Remove and reintegrate an auxiliary standby in an HADR setup

A practical guide

Shruthi Rao (shruthra@in.ibm.com)
DB2 LUW FVT Team Member
IBM

Juilee Joshi (juilee.hapse@in.ibm.com)
DB2 LUW Advanced Support Staff
IBM

13 August 2014

Starting with IBM DB2® 10.1, the High Availability Disaster Recovery (HADR) feature supports multiple standbys. With multiple standbys, you can have your data in more than two sites for improved data protection with a single technology. This article provides detailed steps for removing and reintegrating the auxiliary standby in a multiple standby HADR setup.

Introduction

The DB2 High Availability Disaster Recovery (HADR) feature is a database replication method that provides a high-availability solution for partial and complete site failures. The multiple standby HADR feature provides a single technology to provide high availability as well as disaster recovery.

In this article, learn the detailed steps to remove and reintegrate the auxiliary standby in a multiple standby HADR setup. You can then use the auxiliary standby as a single standard server for any DB2 miscellaneous activities. When needed, you can seamlessly reintegrate the server back into the existing HADR setup as the auxiliary standby.

Multiple standby in HADR

The DB2 HADR feature is a database replication, failover, and recovery method that provides an HA solution in the event of database failure. HADR protects against data loss by replicating data...
changes from a source database, called the primary, to one or more target databases, called the standbys.

Starting with V10.1, DB2 provides an enhanced feature whereby you can have multiple standbys — up to three — in an HADR setup to make failover more robust by providing improved data protection. You can assign one of the databases as the principal HADR standby database and the other standby databases as auxiliary HADR standbys. HADR standbys are synchronized with the HADR primary database through a direct TCP/IP connection. Both types of standbys support reads on standby, and you can configure both types for time-delayed log replay. You can also issue a forced or non-forced takeover on any standby.

There are a couple of important distinctions between the principal and auxiliary standbys, as follows:

- IBM Tivoli® System Automation for Multiplatforms (SA MP) automated failover is supported only for the principal standby. You must issue a takeover manually on one of the auxiliary standbys to make one of them the primary. Before issuing a manual takeover, you should disable SA MP.
- All HADR sync modes are supported on the principal standby, but the auxiliary standbys can only be in SUPERASYNC mode.
- To enable HADR multiple-standby mode, use the new `hadr_target_list` database configuration parameter. The number of entries specified by this parameter on the primary determines the number of standbys a primary database has.

**Initializing HADR in multiple standby mode**

As of DB2 10.5, the procedure for initializing your HADR databases changed. You should now use the `hadr_target_list` configuration parameter even if you are only configuring one standby. You can use this parameter to specify up to three standby databases.

**Enabling multiple standby mode on a pre-existing HADR setup**

To enable multiple standby mode on a pre-existing HADR setup, you first create and configure the new standbys only. By keeping the original configuration until the final steps, you can keep your primary-standby pair functioning for as long as possible. To review the procedure in detail, see Enabling multiple standby mode on a pre-existing HADR setup in the IBM Knowledge Center.

**Enabling multiple standby mode on a new HADR setup**

Multiple standby mode requires that you set the `hadr_target_list` configuration parameter on all participating databases. To review the procedure in detail, see Initializing HADR in multiple standby mode in the IBM Knowledge Center.

**Deploying an HADR multiple standby database setup**

Review the Scenario: Deploying an HADR multiple standby database setup in the IBM Knowledge Center to learn about planning, configuring, and deploying an HADR setup for a bank called ExampleBANK. The setup has three standby databases: one principal standby and two auxiliary standbys.
Removing the auxiliary standby dynamically

After the setup of HADR with multiple standbys is done, you might want to add or remove the auxiliary standby server dynamically from the existing setup. If HADR is set up with multiple standby enabled, you can add or drop auxiliary standby servers dynamically. The auxiliary standby can then be used as a standard server to do database activities.

For a dynamic update to take place, the database has to be active and must have at least one connection to it. On a standby database, reads on standby must be enabled to allow connections. If the update is not dynamic, it takes effect the next time HADR is activated.

The following example includes a four host HADR setup with one primary server named PRIMARY_HOST, one principal standby server named P_HOST, and two auxiliary standby servers named AUX1_HOST and AUX2_HOST. Listing 1 shows the example. There is one database, named hadrdb, running on this HADR setup.

