IBM Software Group

DB2 HADR Deployment

IBM DB2 Beaverton Lab

Yuke Zhuge（诸葛宇珂）
zhuge@us.ibm.com

Sept. 2011

DB2 Information Management Software

e business on demand software

Blue Pearl
Format

Where is the lab?

- DB2 LUW Labs in North America

Map source: World Fact book

Beaverton is Portland

Above: Portland downtown (source: Portland Oregon Visitor's Association. Free use)

Right: IBM Beaverton office (source: w3.beaverton.ibm.com)

Toronto lab

IBM Toronto Lab - From 55 people in 1967 to 2500 professionals with international mandates today.

Source: w3.torolab.ibm.com

DB2 Global Team

- Canada
- US
- China
- India
- Ireland
- Brazil

Agenda

- Introduction
- Setup
- Role switch and Failover
- HADR State Transition
- Synchronization Modes
- Tuning
- Rolling Update
- Integration with Cluster Manager
- Automatic Client Reroute
- Reads on Standby
- Monitoring and Administration
- Resources

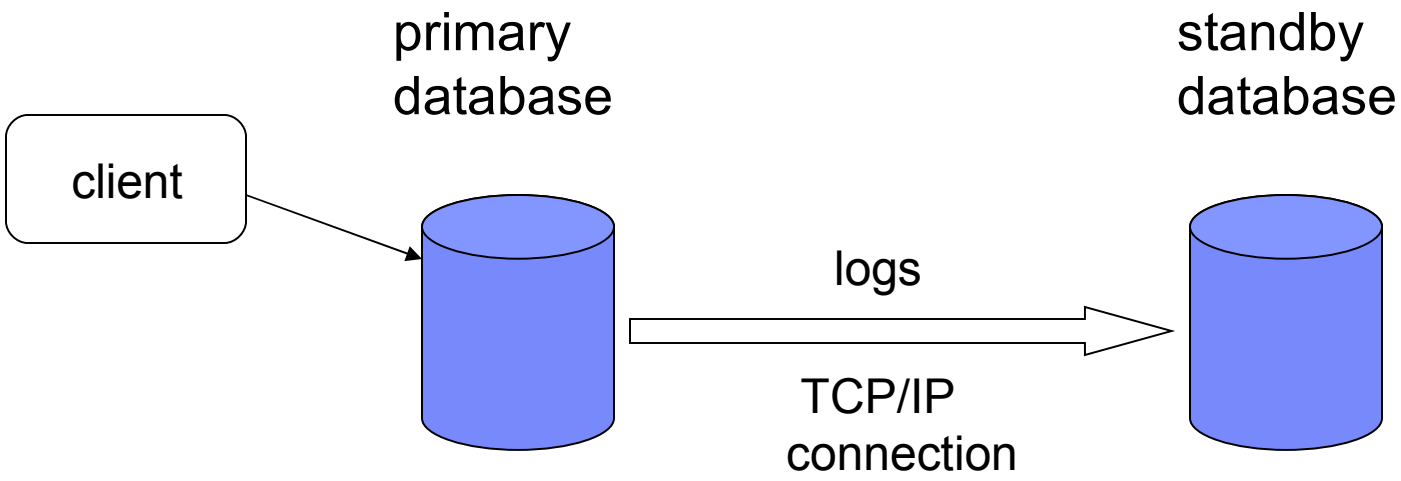
What is HADR?

- High Availability Disaster Recovery

- Disaster recovery via data replication.
- Data from primary database replicated to standby database in real time.
- If primary fails, standby takes over as the new primary.

How is data replicated?

- Replication is done by log shipping
- Whole database is replicated (easy administration)



No special hardware or software needed, just standard TCP

What's replicated, what's not?

- Logged operations are replicated
  - Example: DDL, DML, table space and buffer pool creation/deletion, XML, MDC, etc.
  - Logged LOBs are replicated.
- Not logged operations are not replicated.
  - Example: database configuration parameter. not logged initially table, not logged LOBs, UDF libraries.

Features

- Ultra-fast failover (HA)
- Easy administration
- Handling of site failures (DR)
- High performance, supports high throughput systems.
- Negligible impact on performance
- Configurable degree of consistency (sync modes)
- Upgrades without interruption (rolling update)
- Very easy integration with HA-software
  - Integrated with TSA
- Transparent failover and failback for applications
  - Automatic client reroute,
  - Virtual IP.

