monitoring HDAR in v10.5

New behavior to existing fields

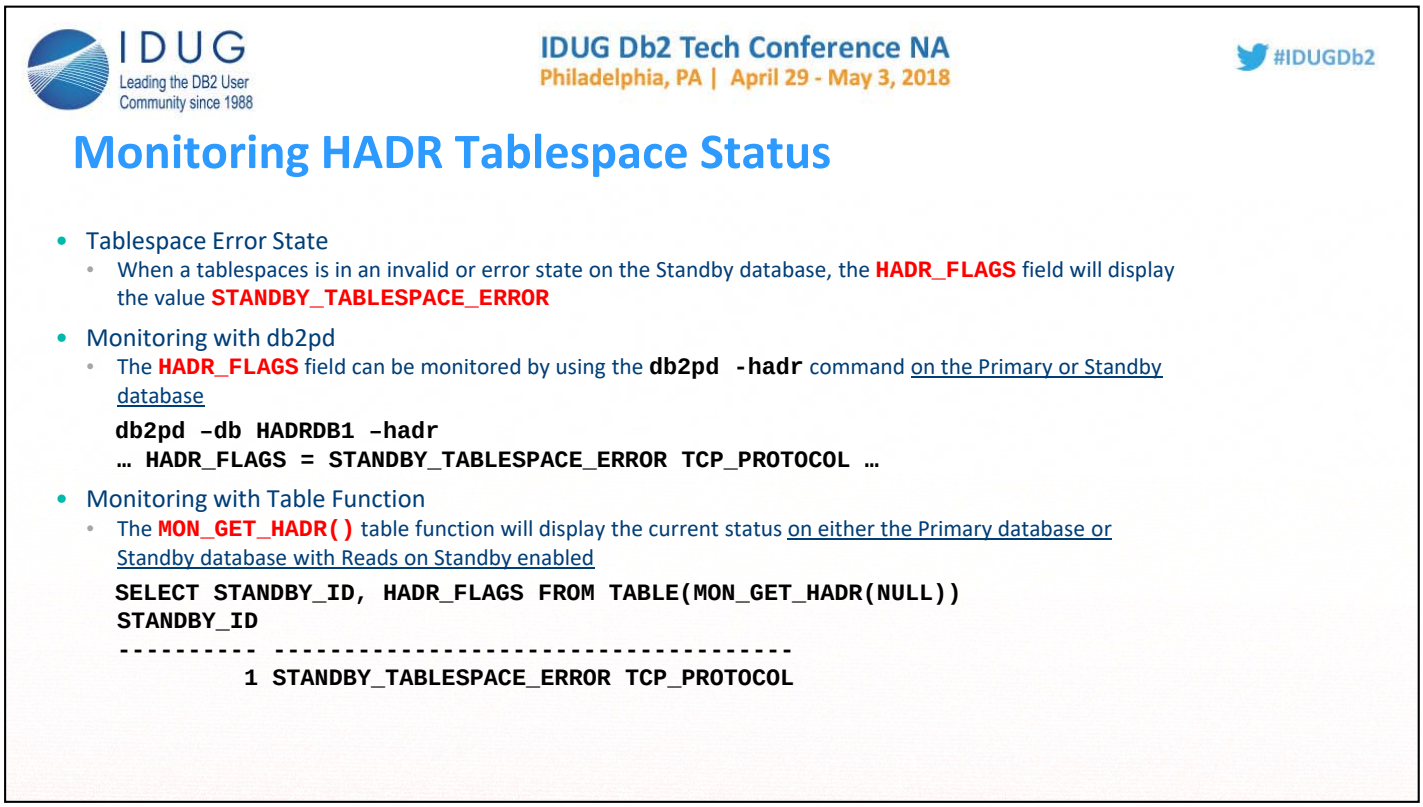
- When HADR_SPOOL_LIMIT is AUTOMATIC (new value in Kepler), the STANDBY_SPOOL_LIMIT monitor field will return the computed spool size in unit of pages, rather than some special value that represents AUTOMATIC
- If a member has never connected to the remote db, remote info such as MEMBER, HOST, INST is unknown. Before some of the info might have been returned as blanks or 0. Now we return NULL.

