Client Failover Best Practices for Highly Available Oracle Databases: Oracle Database 11g Release 2

Oracle Maximum Availability Architecture White Paper
February 2011

Maximum Availability Architecture

Oracle Best Practices For High Availability

ORACLE

Executive Overview 2
Introduction 3
Prerequisites 4
Automatic Failover for Client Applications that Support FAN 6
    Role-Based Database Services 6
    OCI and JDBC Client Configuration 10
    OLE DB and ODP .Net Clients 15
Configuring PeopleSoft for Automatic Client Failover 16
Automatic Failover for Clients Applications that do not Support FAN 17
    OCI Applications: 17
    JDBC Applications: 18
Additional Considerations for Active Data Guard Connections 18
    Initial Connections: 18
    Managing Existing Connections for Unplanned Outages 19
Additional Considerations for Data Guard SQL Apply 19
Client Transition during Switchover Operations 20
    Physical standby switchover: 21
    Logical standby switchover: 21
Considerations for Prior Oracle Releases 21
Conclusion 22

Executive Overview

Oracle Maximum Availability Architecture (MAA)[1] is a best practices blueprint for implementing Oracle high availability technologies. Oracle Data Guard is a key component of MAA. It provides the management, monitoring, and automation software to create and maintain one or more synchronized standby databases to protect Oracle data from failures, disasters, errors, and data corruptions. If a production database becomes unavailable for any reason, administrators can execute either manual or automatic failover to a standby database in order to maintain high availability. This paper describes best practices to automatically transition application connections from a failed primary database to a new primary database after a Data Guard role transition has occurred.

[1] http://www.oracle.com/goto/maa

Introduction

Unplanned failures of an Oracle Database instance fall into three general categories:

1. A server failure or other fault that causes the crash of an individual Oracle instance in an Oracle RAC database. To maintain availability, application clients connected to the failed instance must be quickly notified of the failure and immediately establish a new connection to the surviving instances of the Oracle RAC database. Fast Application Notification (FAN) will break connected clients out of TCP timeout, and Transparent Application Failover (OCI clients) or Fast Connection Failover (JDBC clients) will automatically fail clients over to database services running on surviving database instances. Detailed best practices for this category of failure are described in, *Automatic Workload Management with Oracle Real Application Clusters 11g Release 2*[2], and the *Oracle Real Application Clusters Administration and Deployment Guide*[3].
2. A complete-site failure that results in both the application and database tiers being unavailable. To maintain availability users must be redirected to a secondary site that hosts a redundant application tier and a synchronized copy of the production database. MAA best practice is to maintain a running application tier at the standby site to avoid startup time and to use Data Guard to maintain the synchronized copy of the production database. A WAN traffic manager is used to execute a DNS failover (either manually or automatically) to redirect users to the application tier at standby site while a Data Guard failover transitions of the standby database to the primary production role. See *Oracle Database High Availability Best Practices*[4] documentation for information on automating complete site failover.
3. A partial-site failure where the primary database (a single-instance database or all nodes in an Oracle RAC database) has become unavailable but the application tier at the primary site remains intact. If there is a local Data Guard standby database then all that is required to maintain availability is to redirect the application tier to the new primary database after a Data Guard failover. The same holds true when there is a remote Data Guard standby database if the surviving application tier can deliver acceptable performance using a remote connection

[2] http://www.oracle.com/technetwork/database/clustering/overview/awm11gr2-130711.pdf

[3] http://download.oracle.com/docs/cd/E11882_01/rac.112/e16795/hafeats.htm#BABECAFD

[4] http://download.oracle.com/docs/cd/B19306_01/server.102/b25159/outage.htm#BABHCAIA

after a database failover has occurred. Similar to the Oracle RAC use case, FAN will break connected clients out of TCP timeout, and Transparent Application Failover (OCI clients) or Fast Connection Failover (JDBC clients) will automatically fail applications over to the new primary database. This paper provides best practices for automatic application failover to a new primary database for this category of outage.