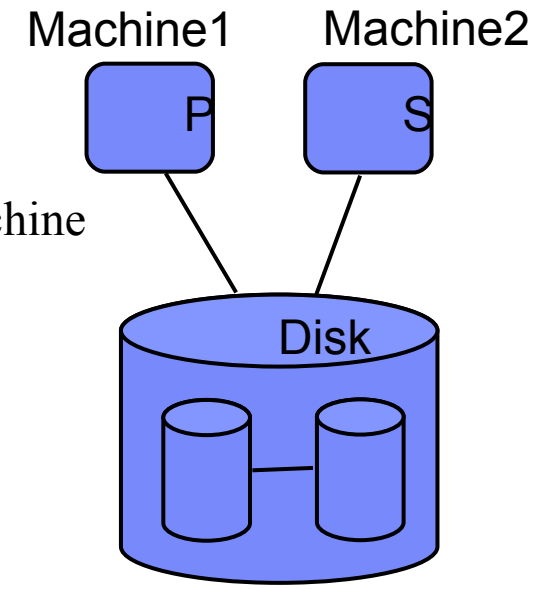
History

- Built with Informix HDR technology.
- First released on DB2 V8.2 in 2004.
- Collaboration of Portland and Toronto lab.
- Enhancements:
  - Peer window released on DB2 V9.5
  - Integrated with cluster manager in V95
  - Reads on Standby on DB2 V9.7 FP1
  - SuperAsync mode on V9.5 FP8 and V97 FP5
  - More to come.

When to use HADR

- High Availability
  - Primary and standby at same site.
  - No crash recovery. Standby already online
    - Faster than restarting database on same or backup machine.
  - Ultra fast failover
    - Sub-minute failover
- Disaster recovery
  - Primary and standby at different sites
  - Protect against total loss of primary site

Alternatives



- Shared disk HA cluster (spare machine)
  - When one machine fails, start database on another machine
  - Restart can be slow.
- RAID or disk mirroring (spare disk)
  - Protects against disk failure
  - Does not work well for remote standby site
    - Complex and slow. Need to replicate all database storage, including tablespaces and preserve write order
    - Special hardware and software often needed.
    - HADR only replicates delta (database log)

Solutions outside of the database cannot keep the standby database online. Even if storage is replicated, database restart on standby site is needed.

Alternatives (continued)

- SQL based replication:
  Data propagator, Q rep., Data mirror
  - Performance
    - Overhead of reconstructing SQL and processing SQL.
  - Async only. Not good for OLTP systems.
  - More complex to setup and admin

HADR Setup

2 ways to create standby

- Backup and restore
  - Backup on primary database (online or offline)
  - Restore onto standby machine
    - Do not use “without rolling forward” option on restore
- Split mirror (or physical copy) from primary
  - Run “db2inidb *dbalias* as standby” on the copy

Info Center Topic: Initializing high availability disaster recovery (HADR)

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.ha.doc/doc/t0011725.html

HADR Setup (continued)

- Update config on both databases

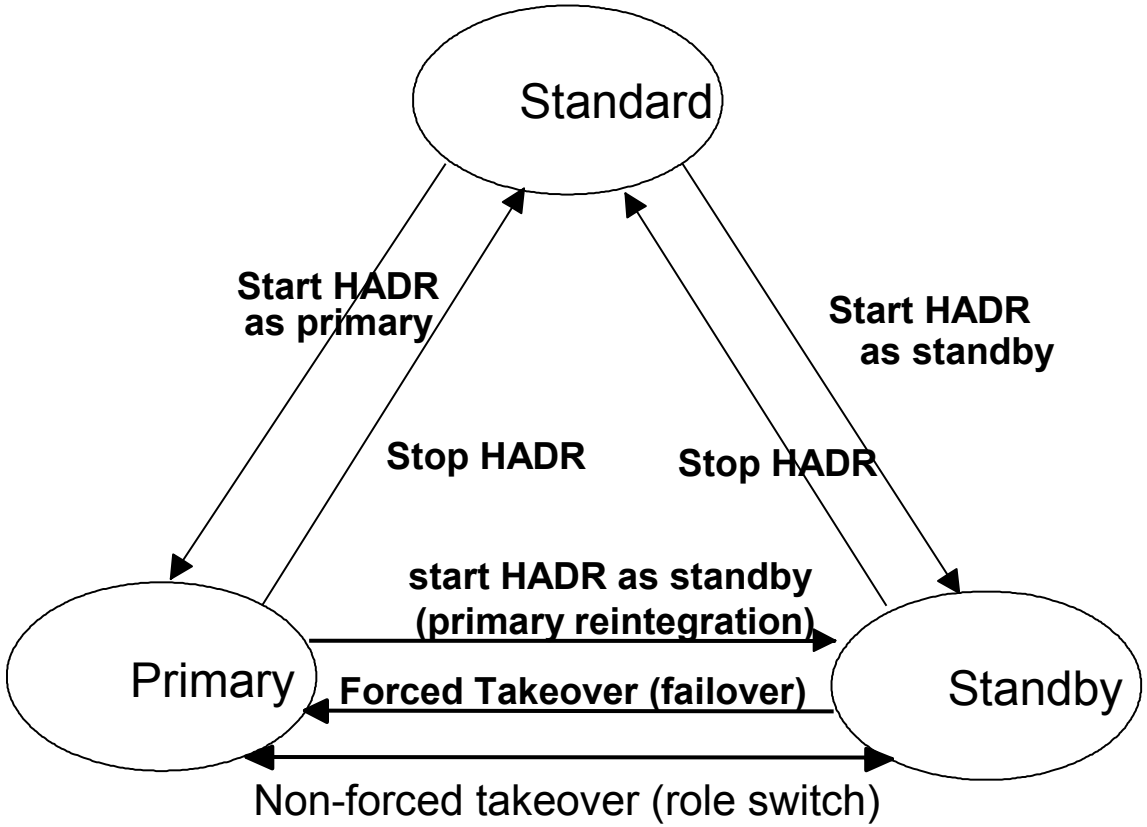
```
HADR local host name                      (HADR_LOCAL_HOST) =
 host1.ibm.com
HADR local service name                    (HADR_LOCAL_SVC) = hadr_port1
HADR remote host name                    (HADR_REMOTE_HOST) =
 host2.ibm.com
HADR remote service name                  (HADR_REMOTE_SVC) = hadr_port2
HADR instance name of remote server  (HADR_REMOTE_INST) = db2inst2
HADR timeout value                           (HADR_TIMEOUT) = 120
HADR log write synchronization mode     (HADR_SYNCMODE) = SYNC
```