New monitoring fields

- HEARTBEAT_MISSED (available in fp1)
  - Number of heartbeat messages not received on time on this log stream, accumulated since database startup on the local member
  - Used with HEARTBEAT_EXPECTED to determine network health
- HEARTBEAT_EXPECTED (available in fp1)
  - Number of heartbeat messages expected on this log stream, accumulated since database startup on the local member
  - Used with HEARTBEAT_MISSED to determine network health
- STANDBY_SPOOL_PERCENT
  - Percentage of spool space used, relative to configured spool limit
- STANDBY_ERROR_TIME
  - The most recent time when standby encountered a major error
- HADR_FLAGS
  - ASSISTED_REMOTE_CATCHUP
  - ASSISTED_MEMBER_ACTIVE
  - STANDBY_LOG_RETRIEVAL
  - STANDBY_RECV_BLOCKED
  - STANDBY_LOG_DEVICE_FULL
  - STANDBY_REPLAY_NOT_ON_PREFERRED

Worth Paying Attention To

- HADR_STATE
  - Verify the primary and standby are in connected and in PEER mode (or REMOTE_CATCHUP for superasync sync mode)
- HADR_CONNECT_STATUS
  - SQLM_HADR_CONN_CONNECTED - the good
  - SQLM_HADR_CONN_DISCONNECTED - the bad
  - SQLM_HADR_CONN_CONGESTED - and the ugly
  - If you see congestion, either there are network issues or the standby is not keeping up.
  - Use Spooling if the standby is not keeping up
- LOG_HADR_WAIT_CUR
  - Provides current logger waiting time on an HADR log shipping request
  - Helps identify a primary that is being blocked by log shipping to the standby
  - Should never reach more than 5 seconds
  - Can serve as an early indication of network or standby problems
  - You can cap this using the registry variable DB2_HADR_PEER_WAIT_LIMIT
- HADR_LOG_GAP
  - Use this element to determine the gap between the primary and standby HADR database logs
  - It should be kept to a minimum
  - An increasing number indicates a standby that is not keeping up
- HEARTBEAT_EXPECTED vs HEARTBEAT_MISSED
  - Number of heartbeat messages expected vs those that were not received, accumulated since database startup on the local member
  - Available in v10.5 fp1



HADR Tablespace Recovery

In an HADR environment, when a Standby database has a tablespace in an invalid or error state, the replay of transactions on

this tablespace will stop, while the replay of transactions on other valid tablespaces will continue. The Primary database will not

be affected, and the condition of this tablespace on the Standby may go unnoticed.

If this tablespace condition exists on the Standby database, then sometime later when a TAKEOVER operation is performed on

the Standby database, applications may be impacted by the unavailability of this tablespace. Now the erroneous tablespaces can be recovered on the Standby database by either reinitializing the affected tablespaces, or the entire database.

Starting in v11.1.1.1 and v10.5 Fix Pack 9, when one or more tablespaces is in an invalid or error state on the Standby database, the HADR_FLAGS field will display the value 'STANDBY_TABLESPACE_ERROR'. The HADR_FLAGS field can be monitored by using the "db2pd -hadr" command on the Primary or Standby database, or by using the MON_GET_HADR() table function on the Primary database or the Standby database when the Reads-on-Standby feature is

enabled.

For example:

$db2pd -db HADRDB1 -hadr

HADR_ROLE = PRIMARY

IBM Monitoring and identifying tablespaces in invalid or error state, ... http://www-01.ibm.com/support/docview.wss?uid=swg21993013

1 of 5 2017-06-20, 6:02 PM

REPLAY_TYPE = PHYSICAL

HADR_SYNCMODE = SYNC

STANDBY_ID = 1

LOG_STREAM_ID = 0

HADR_STATE = PEER

HADR_FLAGS = STANDBY_TABLESPACE_ERROR**PRIMARY_INSTANCE = db2inst1**

PRIMARY_MEMBER = 0

STANDBY_MEMBER_HOST = hotellnx119

STANDBY_INSTANCE = db2inst2

STANDBY_MEMBER = 0

HADR_CONNECT_STATUS = CONNECTED

...etc...