### Listing 1. HADR output of database configuration on primary

```bash
$ db2 get db cfg for hadrdb | grep HADR*
HADR database role                                      = PRIMARY
HADR local host name                  (HADR_LOCAL_HOST) = PRIMARY_HOST
HADR local service name                (HADR_LOCAL_SVC) = xshruthra
HADR remote host name                (HADR_REMOTE_HOST) = P_STANDBY
HADR remote service name              (HADR_REMOTE_SVC) = xshruthra
HADR instance name of remote server  (HADR_REMOTE_INST) = shruthra
HADR timeout value                       (HADR_TIMEOUT) = 120
HADR target list                     (HADR_TARGET_LIST) = P_STANDBY:xshruthra|
                                        AUX1_HOST:xshruthra|
                                        AUX2_HOST:xshruthra
HADR log write synchronization mode     (HADR_SYNCMODE) = ASYNC
HADR spool log data limit (4KB)        (HADR_SPOOL_LIMIT) = AUTOMATIC(25600)
HADR log replay delay (seconds)       (HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)   (HADR_PEER_WINDOW) = 0
```

As shown in Listing 1, the hadr_target_list consists of target host:port pairs. xshruthra represents the SVCENAME on each host, which is the same for all hosts in this example.

Removing auxiliary standby for read-only operations

If the reads on standby feature is not enabled on the standby and you need to use the auxiliary standby for read-only operations, you can do so by simply stopping HADR, as shown below.

### Listing 2. Stopping HADR to remove auxiliary standby from HADR setup

```bash
$ db2 stop hadr on db hadrdb
DB20000I  The STOP HADR ON DATABASE command completed successfully.
```

After your read-only operations are completed, you can reintegrate the host back to auxiliary standby. Once reintegrated, the AUX2_HOST will be in remote catchup mode, replaying log files from PRIMARY_HOST.

Removing auxiliary standby for read and write operations

To dynamically remove auxiliary standby from the HADR setup for read as well as write activities, take the steps outlined below.
On the auxiliary standby, in this case AUX2_HOST, deactivate the database and stop HADR by using the commands in Listing 3.

**Listing 3. Commands to deactivate and stop HADR**

```sh
$ db2 deactivate db hadrdb
DB20000I  The DEACTIVATE DATABASE command completed successfully.

$ db2 stop hadr on db hadrdb
DB20000I  The STOP HADR ON DATABASE command completed successfully.
```

The primary host sends redirect messages to each auxiliary standby approximately every 20 seconds. These attempts will fail because we've stopped HADR on auxiliary, so primary won't be able to create a TCP connection. This causes error messages to be logged in db2diag.log at the primary, as shown in Listing 4. To avoid this, remove the auxiliary standby from the `hadr_target_list` on the primary.

**Listing 4. Error messages logged in primary after stopping HADR on auxiliary standby**

```
2014-05-14-23.10.59.220111-420 I704519E488           LEVEL: Error
PID     : 6677                 TID : 46912858220864  PROC : db2sysc
INSTANCE: shruthra             NODE : 000            DB   : HADRDB
HOSTNAME: PRIMARY_HOST
EDUID   : 70                   EDUNAME: db2hadrs.0.3 (HADRDB)
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSendRedirectMsgToOneAddress, probe:43900
MESSAGE : ZRC=0xFFFFFFFF=-1
DATA #1 : <preformatted>
The HADR primary was not able to form a TCP connection with the standby: 9.30.14.252:27070.
```

To remove the AUX2_HOST from the `hadr_target_list` on the primary, a connection to the database is needed. The `hadr_target_list` will be updated successfully after you enter the command in Listing 5.

**Listing 5. Updating hadr_target_list on primary database**

```sh
db2 "update db cfg for hadrdb using HADR_TARGET_LIST P_STANDBY:xshruthra|
    AUX1_HOST:xshruthra"
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
```

Similarly, `hadr_target_list` can be updated on the principal standby and other auxiliary standby to remove the AUX2_HOST from their `target_hadr_list`. The reads on standby feature needs to be enabled on the standby database using the `DB2_HADR_ROS` registry variable, as shown below.