- Start HADR
  - "Start HADR … as standby" on standby
    - Start standby first, or start primary first, then start standby within HADR_TIMEOUT
  - "Start HADR … as primary" on primary

Easy to use: Only 3 HADR commands: start/stop/takeover HADR

HADR commands

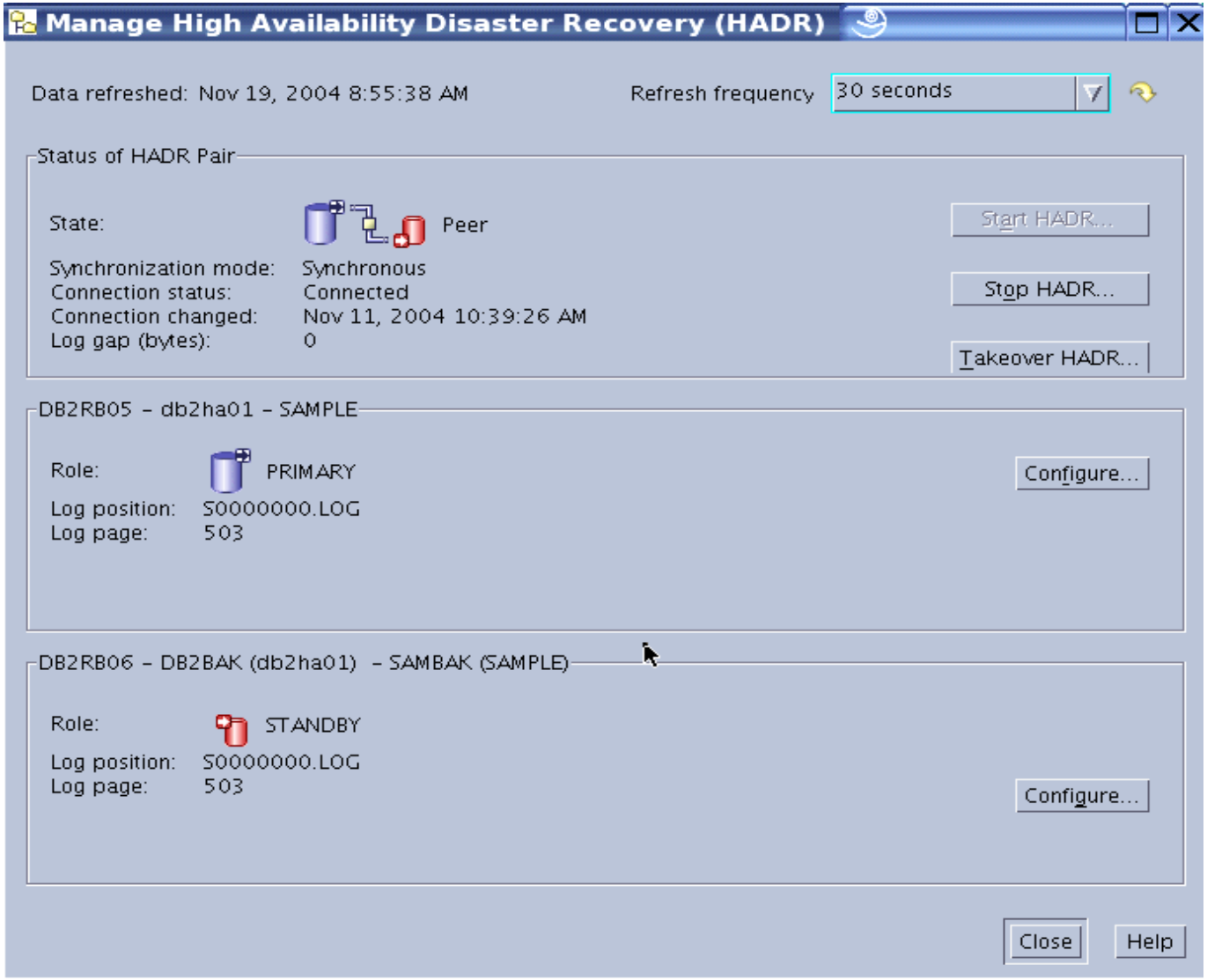
- 3 commands: Start, stop, takeover HADR