$ db2 "SELECT STANDBY_ID, HADR_FLAGS from table(MON_GET_HADR(NULL))"

STANDBY_ID HADR_FLAGS

---------- ----------------------------------

1 STANDBY_TABLESPACE_ERROR **TCP_PROTOCOL**

When this condition occurs, the affected tablespace(s) can be identified on the Standby database by using traditional methods,

such as by examining the 'State' value of "db2pd -tablespaces" output, or the 'tablespace_state

Restore REBUILD in pureScale

Using Data Studio / DSM Alerts

- IBM Data Studio Web Console version 3.1 has enhanced HADR setup and management
  - provides built-in alerts for HADR
  - Allows you to configure your own alerts
- IBM Data Server Manager support
  - https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/W377e39cb6e13_42a7_b752_005b67cb913f/page/Card%20IO%20-%20HADR

To configure data studio alerts:

In the web console, select Open > Alerts and click Health Alerts Configuration.

Select a database for which to view and edit the configurable alert parameters.

Enable or disable database health monitoring for the database. To display alert information for the database in the Health Summary and Alert List, database health monitoring must be enabled. Enable health monitoring by selecting the Monitor database health check box.Note: When you add a database connection, health monitoring is turned on by default. Disable health monitoring by clearing the Monitor database health check box.

Set the refresh rate for the database. The refresh rate controls how often the database is checked for conditions that might trigger alerts.

Configure alerts for the database.

- Select an alert type and click Edit.
- If prompted, sign in with a database user ID that has permission to manage alerts on the database.
- Enable or disable the alert for the database, and configure the critical and warning thresholds for the alert type.

Tools available to measure HADR related performance

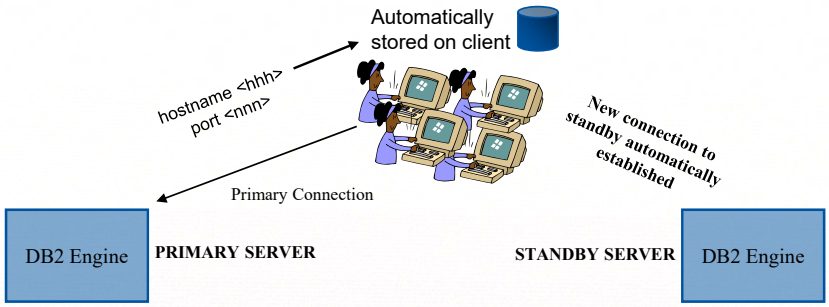
- Check out https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/DB2HADR/page/Perf%20Tuning
- HADR Simulator to measure network and disk speed
- DB2 Log Scanner to analyze database workload
- HADR Calculator to estimate performance of various HADR sync modes

Agenda

- Comparison to other DB2 HA offerings
- HADR Review
- HADR Sync Modes
- HADR Log Spooling
- HADR Time Delay
- HADR Multiple Standby
- HADR Configuration
- HADR Monitoring
- Automatic Client Reroute
- Futures

Automatic Client Reroute

- Automatic, transparent connection to alternate server when primary connection fails
  - If there is a currently executing SQL statement, it will fail with sqlcode -30108
  - Transaction can then be re-driven without re-establishing a connection
- Alternate information Stored on client
  - System database directory
  - alternateDataSource property (Java Type 4 driver)
- Works with HADR, EE/ESE, EEE/DPF, Replication



db2 update alternate server for database <dbname> using hostname <hhh> port <nnn>

Without DB2 Automatic Client Reroute Enabled

- During a server or database failure
  - **the current in-flight transaction will be rolled back**
  - Application receives SQLCODE:
    - -30081, -1224, -1776
    - java.sql.SQLException SQLCODE -4499 for JDBC and SQLJ clients
  - Application must
    - Re-establish connection
    - Setup environment
      - Create statement, prepare statement
    - Resubmit statement or transaction

With DB2 Automatic Client Reroute Enabled

- During a server or database failure
  - **the current in-flight transaction will be rolled back**
  - Application receives connections re-established SQLCODE:
    - -30108
    - java.sql.SQLException SQLCODE -4498 for JDBC and SQLJ clients
  - Application connection is automatically re-established and environment maintained
    - Do not need to retry connect, create statement or prepare statement
  - Application need only resubmit the statement or transaction
    - Similar behavior to application getting lock timeout or deadlock sql error, -911 (ie. treat the event as a rollback)