Prerequisites

Database Services are foundational to the application failover best practices described in this paper. If you do not already have a thorough understanding of database services please review the Oracle Database Net Services Administrator's Guide[5] before proceeding. Similarly, you must have an understanding of the highly available application framework that Oracle provides for both single instance databases using Oracle Restart[6], and for Oracle RAC databases. Please see the following for details on this framework: *Automatic Workload Management with Oracle Real Application Clusters 11g Release 2*[7] and the *Oracle Real Application Clusters Administration and Deployment Guide*[8].

The best practices described in this paper require that the Data Guard configuration be managed by the Data Guard Broker[9]. The Data Guard Broker is responsible for sending FAN events to client applications in order to clean up their connections to the down database and reconnect to the new production database. In addition, Oracle Clusterware must be installed and active on the primary and standby sites for both single instance (using Oracle Restart) and Oracle RAC databases. The Data Guard broker will coordinate with Oracle Clusterware to properly fail over role-based services to a new primary database after a Data Guard failover has occurred.

In order to receive and react to FAN events client applications must meet certain requirements:

<u>JDBC applications:</u>

- The implicit connection cache is enabled.

[5] http://download.oracle.com/docs/cd/E11882_01/network.112/e10836/concepts.htm#NETAG175

[6] http://download.oracle.com/docs/cd/E11882_01/server.112/e17120/restart001.htm#ADMIN13388

[7] http://www.oracle.com/technetwork/database/clustering/overview/awm11gr2-130711.pdf

[8] http://download.oracle.com/docs/cd/E11882_01/rac.112/e16795/hafeats.htm#RACAD7136

[9] http://www.oracle.com/pls/db112/to_toc?pathname=server.112/e17023/toc.htm

- The application uses service names to connect to the database.
- The underlying database has Oracle Database 11g Real Application Clusters (Oracle RAC) capability or Oracle Restart (for single instance databases).
- Oracle Notification Service (ONS) is configured and available on the node where JDBC is running.
- The Java Virtual Machine (JVM) in which your JDBC instance is running must have `oracle.ons.oraclehome` set to point to your ORACLE_HOME.
- For more information see the Oracle Database JDBC Developer's Guide.[10]

<u>OCI applications:</u>

- An Oracle RAC environment with Oracle Clusterware set up and enabled or a single node (non-Oracle RAC) database with Oracle Restart..
- The application must have been linked with the threads library.
- The OCI environment must be created in `OCI_EVENTS` and `OCI_THREADED` mode.
- For more information see the Oracle Call Interface Programmer's Guide.[11]

<u>ODP .Net:</u>

- Namespace: Oracle.DataAccess.Client, Assembly: Oracle.DataAccess.dll
- Microsoft .NET Framework Version 2.0 or later.
- For more information see the Oracle Database Administrator's Guide.[12]

[10] http://download.oracle.com/docs/cd/E11882_01/java.112/e16548/fstconfo.htm#CIHJBFFC

[11] http://download.oracle.com/docs/cd/E11882_01/appdev.112/e10646/oci09adv.htm#sthref1523

[12]http://download.oracle.com/docs/cd/E11882_01/server.112/e17120/restart002.htm#ADMIN13196

Automatic Failover for Client Applications that Support FAN

At a high level, automating client failover in a Data Guard configuration includes relocating database services to the new primary database as part of a Data Guard failover, notifying clients that a failure has occurred in order to break them out of TCP timeout, and redirecting clients to the new primary database.

The sections below describe how to create role based database services for both OCI and JDBC applications. Subsequent sections provide detailed configuration steps for enabling OCI and JDBC, OLE DB and ODP .Net application clients to receive FAN notifications and reconnect to a new primary database. If your application client does not support FAN, then please refer to the section of this paper titled *Automatic Failover for Applications that do not Support FAN*.

Role-Based Database Services

Beginning with Data Guard 11g Release 2 you can automatically control the startup of database services on primary and standby databases by assigning a database role `[-l {[PRIMARY] | [PHYSICAL_STANDBY] | [LOGICAL_STANDBY] |[SNAPSHOT_STANDBY]}]` to each service.[13] A database service will automatically start upon database startup if the management policy of the service is AUTOMATIC and if one of the roles assigned to that service matches the current role of the database.

Services must be configured with the Server Control (`SRVCTL`) utility identically on all databases in a Data Guard configuration. In the following examples, a service named `oltpworkload` is configured to be active when the database `Austin` is in the primary role (`-l PRIMARY`). The same service is also configured on the standby database `Houston` so that is started whenever `Houston` functions in the primary role.