Role switch and Failover

- Role switch
  - Issue “Takeover HADR” on standby
  - Standby becomes primary
  - Primary becomes standby
- Failover
  - Issue “Takeover HADR … by force” on standby
  - Standby becomes primary
  - Primary is disabled if connected to standby.
- Primary reintegration
  - Old primary can be converted to standby of the new primary if primary and standby log streams did not diverge (no data loss on failover)

GUI: Control Center



HADR State Transition

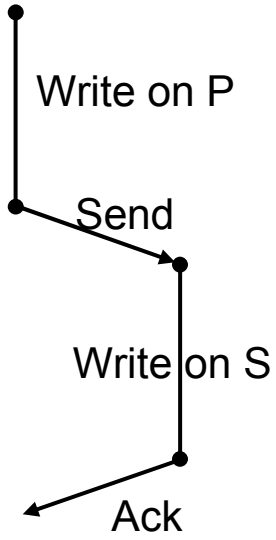
- Local catchup
  - After startup, standby first tries to read logs from local source.
  - Reads from log path, overflow path, and archive
- Remote catchup pending
  - After local catchup reaches local end of log, if there is no connection to primary, standby waits in this state.
- Remote catchup
  - Standby retrieves on-disk or archived logs from primary via HADR connection
  - Relatively “old” log
- Peer
  - Standby retrieves in-memory logs from primary. Primary sends a copy to standby when it writes logs to local disk.
  - Current log.

Synchronization Modes

- SYNC
- NEARSYNC
- ASYNC
- SUPERASYNC

SYNC mode

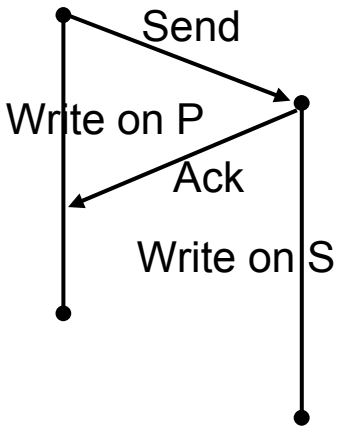
In peer state (Primary sends logs to standby when it writes logs):



- Transactions on primary will commit only after logs have been written to disk on both primary and standby.
- Maximal protection, with performance cost.
- After writing logs to local disk, primary sends a copy to standby. Primary will then wait for “log written” ack message from standby.
- Serial write and send on primary
- **In peer state,** any transaction committed on primary is guaranteed to have committed on standby too.
- **In peer state,** if a failover occurs, you will not lose any committed transaction.

NEARSYNC mode

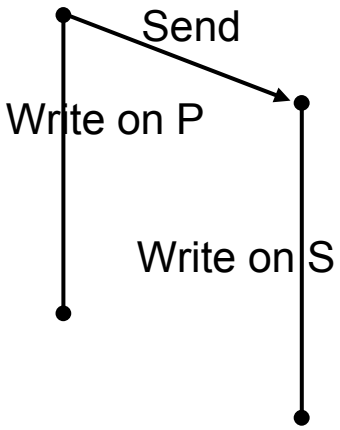
In peer state (Primary sends logs to standby when it writes logs):



- Transactions on primary will commit only after logs have been written to disk on primary and received into memory on standby.
- Protection nearly as good as SYNC mode. Performance is better than SYNC mode.
- **In peer state,** you will lose data only if both primary and standby fail at the same time.
- When writing logs to local disk, primary also sends a copy to standby. Primary will then wait for “log received” ack message from standby.
- Parallel write and send on primary

ASYNC mode

In peer state (Primary sends logs to standby when it writes logs):



- Transactions on primary will commit only after logs have been written to local disk and sent to standby
- Best performance, least protection.
- When writing logs to local disk, primary also sends a copy to standby. Primary will go on as soon as send() call to TCP returns.
- **In peer state,** any transaction committed on primary is guaranteed to have been "sent" to standby.
- **In peer state,** if a failover occurs, logs sent but not yet received can be lost.

SUPERASYNC mode

- New in V95fp8 and V97fp5
- HADR pair does not enter peer state.
  - Log shipping only happens in remote catchup state
- Transaction commit on primary has no dependency on replication.
- Slow network or standby will not slow down primary
- But standby can fall behind
  - Monitor log gap closely.
- Role switch allowed in remote catchup state
  - Only allowed in peer state in other sync modes.
  - Check log gap before issuing takeover command
  - Takeover will stop transactions on primary, ship all logs and finish replay. No data loss.
  - Large gap will result in long takeover time
- Larger data loss in failover when log gap is larger.