Considerations for WAS environments

- Ensure pool purge policy is set to entire pool
  - Facilitates entire pool getting re-established upon a communication error or reroute message
  - Reduces subsequent error paths and notifications
    - Which can occur later then when the original failure occurred

Client Reroute Enhancements in DB2 9.5 FP1

- Seamless failover for JAVA applications
  - Suppress the SQLException with SQLCODE -4498
  - Driver re-issues the SQL statement that was being processed when the original connection failed

  - Conditions required for seamless failover to occur:
    - Set enableSeamlessFailover datasource property to True
    - The connection was not being used for a transaction at the time of failure
      - The failure must occur when the first SQL statement in the transaction is executed
    - There is no outstanding global resources in use
      - global temporary tables
      - open, held cursors

Considerations for WAS environments

- With enableSeamlessFailover enabled
  - pool purge policy setting of "**failing connections only**"
    - only purge the failing connection out of the pool.
  - connections in the WAS in-use pool but that are not in a UOW will failover on next touch seamlessly (ie. no -4498 is throw)
  - connections in the WAS free pool will failover on first touch seamlessly
  - connections in the WAS in-use pool and that are in a UOW will get the -4498 (connections re-established) – other pool connections remain
  - any connections that have outstanding global resources that can't be restored will receive the -4498 – other pool connections remain

New Features to Version 11.1

11.1.0.0

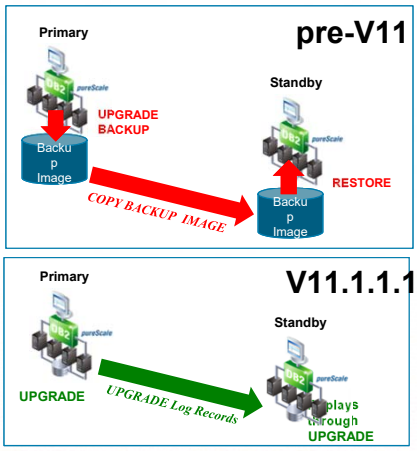
- **Fast pre-allocation for log file creation and resize**
  - DB2_USE_FAST_LOG_PREALLOCATION
- **Backup database has a new option**
  - NO TABLESPACE
- **pureScale HADR sync/nearsync support**
- **HADR upgrade with no standby re-initialization from 10.5fp7 (non-pureScale)**
- **Recovery through migration from 10.5fp7**
- **Db2 backup and log compression on POWER 7+/ 8 using NX842**
  - compress comprlib libdb2nx842.a
  - DB2_BCKP_COMPRESSION

11.1.1.1

- **Increase of limit on LOGFILSIZ (to 64GB)**
- **HADR upgrade with no standby re-initialization from 10.5fp9 (pureScale)**

Seamless HADR Upgrades

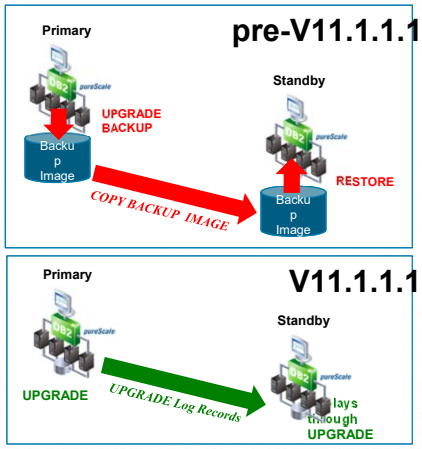
- HADR environments can now be upgraded without re-initializing the standby.
- Standby database will 'replay' the upgrade done on the primary by rolling forward through the update log records
- **Non-pS: needs v11.1.0.0 and 10.5 FP7**
- **pS: needs v11.1.1.1 and 10.5 FP9**
- More details in "Upgrading DB2 servers in HADR pureScale environments (without standby reinitialization)" in the V11 knowledge center