Similarly, a second service named `reports` is configured to be started when `Austin` or `Houston` are functioning in the standby database role (`-l PHYSICAL_STANDBY`). The `reports` service provides real-time reporting using Active Data Guard (the standby database is open read-only at the same time it is applying redo received from the primary database).

[13] http://download.oracle.com/docs/cd/E11882_01/rac.112/e16795/hafeats.htm#RACAD7126

Configuring OCI Database Services

The following example shows how to create database services for OCI clients with HA notifications and server side TAF enabled. The example refers to an Oracle RAC primary and standby database. The same procedures are followed for single-instance databases using Oracle Restart.

1. On the primary and standby hosts create the service (oltpworkload) that the application will use to connect to the database. The service should be created such that it is associated with and runs on the database when it is in the `'PRIMARY'` database role:

Primary cluster:

```
srvctl add service -d Austin -s oltpworkload -r ssa1,ssa2,ssa3,ssa4 -l
PRIMARY -q TRUE -e SESSION -m BASIC -w 10 -z 150
```

Standby cluster:

```
srvctl add service -d Houston -s oltpworkload -r ssb1,ssb2,ssb3,ssb4 -
l PRIMARY -q TRUE -e SESSION -m BASIC -w 10 -z 150
```

2. If the standby is also going to support read-only reporting applications, then create a service specific for this workload (reports) that will start when the database is in `PHYSICAL_STANDBY` role.

Primary cluster:

```
srvctl add service -d Austin -s reports -r ssa1,ssa2,ssa3,ssa4 -l
PHYSICAL_STANDBY -q TRUE -e SESSION -m BASIC -w 10 -z 150
```

Standby cluster:

```
srvctl add service -d Houston -s reports -r ssb1,ssb2,ssb3,ssb4 -l
PHYSICAL_STANDBY -q TRUE -e SESSION -m BASIC -w 10 -z 150
```

In addition to creating the database service "reports" on both clusters, the following SQL statement must also be run on the primary database so that the service definition is transmitted via the redo stream and applied to the physical standby database:

SQL run at the Primary database:

```
EXECUTE DBMS_SERVICE.CREATE_SERVICE('reports', 'reports', NULL,
NULL,TRUE, 'BASIC', 'SESSION', 150, 10, NULL);
```

The above examples illustrate how to create role based services with server side Transparent Application Failover (TAF) enabled. Any OCI client that connects to a service that has the TAF attributes set implicitly inherits those attributes. There is no need to configure TAF at the client side in the tnsnames.ora file. The following table explains the TAF attributes being used:

TABLE 1, TRANSPARENT APPLICATION FAILOVER ATTRIBUTES IMPLICITLY SET FOR OCI CLIENTS

ATTRIBUTE	DESCRIPTION
`-e {NONE |SESSION | SELECT}`	The type of failover. Three types of Oracle Net failover functionality are available by default to Oracle Call Interface (OCI) applications: • session: Set to failover the session. If a user's connection is lost, then a new session is automatically created for the user on the backup. This type of failover does not attempt to recover select operations. • select: Set to enable users with open cursors to continue fetching on them after failure. However, this mode involves overhead on the client side in normal select operations. • none: This is the default. No failover functionality is used. This can also be explicitly specified to prevent failover from happening.
`-m {NONE | BASIC}`	Setting for fast failover from the primary site to the standby site: • none: disable transparent application failover • basic: Set to establish connections at failover time. This option requires almost no work on the backup server until failover time.
`-w integer`	The amount of time in seconds to wait between connect attempts. If RETRIES8 is specified, then DELAY defaults to one second. If a callback function is registered, then this parameter is ignored.
`-z integer`	The number of times to attempt to connect after a failover. If DELAY is specified, then RETRIES defaults to five retry attempts. If a callback function is registered, then this parameter is ignored.
`-q {TRUE | FALSE}`	Send Oracle Advanced Queuing (AQ) HA notifications. For standalone servers, applicable in Oracle Data Guard environments only.