Even stronger protection: Peer Window

- Database config parameter HADR_PEER_WINDOW
- Primary blocks transactions for this amount of seconds if it loses connection to the standby in peer state.
  - Only applicable to Sync and NearSync modes
- Default value 0: no wait.
  - If network fails, primary resumes transactions immediately. If primary fails a short time later and failover is performed. These transaction will not show up on new primary.
- Tuning: How important is the 2nd copy? (0 to infinity)
- "Takeover by force … peer window only"
  - Perform takeover only in peer window. Guarantee no data loss.
  - Makes auto failover safe.

Synchronization Modes (continued)

- What is your business requirement?
  - How much is your tolerance to data loss?
- Is my network fast enough for this sync mode?
  - What is your logging rate?
  - Run HADR simulator to test network
    - http://www.ibm.com/developerworks/wikis/display/data/HADR_sim
    - Generates realistic workload on network
    - Reports estimated logging throughput in various sync modes and TCP configuration.
    - No DB2 instance needed.
- Sync/Nearsync often used on LAN
- Async/SuperAsync often used on WAN

TCP Tuning

- Recommended TCP window size:
  - sendRate * roundTripTime
    - Consider: bestSendRate * worstRoundTripTime
  - System default may not be optimal.
  - Smaller buffer may not make full use of network bandwidth.
- Set by DB2 registry variable
  - DB2_HADR_SOSNDBUF and DB2_HADR_SORCVBUF
  - Recommendation: Find optimal value using HADR simulator. The deploy to database.

HADR Tunables

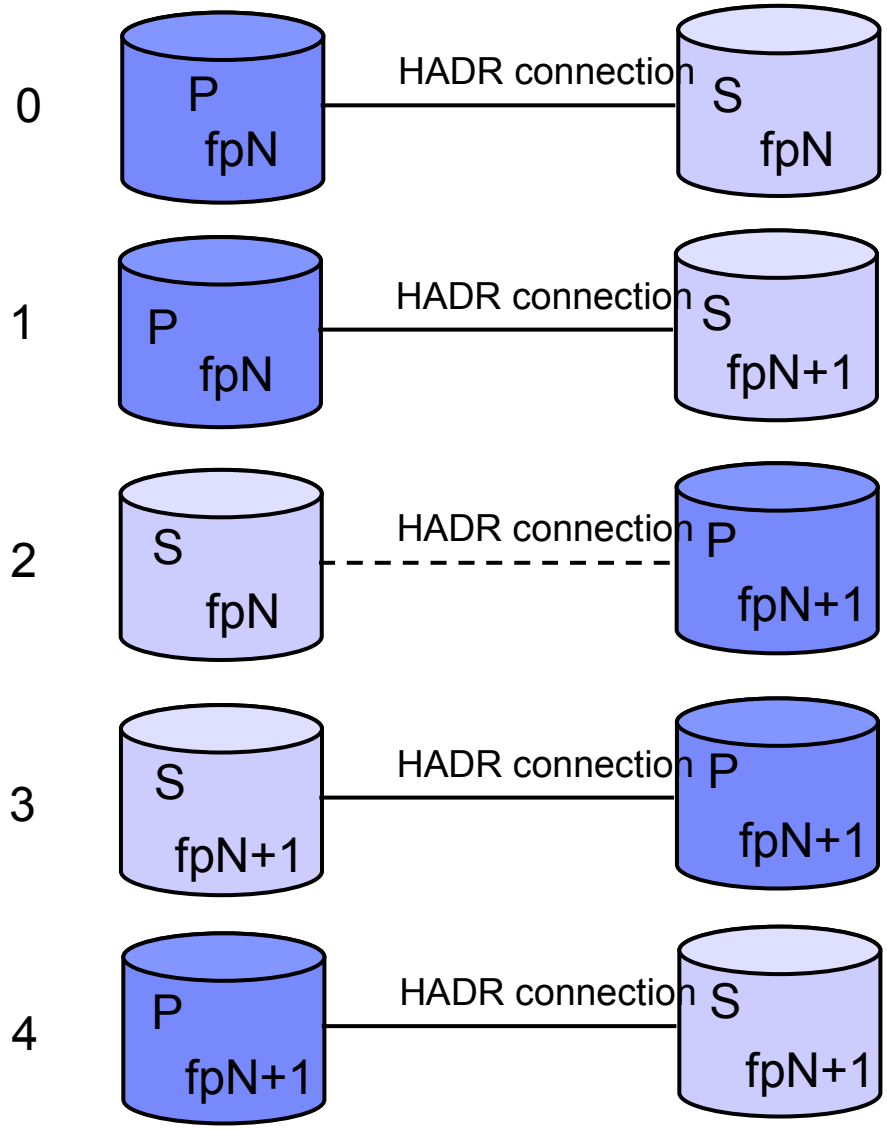
- HADR_PEER_WAIT_LIMIT (registry variable)
  - Wait limit for peer state log replication.
  - Default 0, meaning no limit.
  - Handles slow network
    - Cannot send out data, or ack message delayed.
  - Handles slow standby
    - Slow replay on standby causes “receive buffer full”. Standby cannot receive more logs
    - Async mode: Primary sees “congestion”
    - Sync and nearsync mode: Primary may or may not see congestion. May send out a flush, then wait for ack.
  - “Large” operations slow down standby: Reorg, load, etc.