Seamless HADR Upgrades Extended

Procedure Overview

1. **PRIMARY :**
   DEACTIVATE database; DB2STOP instance
   Upgrade the instance using db2iupgrade
   - db2iupgrade invokes db2ckupgrade which will check to ensure primary and standby log positions match
   - db2ckupgrade will not issue STOP HADR or change role
2. **STANDBY :**
   DEACTIVATE database; DB2STOP instance
   Upgrade the instance using db2iupgrade
   - db2ckupgrade will no longer fail 'because it is an HADR standby'
3. **STANDBY :**
   UPGRADE the database
   - Returns SQL1103W – this indicates that the standby is now waiting for log data from a subsequent UPGRADE issued on the primary
   - No connections can be made to a standby database in this state
   - Can be monitored via db2pd –hadr
4. **PRIMARY :**
   UPGRADE the database
   - Will ship log data to standby; Standby replays these log records
5. **PRIMARY :**
   ACTIVATE the database
   - Now primary and standby can resume normal operations



New Features to Version 11.1

11.1.2.2

- **Databases can now be configured to allow connectivity during crash recovery**
  - DB2_ONLINERECOVERY
- **Database restore rebuild supported in pureScale**
- **An HADR Standby tablespace can now be recovered without a full database re-initialization**
- **New STANDBY_TABLESPACE_ERROR flag for HADR_FLAGS monitor element**
- **Databases can now be configured to avoid lock-escalation**
  - DB2_AVOID_LOCK_ESCALATION
- **Faster pureScale member crash recovery**
- **Crash recovery and rollforward replay performance improvements**

Streamlined HADR Table Space Recovery

We have recently significantly streamlined the act of recovering from a table space error state on the HADR standby:

Prior to V11.1.0.0:

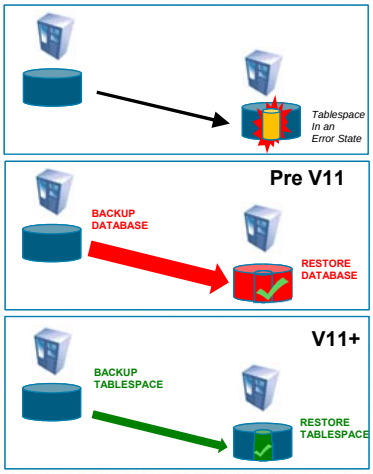
- the entire Standby database needed to be restored on the standby

V11.1.0.0 and V11.1.1.1 (single node) & V11.1.2.2 (single node and pureScale) :

- only the affected tablespace(s) need be restored on the standby

To monitor if restores are required, use db2pd:

https://www-01.ibm.com/support/docview.wss?uid=swg21993013 db2pd -db HADRDB1 -hadr



72

Important notes:

- As before, these recovery operations should occur before any TAKEOVER is issued
- Further details: http://www-01.ibm.com/support/docview.wss?uid=swg21993389

Online Crash Recovery (aka Async UNDO)

Goal:	Significantly increase database availability during crash recovery (*with particular focus on scenarios where large batch operations were updating the database*)
How:	Allow database connections during the **UNDO** phase of crash recovery(also applicable to **HADR TAKEOVER BY FORCE**)

Details

Crash recovery has 2 phases: REDO and UNDO

- *The **UNDO** phase is typically significantly longer if a crash occurred when batch operations or other long-running transactions were executing*

With Online Crash Recovery, DB2 will …

- *Acquire exclusive locks on all tables or table partitions with in-flight operations immediately after the **REDO** phase*
- *Allow new database connections as early as possible in the **UNDO** phase:*
  - *Allow access to unlocked tables/partitions to proceed normally*
  - *Allow UR access to locked tables/partitions*
- *Process the **UNDO** phase asynchronously, releasing table locks as **UNDO** progresses*