Configuring JDBC Database Services

Do not enable HA notifications or TAF on services for JDBC database services that are FAN enabled. The following example illustrates creating the same services as above for JDBC applications:

1. Primary database service performing read/write workload:

Primary cluster:

```
srvctl add service -d Austin -s oltpworkload -r ssa1,ssa2,ssa3,ssa4 -l
PRIMARY -q FALSE -e NONE -m NONE -w 0 -z 0
```

Standby cluster - JDBC:

```
srvctl add service -d Houston -s oltpworkload -r ssb1,ssb2,ssb3,ssb4 -
l PRIMARY -q FALSE -e NONE -m BASIC -w 0 -z 0
```

2. Read-Only database services for an Active Data Guard standby database:

Primary cluster – JDBC r/o service:

```
srvctl add service -d Austin -s reports -r ssa1,ssa2,ssa3,ssa4 -l
PHYSICAL_STANDBY -q FALSE -e NONE -m BASIC -w 0 -z 0
```

Standby cluster – JDBC r/o service:

```
srvctl add service -d Houston -s reports -r ssb1,ssb2,ssb3,ssb4 -l
PHYSICAL_STANDBY -q FALSE -e NONE -m BASIC -w 0 -z 0
```

Note that the service attributes for HA notifications or TAF described in the previous examples have been NOT been enabled for JDBC clients, this is because doing so would interfere with ONS processing for JDBC clients.

In addition to creating services on both clusters the following SQL statement must be run on the primary database so that service definitions are also applied to the standby database:

```
EXECUTE DBMS_SERVICE.CREATE_SERVICE(
         service_name => 'reports'
         network_name => 'reports'
         goal => 'NULL'
         dtp  => 'NULL'
         aq_ha_notifications => 'FALSE'
         failover_method => 'NONE'
         failover_type => 'NONE'
         failover_retries => 0
         failover_delay => 0
         clb_goal => 'NULL');
```

Configuring a Database Service for both OCI and JDBC Access

The examples above describe service definitions that are unique for either OCI or JDBC clients. Each type of client uses a different HA notification framework (OCI clients use Oracle AQ and JDBC clients use ONS). This means that different service definitions (one for OCI and one for JDBC) will need to be created to enable both OCI and JDBC clients to connect to the same database service. Building on the example of the service `oltpworkload` from above, service definitions would be as follows to enable access by both OCI and JDBC clients:

<u>Primary cluster:</u>

```
srvctl add service -d Austin -s oltpworkload_oci -r
ssa1,ssa2,ssa3,ssa4 -l PRIMARY -q TRUE -e SESSION -m BASIC -w 10 -z
150

srvctl add service -d Austin -s oltpworkload_jdbc -r
ssa1,ssa2,ssa3,ssa4 -l PRIMARY -q FALSE -e NONE -m NONE -w 0 -z 0
```

<u>Standby cluster</u>:

```
srvctl add service -d Houston -s oltpworkload_oci -r
ssb1,ssb2,ssb3,ssb4 -l PRIMARY -q TRUE -e SESSION -m BASIC -w 10 -z
150

srvctl add service -d Houston -s oltpworkload_jdbc -r
ssb1,ssb2,ssb3,ssb4 -l PRIMARY -q FALSE -e NONE -m BASIC -w 0 -z 0
```

OCI and JDBC Client Configuration

The following sections describe how to configure OCI and JDBC client applications to enable FAN support. This allows application clients to receive notification that a failure has occurred, break them out of TCP timeout, and reconnect to database services running on the new primary database.

If your application client does not support FAN, then please refer to the section of this paper titled *Automatic Failover for Applications that do not Support FAN*.

Configuring Automatic Failover for OCI Clients

1. Enable FAN for OCI clients by initializing the environment with the OCI_EVENTS parameter, as in the following example:

```
OCIEnvCreate(...OCI_EVENTS...)
```