HADR Tunables

- DB2_HADR_BUF_SIZE (registry variable)
  - Standby receive buffer size
  - Default 2x primary log write buffer.
  - Temporarily buffers primary load spike.
    - Recommends holding at least 10 seconds of primary log
  - Won't help with sustained throughput.
  - Monitor usage percentage via "db2pd –hadr" on standby
    - 100% means standby replay is slow
    - 0% means network is slow.
- More tuning on
  - http://www.ibm.com/developerworks/wikis/display/data/HADR_tune

Rolling Update

No downtime during update.
Also used for OS and hardware upgrade



Both P and S on fpN

Shutdown S. Upgrade S, Restart S,
S on fpN+1

Takeover.
P on fpN+1
Connection is closed at end of takeover

Shutdown S. Upgrade S, Restart S,
Both on fpN+1

Optional 2nd takeover
Both on fpN+1

Integration with Cluster Manager

- TSA bundled with DB2
- Use “db2haicu” tool to setup
- No more scripting
  - DB2 informs TSA of state changes such as activation/deactivation, HADR takeover.
- Auto failover supported
  - Using peer window. Guaranteed no data loss

Automatic Client Reroute

- Set primary and standby as alternate server of each other.
  - Client auto rerouted to new primary in failover
  - Application gets a specific SQL code informing it that it has been rerouted to the new primary.
  - Application does not need to have reconnection logic
    - Application does not need to know where the new primary is.
  - Transaction context is not carried over to new primary
    - Application usually can just resubmit the transaction.

Monitoring HADR

- Monitoring interfaces
  - DB2 CLP snapshot
  - db2pd
  - monitor API
- Current role of a database
  - Read-only database config parameter

    ```
    HADR database role = PRIMARY|STANDBY|STANDARD
    ```
  - Works even when database is offline

CLP snapshot

- CLP snapshot: db2 get snapshot for db on <dbAlias>

```
HADR Status
  Role                       = Primary
  State                      = Peer
  Synchronization mode       = Sync
  Connection status          = Connected, 09/05/2006 14:05:09.002221
  Heartbeats missed          = 0
  Local host                 = server1.ibm.com
  Local service              = hdr_svc1
  Remote host                = server2.ibm.com
  Remote service             = hdr_svc2
  Remote instance            = accounting
  timeout(seconds)           = 100
  Primary log position(file, page, LSN) = S0000833.LOG, 0, 0000000001CA4000
  Standby log position(file, page, LSN) = S0000832.LOG, 3, 0000000001CA3A77
  Log gap running average(bytes) = 1412
```

- LSN: Log serial number. Byte offset in log stream.

db2pd -hadr

- db2pd -hadr -db <dbAlias>

```
Database Partition 0 -- Database HADRDB -- Active -- Up 0 days 00:19:25

HADR Information:
Role    State                SyncMode HeartBeatsMissed   LogGapRunAvg (bytes)
Primary Peer                 Sync     0                  1412

ConnectStatus ConnectTime                               Timeout
Connected     Mon Sep 05 14:05:09 2006 (1158008709)     8

LocalHost                                 LocalService
server1.ibm.com                           hdr_svc1

RemoteHost                                RemoteService      RemoteInstance
server2.ibm.com                           hdr_svc2           accounting

PrimaryFile  PrimaryPg  PrimaryLSN
S0000833.LOG 0          0x0000000001CA4000

StandByFile  StandByPg  StandByLSN
S0000832.LOG 3          0x0000000001CA3A77
```

Reads on standby

- Available on DB2 V97 fp1 and later.
- Read only queries on standby
  - Disabled by default. Enable via registry variable DB2_HADR_ROS
  - Only Uncommitted Read (UR) isolation level supported
  - Set registry variable DB2_STANDBY_ISO=UR to suppress errors when other levels are requested
  - Write operations get error
- Great for reporting and analytical workload
- Offloads the primary.

HADR Admin

- Monitor HADR status
- Check standby tablespace status regularly
  - "db2pd -tablespaces -db <dbName>“
  - Replay error can bring a tablespace offline
    - Disk full, I/O error, etc.
    - Subsequent log records on the tablespace ignored.
- Use Reads on standby to check standby data content
- Test role switch and failover periodically

Resources

- DB2 Info Center, High Availability Topic
  - http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.admin.ha.doc/doc/c0006354.html
- IBM developer works HADR wiki
  - http://www.ibm.com/developerworks/wikis/display/data/HADR_home
- DB2 best practice white papers
  - http://www.ibm.com/developerworks/data/bestpractices/db2luw/ Look for "DB2 High Availability Disaster Recovery"
- Search “HADR” on IBM software group library
  - http://www-306.ibm.com/software/data/pubs/papers/
- Search “HADR” on IBM developer works
  - http://www-128.ibm.com/developerworks/

- Backup Slides

Block Non Logged

- DB config: BLOCKNONLOGGED
- Disallow "not logged initially“ tables and not logged LOB columns, either by create table or alter table statement.
- Prevents new non logged tables and columns.
- Existing non logged tables and columns are not affected.
  - Check existing non logged tables and columns, remove or modify as necessary.