73

Enabling Online Crash Recovery

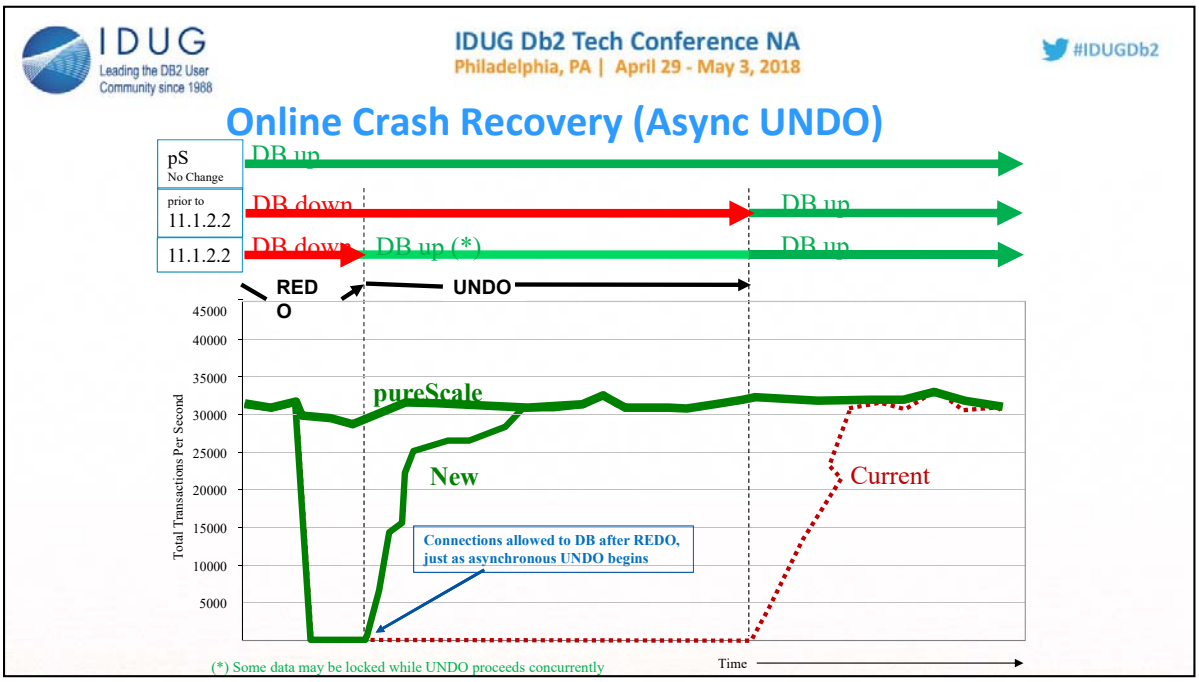
Assign the value **YES** to this registry variable:

```
db2set DB2_ONLINERECOVERY=YES
```

Restarting the instance is not necessary (the registry variable value active at the start of crash recovery is used)

Note *For more detail, see **Database accessibility during Backward phase of crash-recovery or HADR Takeover by Force** at: http://www-01.ibm.com/support/docview.wss?uid=swg21994342*

Note *This feature was present as a technical preview (for non-production use) in V11.1.1.1, and was enabled via an undocumented registry variable, DB2_ASYNC_UNDO=YES. The feature is now available for production use in V11.1.2.2, and enabled via DB2_ONLINERECOVERY=YES.*



This graph shows the availability characteristics of a Db2 database (with pureScale, non-pureScale prior to 11.1.2.2, and now non-pureScale in 11.1.2.2) following a crash – when Db2 is performing crash recovery.

We'll focus on the non-pureScale example here, as pureScale had already been designed to offer maximum database and data availability in case of a crash.

Db2 has a REDO and an UNDO phase of crash recovery. REDO ensures that all activities that had occurred prior to the crash (for log records that were written to the log files – for example, committed transaction activity) are replayed and in the database as if the crash had not occurred. The UNDO pass then rolls back all uncommitted transactions that were in flight at the time of the crash. Connections to the database are not allowed until both the REDO and UNDO phases of crash recovery are completed.

Now in Db2 11.1.2.2, Db2 will **allow connections to the database after the REDO phase and during/while the UNDO phase** executes. Data (tables) that have uncommitted activity to be undone will be locked, but access to all other tables will be allowed. This is of most benefit in batch/ETL processing environments where a large batch job which inserted/updated/deleted many many rows will take some time to undo – this should not prevent applications from connecting to the database while the large and uncommitted transaction is being rolled back in the crash recovery UNDO phase.