**Listing 6. Read on standby enabled on principal standby**

```sh
$ db2set
DB2_RESTORE_GRANT_ADMIN_AUTHORITIES=ON
DB2_HADR_ROS=ON
DB2AUTH=OSAUTHDB
DB2_SYSTEM_MONITOR_SETTINGS=LAST_USE_INTERVAL:0
DB2SLEEP=OFF
DB2COMM=TCPIP
```

The `hadr_target_list` can then be updated after connecting to the databases on the respective standby, just as you did on the primary database. The updated `hadr_target_list` can be checked
on the primary database, as shown in Listing 7. Similarly, `hadr_target_list` can be checked on the standby targets.

**Listing 7. Updated `hadr_target_list` on primary database host**

```
$ db2 get db cfg for hadrdb | grep HADR*
HADR database role = PRIMARY
HADR local host name  (HADR_LOCAL_HOST) = PRIMARY_HOST
HADR local service name  (HADR_LOCAL_SVC) = xshruthra
HADR remote host name  (HADR_REMOTE_HOST) = P_STANDBY
HADR remote service name  (HADR_REMOTE_SVC) = xshruthra
HADR instance name of remote server  (HADR_REMOTE_INST) = shruthra
HADR timeout value  (HADR_TIMEOUT) = 120
HADR target list  (HADR_TARGET_LIST) = P_STANDBY:xshruthra|AUX1_HOST:xshruthra
HADR log write synchronization mode  (HADR.Syncmode) = ASYNC
HADR spool log data limit (4KB)  (HADR_SPOOL_LIMIT) = AUTOMATIC(25600)
HADR log replay delay (seconds)  (HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)  (HADR_PEER_WINDOW) = 0
```

The auxiliary standby, AUX2_HOST in our example, will become a standard database.

**Listing 8. AUX2-HOST to be standard database**

```
$ db2 get db cfg for hadrdb | grep HADR*
HADR database role = STANDARD
HADR local host name  (HADR_LOCAL_HOST) = AUX2_HOST
HADR local service name  (HADR_LOCAL_SVC) = xshruthra
HADR remote host name  (HADR_REMOTE_HOST) = PRIMARY_HOST
HADR remote service name  (HADR_REMOTE_SVC) = xshruthra
HADR instance name of remote server  (HADR_REMOTE_INST) = shruthra
HADR timeout value  (HADR_TIMEOUT) = 120
HADR target list  (HADR_TARGET_LIST) = PRIMARY_HOST:xshruthra|P_STANDBY:xshruthra|AUX1_HOST:xshruthra
HADR log write synchronization mode  (HADR.Syncmode) = SUPERASYNC
HADR spool log data limit (4KB)  (HADR_SPOOL_LIMIT) = AUTOMATIC(25600)
HADR log replay delay (seconds)  (HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)  (HADR_PEER_WINDOW) = 0
```

If you plan to use the database hadrdb, you need to roll this database forward so it can be used to run the scheduled operations. Listing 9 shows the hadrdb in rollforward and then connected to hadrdb.

**Listing 9. hadrdb in roll forward and then connect to hadrdb**

On AUX2_HOST:
```
$ db2 rollforward db hadrdb complete
Rollforward Status
Input database alias = hadrdb
Number of members have returned status = 1
Member ID = 0
Rollforward status = not pending
Next log file to be read =
Log files processed = S0000000.LOG - S0000000.LOG
Last committed transaction = 2014-05-15-04.18.16.000000 UTC
DB20000I The ROLLFORWARD command completed successfully.
```

$ db2 connect to hadrdb
After the scheduled operations are done, the standard database needs to be reintegrated as an auxiliary standby, as explained in the next section.

Reintegrating the auxiliary standby in an existing HADR setup

Various database operations such as selects, inserts, updates, deletes, and so on might have been conducted on the hadrdb database. Most of these operations could have caused the database data to be in a different state from data on the primary database. Thus, when you need to reintegrate such a database in the HADR setup, you need to perform initial setup steps, as outlined below.