Indexes

- Database configuration parameter `LOGINDEXBUILD`
  - Whether to log index build or not. Controls index creation, recreation and reorg.
  - Index update is always logged.
  - If log replay encounters an unlogged index creation, the index will be marked as invalid.
  - The index will have to be rebuilt later.
  - Default is OFF. Recommend ON for HADR databases.
  - Table level attribute overrides database level config.
    - Can be set by “alter table” statement.
    - No option available on “create table” statement.

Indexes (continued)

- Configuration parameter `INDEXREC`
  - At both database manager and database level.
  - SYSTEM: Database level only. Use database manager configuration. Default for database config.
  - RESTART: Rebuild invalid indexes on database restart or HADR takeover. Default for database manager.
    - Index rebuild at takeover time is asynchronous.
    - Index rebuild at restart time is synchronous.
  - RESTART_NO_REDO: Same as RESTART, except that the rebuild will not be replayed during rollforward or HADR standby replay.
  - ACCESS: Rebuild invalid indexes on first access of the table.
  - ACCESS_NO_REDO: Same as ACCESS, except that the rebuild will not be replayed during rollforward or HADR standby replay.

Load

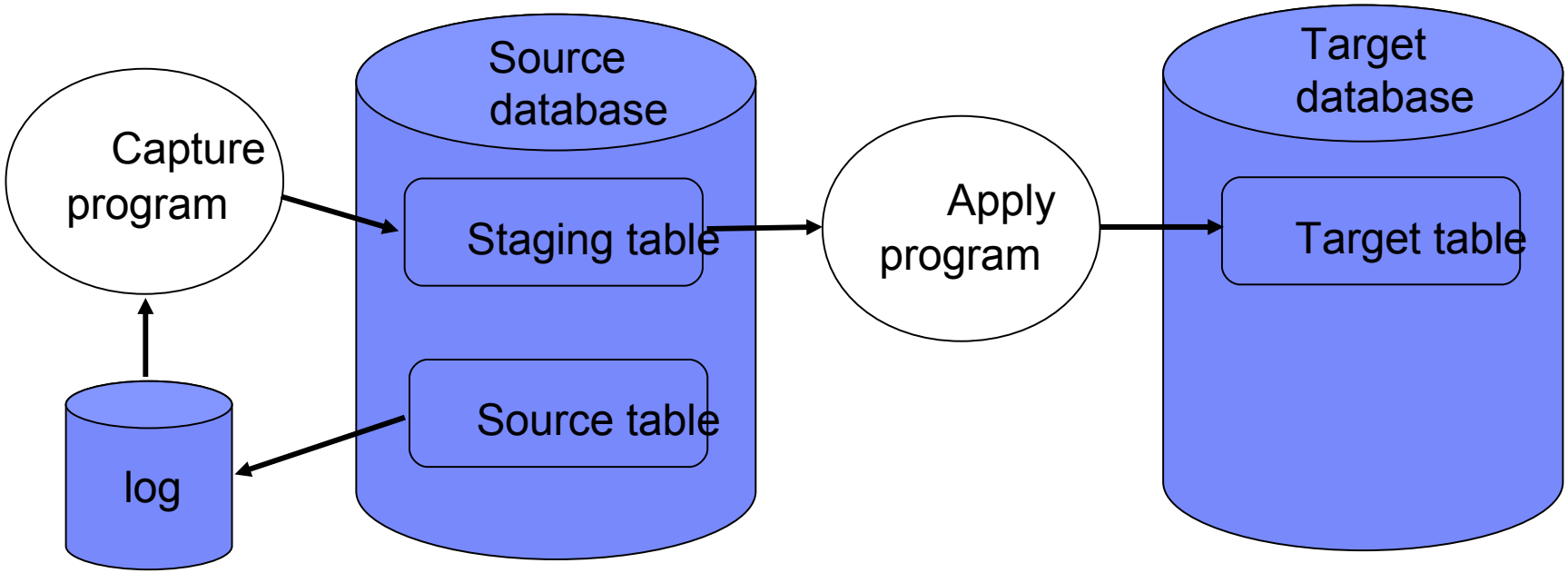
- **Load copy yes**
  - Data is replicated if standby can access the copy. Otherwise, the table space containing the table is marked bad on the standby.
- **Load nonrecoverable**
  - The table (not the whole table space) is marked bad on the standby.
  - User can issue LOAD COPY YES REPLACE on primary to restore the table.
- **Load copy no**
  - Converted to NONRECOVERABLE by default.
  - Registry variable DB2_LOAD_COPY_NO_OVERRIDE
    - Set on primary. Ignored on standby. Also applicable to standard database.
    - "copy yes …" converts copy no to yes. "…" is copy location.
      - Example: "copy yes to /data/load_copy"
    - "nonrecoverable" converts copy no to nonrecoverable.