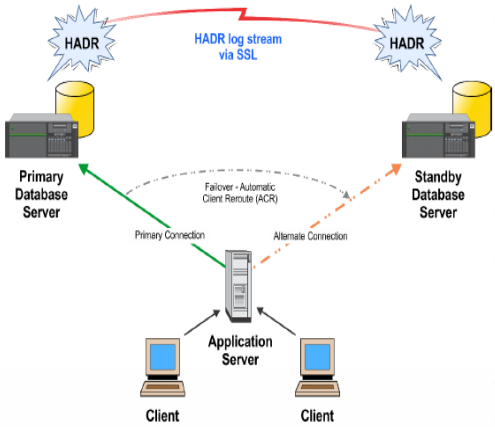
New Features to Version 11.1

11.1.3.3

- **HADR integrity checking of transaction log data during network transmission between the primary and standby servers is improved**
  - When an integrity check failure is detected, seamless retransmission of the log data is performed to auto-correct the condition, with no user intervention required
- **HADR Reads On Standby (ROS) diagnostic improvements allow for easier identification of operations which cause replay-only windows**
  - DB2_HADR_REPLAY_ONLY_WINDOW_DIAGLEVEL
- **The archival of log files using VENDOR or TSM methods can now be configured with a timeout on Unix environments**
  - LOGARCHOPT1/2: --VENDOR_ARCHIVE_TIMEOUT
- **SSL support for the transmission of transaction log data between the HADR primary and standby database servers on all platforms, excluding pureScale**
- **CREATE INDEX operations in a Db2 pureScale environment can now allow concurrent write access to the table**
  - DB2_INDEX_CREATE_ALLOW_WRITE

SSL Encryption Between HADR Primary and Standbys

- Provides integrated protection of sensitive data in the log stream
- Enabled via the **HADR_SSL_LABEL** database configuration parameter
- Supports all HADR synchronization modes
- Supports multiple standbys
- SSL Encryption for non-PS HADR (all platforms as of 11.1.3.3)




```
db update database configuration for <database_name> using HADR_SSL_LABEL <SSL_certificate_label_name>
```

Export of HADR TSAMP config to XML File

- db2haicu has a new "-o" option
  - Used to facilitate rapid HA configuration backup for fast redeployment / rebuild.
  - Supported for HADR, DPF, and single-partitioned (non-DPF) environment.
    - Run db2haicu to create XML file,

```
Original_system>  db2haicu -o primary.xml
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu). db2haicu determined the current DB2 database
manager instance is 'db2inst1'. The cluster configuration that applies to this instance will be exported. All cluster
configurations have been exported successfully. db2haicu exiting.
```

    - edit XML file for correct local/remoteHost

If the TSAMP configuration exported from the primary host is used to import the configuration on both primary and standby hosts, the following fields for the HADRDBSet element will depend on the host where import is being performed. The localhost must refer to the host on which the import is being run and remote who it connects to.

<HADRDBSet><HADRDB databaseName="SAMPLE" localInstance="amahadev" remoteInstance="amahadev" localHost="ceha04" remoteHost="ceha03" monitorDataMounts="false"/> <VirtualIPAddress baseAddress="9.26.54.119" subnetMask="255.255.255.0" networkName="db2_public_network_0"/> </HADRDBSet>

    - run db2haicu on new system(s), re-editing as needed.

```
new_system>  db2haicu -f primary.xml
```

Agenda

- Comparison to other DB2 HA offerings
- HADR Review
- HADR Sync Modes
- HADR Log Spooling
- HADR Time Delay
- HADR Multiple Standby
- HADR Configuration
- HADR Monitoring
- Automatic Client Reroute
- Futures

Future directions (This is NOT a commitment that any of these items will ever be delivered)!

- Enhanced (usable) Read On Standby
  - DDL changes will only affect queries accessing the affected tables
  - Changes to system catalogs will no longer kick off readers
  - How much are you willing to pay for this?
- Monitoring the progress and performance of backup and restore operations can be achieved using the new db2pd -barstats option
- Backup on Standby
  - How should this behave?
    - Issue on Primary?
    - Issue on Standby?
    - Pause Replay during backup?
- Maintain the configuration parameter settings on all standbys?
- Rollback performance improvements using buffered I/O when reading transaction log file data

IDUG
Leading the DB2 User Community since 1988

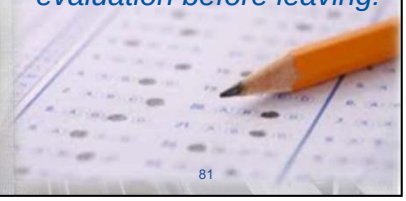
IDUG Db2 Tech Conference NA
Philadelphia, PA | April 29 - May 3, 2018

#IDUGDb2

Dale McInnis
IBM Canada Ltd.
dmcinnis@ca.ibm.com

Session code: D12

Please fill out your session evaluation before leaving!



81