1. Back up the hadrdb database on the primary host, as shown in Listing 10.

   **Listing 10. Database hadrdb backed-up**

   ```
   On PRIMARY_HOST:
   $ db2 backup db hadrdb online to /nfshome/shruthra/shared_dir_path
   Backup successful. The timestamp for this backup image is : 20140514230249
   $ ls /nfshome/shruthra/shared_dir_path
   HADRDB.0.shruthra.DBPART000.20140514230249.001
   ```

2. Drop the database on AUX2_HOST and restore the backup from primary.

   **Listing 11. Database dropped and restored on AUX2-HOST**

   ```
   On AUX2_HOST:
   $ db2 drop db hadrdb
   DB20000I  The DROP DATABASE command completed successfully.
   $ db2 "restore db hadrdb from /nfshome/shruthra/shared_dir_path ON /work/shruthra WITHOUT PROMPTING"
   DB20000I  The RESTORE DATABASE command completed successfully.
   ```

3. Update the `hadr_target_list` on the primary and standby targets to include the added AUX2_HOST.
Listing 12. hadr_target_list for primary and standby targets updated

on PRIMARY_HOST:
$ db2 "update db cfg for hadrdb using HADR_TARGET_LIST P_STANDBY:xshruthra|AUX1_HOST:xshruthra|AUX2_HOST:xshruthra"
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.

on P_STANDBY:
$ db2 "update db cfg for hadrdb using HADR_TARGET_LIST PRIMARY_HOST:xshruthra|AUX1_HOST:xshruthra|AUX2_HOST:xshruthra"
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.

on AUX1_HOST:
$ db2 "update db cfg for hadrdb using HADR_TARGET_LIST PRIMARY_HOST:xshruthra|P_STANDBY:xshruthra|AUX2_HOST:xshruthra"
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.

4. Update all the HADR db cfg parameters on AUX2_HOST, as shown in Listing 13 and Listing 14.

Listing 13. Update command on AUX2_HOST

$ db2 "update db cfg for hadrdb using HADR_LOCAL_HOST AUX2_HOST
HADR_REMOTE_HOST PRIMARY_HOST
HADR_REMOTE_SVC xshruthra
HADR_REMOTE_INST shruthra
HADR_TIMEOUT 120
HADR_TARGET_LIST PRIMARY_HOST:xshruthra|P_STANDBY:xshruthra|AUX1_HOST:xshruthra
HADR_SYNCHMODE SUPERASYNC"
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.

Listing 14. HADR parameters updated

$ db2 get db cfg for hadrdb | grep HADR*
HADR database role                              = STANDARD
HADR local host name                        (HADR_LOCAL_HOST) = AUX2_HOST
HADR local service name                      (HADR_LOCAL_SVC) = xshruthra
HADR remote host name                        (HADR_REMOTE_HOST) = PRIMARY_HOST
HADR remote service name                     (HADR_REMOTE_SVC) = xshruthra
HADR instance name of remote server          (HADR_REMOTE_INST) = shruthra
HADR timeout value                           (HADR_TIMEOUT) = 120
HADR target list                            (HADR_TARGET_LIST) = PRIMARY_HOST:xshruthra|P_STANDBY:xshruthra|AUX1_HOST:xshruthra
HADR SYNCHMODE                              SUPERASYNC
HADR spool log data limit (4KB)              (HADR_SPOOL_LIMIT) = AUTOMATIC(25600)
HADR log replay delay (seconds)              (HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)          (HADR_PEER_WINDOW) = 0

5. Start HADR on AUX2_HOST.

Listing 15. HADR being started on AUX2_HOST

$ db2 start hadr on db hadrdb as standby
DB20000I  The START HADR ON DATABASE command completed successfully.

6. The AUX2_HOST is now reintegrated as the auxiliary standby in the HADR setup with the command in Listing 16.
Listing 16. AUX2_HOST integrated in HADR setup

$ db2 get db cfg for hadrdb | grep HADR*

<table>
<thead>
<tr>
<th>HADR database role</th>
<th>STANDBY</th>
</tr>
</thead>
<tbody>
<tr>
<td>HADR local host name</td>
<td>(HADR_LOCAL_HOST) = AUX2_HOST</td>
</tr>
<tr>
<td>HADR local service name</td>
<td>(HADR_LOCAL_SVC) = xshruthra</td>
</tr>
<tr>
<td>HADR remote host name</td>
<td>(HADR_REMOTE_HOST) = PRIMARY_HOST</td>
</tr>
<tr>
<td>HADR remote service name</td>
<td>(HADR_REMOTE_SVC) = xshruthra</td>
</tr>
<tr>
<td>HADR instance name of remote server</td>
<td>(HADR_REMOTE_INST) = shruthra</td>
</tr>
<tr>
<td>HADR timeout value</td>
<td>(HADR_TIMEOUT) = 120</td>
</tr>
<tr>
<td>HADR target list</td>
<td>(HADR_TARGET_LIST) = PRIMARY_HOST:xshruthra</td>
</tr>
<tr>
<td>HADR log write synchronization mode</td>
<td>(HADR_SYNCMODE) = SUPERASYNC</td>
</tr>
<tr>
<td>HADR spool log data limit (4KB)</td>
<td>(HADR_SPOOL_LIMIT) = AUTOMATIC(25600)</td>
</tr>
<tr>
<td>HADR log replay delay (seconds)</td>
<td>(HADR_REPLAY_DELAY) = 0</td>
</tr>
<tr>
<td>HADR peer window duration (seconds)</td>
<td>(HADR_PEER_WINDOW) = 0</td>
</tr>
</tbody>
</table>