SQL and Q Replication

- SQL based replication
  - Construct SQL statements based on database log and apply it to another database.
  - Because the SQL statements are applied through regular client interface, the target database can stay in normal mode and accept other client sessions for read or write.
  - There is performance cost to construct and execute the SQL statements.
  - Consistency between source and target database not guaranteed.
- Two IBM products: SQL replication and Q replication.

SQL replication

- SQL replication (a.k.a Data Propagator)
  - Capture and apply programs connect to the databases via SQL client/server connection. They need not be on the same machine as the source or target database.

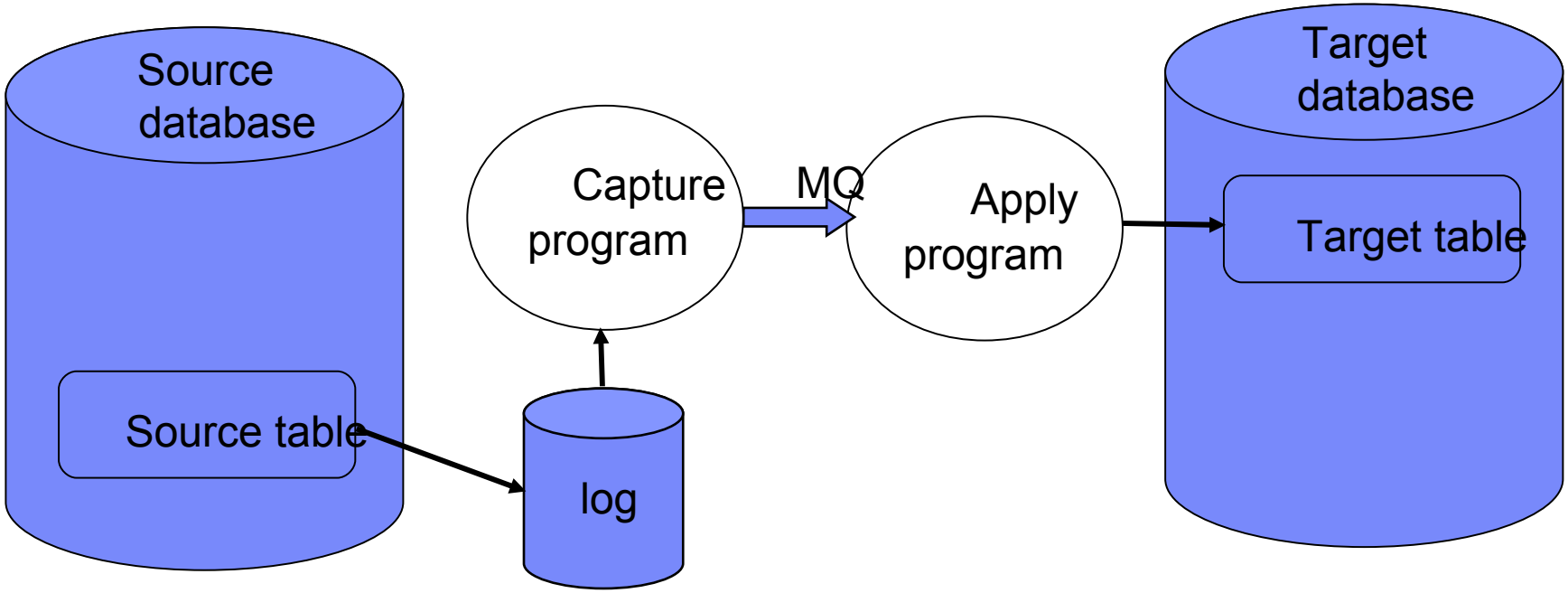


SQL replication

- Apply method, choice of
  - Table mode: Changes applied to each table separately.
  - Transaction mode: Transactions from source database are serialized, then applied.
- Conflict detection
  - Can be enabled. Key based.
  - Changes originated from source database takes precedence.

Q replication

- Capture and Apply programs are connected by WebSphere MQ
- MQ (Message Queue) has its own on-disk staging.



Q replication

- Apply method
  - Parallel (multi-thread) application.
  - Dependencies among transactions detected and used to schedule parallel application.
- Conflict detection
  - More sophisticated than SQL replication. May involve triggers and extra columns for row versioning in source table.
  - Exception table records conflicts.

Comparing HADR and SQL/Q Rep

	HADR	SQL Rep, Q-Rep
Interface to target db	log replay	SQL
Replication unit	Database	Table, filter optional, DDL not replicated
Symmetric host machine	Yes	No
Synchronization mode	User configurable	Async only
Performance	High	Medium: Q-Rep Low: SQL Rep

Comparing HADR and SQL/Q Rep (continued)

	HADR	SQL Rep, Q-Rep
Administration	Easy	Complex, but flexible
Replicate to non-DB2 databases	No	Yes
Read/write on targets	No	Yes
Bi-directional replication	No	Yes
Multiple targets	No	Yes