2. Link the OCI client applications with thread library libthread or libpthread.

3. Your application will need the ability to check if an event has occurred by using code similar to that used in the following example:

```
void evtcallback_fn(ha_ctx, eventhp)
...
printf("HA Event received.\n");
  if (OCIHandleAlloc( (dvoid *)envhp, (dvoid **)&errhp, (ub4)
OCI_HTYPE_ERROR,
                          (size_t) 0, (dvoid **) 0))
    return;
  if (retcode = OCIAttrGet(eventhp, OCT_HTYPE_EVENT, (dvoid
*)&srvhp, (ub4 *)0,
                               OCI_ATTR_HA_SRVFIRST, errhp))
    checkerr (errhp, (sword)retcode;
  else {
     printf("found first server handle.\n");
     /*get associated instance name */
     if (retcode = OCIAttrGet(srvhp, OCI_HTYPE_SERVER, (dvoid
*)&instname,
                               (ub4 *)&sizep, OCI_ATTR_INSTNAME,
errhp))
       checkerr (errhp, (sword)retcode);
     else
       printf("instance name is %s.\n", instname);
```

4. Clients and applications can register a callback that is invoked whenever a high availability event occurs, as shown in the following example:

```
/*Registering HA callback function */
  if (checkerr(errhp, OCIAttrSet(envhp, (ub4) OCI_HTYPE_ENV,
                             (dvoid *)evtcallback_fn, (ub4) 0,
                             (ub4)OCI_ATTR_EVTCBK, errhp)))
  {
    printf("Failed to set register EVENT callback.\n");
    return EX_FAILURE;
  }
  if (checkerr(errhp, OCIAttrSet(envhp, (ub4) OCI_HTYPE_ENV,
                                (dvoid *)evtctx, (ub4) 0,
                                (ub4)OCI_ATTR_EVTCTX, errhp)))
  {
    printf("Failed to set register EVENT callback context.\n");
    return EX_FAILURE;
  }
return EX_SUCCESS;
```

After registering an event callback and context, OCI will call the registered function once for each high availability event.

5. Configure an Oracle Net alias that the OCI application will use to connect to the database. The Oracle Net alias should specify both the primary and standby SCAN hostnames. For best performance while creating new connections the Oracle Net alias should have `LOAD_BALANCE=OFF` for the `DESCRIPTION_LIST` so that `DESCRIPTION`s are tried in an ordered list, top to bottom. With this configuration the second `DESCRIPTION` is only attempted if all connection attempts to the first `DESCRIPTION` have failed.

```
SALES=
 (DESCRIPTION_LIST=
  (LOAD_BALANCE=off)
  (FAILOVER=on)
  (DESCRIPTION=
(CONNECT_TIMEOUT=5)(TRANSPORT_CONNECT_TIMEOUT=3)(RETRY_COUNT=3)
    (ADDRESS_LIST=
      (LOAD_BALANCE=on)
      (ADDRESS=(PROTOCOL=TCP)(HOST=Austin-scan)(PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME=oltpworkload)))
  (DESCRIPTION=
(CONNECT_TIMEOUT=5)(TRANSPORT_CONNECT_TIMEOUT=3)(RETRY_COUNT=3)
    (ADDRESS_LIST=
      (LOAD_BALANCE=on)
      (ADDRESS=(PROTOCOL=TCP)(HOST= Houston-scan)(PORT=1521)))
    (CONNECT_DATA=(SERVICE_NAME=oltpworkload))))
```

When a new connection is made using the above Oracle Net alias the following logic is used:

a) Oracle Net contacts DNS and resolves Austin-scan to a total of 3 IP addresses.

b) Oracle Net randomly picks one of the 3 IP address and attempts to make a connection. If the connection attempt to the IP address does not respond in 3 seconds (`TRANSPORT_CONNECT_TIMEOUT`) the next IP address is attempted. All 3 IP addresses will be tried a total of four times (initial attempt plus `RETRY_COUNT` in the above example).

c) If the connection to primary site is unsuccessful, it then contacts DNS and resolves Houston-scan to 3 addresses.

d) The same sequence is performed for the standby Houston-scan as it was for the Austin-scan.

Note that the above is true only for Oracle Database 11g Release 2 clients. For additional information on SCAN consult the Oracle Real Application Clusters 11g Release 2 Overview of SCAN technical whitepaper[14].