7. After being connected, you should see Handshake messages in the primary and AUX2_HOST db2diag.log, as shown in Listing 17 and Listing 18.

Listing 17. Messages on PRIMARY_HOST db2diag.log

part of db2diag.log on PRIMARY_HOST :
PID : 2784       TID : 4691293718336       PROC : db2sysc
INSTANCE: shruthra      NODE : 000      DB   : HADRDB
HOSTNAME: PRIMARY_HOST
EDUID   : 52       EDUNAME: db2hadrp.0.1 (HADRDB)
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrHandleHsAck, probe:43900
DATA #1 : <preformatted>
Handshake HDR_MSG_HDRHS message is received from AUX2_HOST:xshruthra (9.30.203.76:27070)

PID : 2784       TID : 46912921135424       PROC : db2sysc
INSTANCE: shruthra      NODE : 000      DB   : HADRDB
HOSTNAME: PRIMARY_HOST
EDUID   : 79       EDUNAME: db2hadrp.0.3 (HADRDB)
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrAfterVerifySystem, probe:30440
DATA #1 : <preformatted>
Connection succeeded, connId=1

Listing 18. Messages on AUX2_HOST db2diag.log

part of db2diag.log on AUX2_HOST :
2014-05-14-23.25.59.220155-420 I25831941E427 LEVEL: Info
PID : 6677       TID : 4691293718336       PROC : db2sysc
INSTANCE: shruthra      NODE : 000      DB   : HADRDB
HOSTNAME: AUX2_HOST
EDUID   : 70       EDUNAME: db2hadrs.0.0 (HADRDB)
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrHandleHsAck, probe:43900
DATA #1 : <preformatted>
Handshake HDR_MSG_HDRACK message is received from PRIMARY_HOST:xshruthra (9.30.203.76:27070)

PID : 6677       TID : 46912921135424       PROC : db2sysc
INSTANCE: shruthra      NODE : 000      DB   : HADRDB
HOSTNAME: AUX2_HOST
EDUID   : 70       EDUNAME: db2hadrs.0.0 (HADRDB)
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrAfterVerifySystem, probe:30440
DATA #1 : <preformatted>
Connection succeeded, connId=1
Conclusion

In this tutorial, you have learned the simple steps for dropping and adding an auxiliary standby target in an HADR setup.

Acknowledgements

We would like to thank Effi Ofer and Tamilselvan Narayanaswamy for reviewing this article and the developerWorks Information Management team for their editorial contributions.
Resources

Learn

- Learn more about Initializing HADR in the IBM Knowledge Center.
- Read about Troubleshooting HADR in the IBM Knowledge Center.
- "DB2 10.1 HADR Multiple Standbys" (developerWorks, 2012) shows how HADR multiple standbys can be set up, configured, and monitored.
- Visit the developerWorks Information Management zone to find more resources for DB2 developers and administrators.
- Stay current with developerWorks technical events and webcasts focused on a variety of IBM products and IT industry topics.
- Follow developerWorks on Twitter.

Get products and technologies

- Evaluate IBM products in the way that suits you best: Download a product trial, try a product online, or use a product in a cloud environment.

Discuss

- Get involved in the developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.
About the authors

Shruthi Rao

Shruthi Rao has been working on the DB2 LUW team for the past six years. He has worked on various releases of DB2 and on various features. For the past year, he has focused on DB2 HADR.

Juilee Joshi

Juilee Joshi has worked with DB2 LUW L2 support for the past five years. As a support analyst, she works with DB2 customers to help them resolve technical issues.

© Copyright IBM Corporation 2014
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)