Additional information on the Oracle Net parameters used in the above alias:

[14] http://www.oracle.com/technetwork/database/clustering/overview/scan-129069.pdf

- `LOAD_BALANCE` is `ON` by default for `DESCRIPTION_LIST` only. This parameter by default is `OFF` for an address list within a `DESCRIPTION`. Setting this ON for a SCAN-based address implies that new connections will be randomly assigned to one of the 3 SCAN-based IP addresses resolved by DNS.
- In certain situations, round-robin address assignment by DNS may not be possible - see the Oracle Database 11.2.0.2 Readme. The best practice to ensure connect-time client load balancing across the 3 SCAN IP addresses is to explicitly specify `LOAD_BALANCE=on`. Note that this behavior is independent of server-side load balancing which will occur subsequently, after the initial SCAN listener receives the connection request.
- The default value for the `FAILOVER` parameter is `ON` for an address list within a `DESCRIPTION`. This impacts the 3 SCAN IP addresses the same way as if those 3 IP addresses were listed explicitly in the connect descriptor. This means that if the initial connection requests to the first randomly-assigned SCAN IP address fails, the connection will failover to another SCAN IP address, and will continue to do so, till it iterates the complete address list. Note that this parameter is relevant only to new connections. Failover of existing connections is handled by TAF, which is controlled by the separate `FAILOVER_MODE` parameter.
- The `CONNECT_TIMEOUT` parameter is the time to connect to the database instance providing the requested service, and includes the time to establish a TCP connection to the listener. The TCP duration is controlled by `TRANSPORT_CONNECT_TIMEOUT`, which has a default value of 60 seconds. If both timeouts are specified, it is recommended that `CONNECT_TIMEOUT` be set to a value slightly greater than `TRANSPORT_CONNECT_TIMEOUT`. The timeout interval is applicable for each `ADDRESS` in an `ADDRESS_LIST`, and each IP address to which a host name is mapped. Set the `CONNECT_TIMEOUT` parameter to the maximum amount of time (in seconds) to wait for a response from an address before skipping to the next address. A setting of three seconds is recommended and is acceptable in most cases. Do not set this parameter to fewer than three seconds.

  The equivalent global parameter in sqlnet.ora is `SQLNET.OUTBOUND_CONNECT_TIMEOUT`. If the same timeout value is sufficient for all connect strings, it would be simpler to set the global parameter. Otherwise, a separate setting can be done for each connect string.

  The equivalent global parameter for `TRANSPORT_CONNECT_TIMEOUT` is `TCP.CONNECT_TIMEOUT`. Both these parameters are applicable only when the protocol is TCP.
- The `RETRY_COUNT` parameter specifies the number of times an address list is traversed before the new connection attempt is terminated. The default value is 0. With respect to SCAN, with `FAILOVER = on`, setting this `RETRY_COUNT` parameter to a value of 2 (for example), means the 3 SCAN IP addresses are traversed thrice (i.e. 3*3=9 connect attempts), before the connection is terminated:

- When the connection request initially comes in, the first randomly assigned IP address tries to service that request, followed by the two remaining IP addresses (this behavior is controlled by the `FAILOVER` parameter);
- The retries then kick in and the list of 3 IP addresses is tried two more times. `RETRY_COUNT` is only supported at `DESCRIPTION` level in connect string, but not at global (i.e. sqlnet.ora) level.

Configuring Automatic Failover for JDBC Clients

1. Configure JDBC clients to use a connect descriptor that includes an address list that in turn includes the SCAN address for each site and connects to an existing service. Do not configure TAF with Fast Connection Failover for JDBC thick clients as TAF processing will interfere with FAN ONS processing.

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource")
;
    pds.setUser("system");
    pds.setPassword("oracle");
    String dbURL =
      "jdbc:oracle:thin:@" +
  "(DESCRIPTION_LIST=" +
   "(LOAD_BALANCE=off)" +
   "(FAILOVER=on)" +
   "(DESCRIPTION=" +
     "(ADDRESS_LIST=" +
       "(LOAD_BALANCE=on)" +
       "(ADDRESS=(PROTOCOL=TCP)(HOST=Austin-scan)(PORT=1521)))" +
     "(CONNECT_DATA=(SERVICE_NAME=oltpworkload)))" +
   "(DESCRIPTION=" +
     "(ADDRESS_LIST=" +
       "(LOAD_BALANCE=on)" +
       "(ADDRESS=(PROTOCOL=TCP)(HOST= Houston-scan)(PORT=1521)))" +
     "(CONNECT_DATA=(SERVICE_NAME=oltpworkload))))";
     System.out.println("Url=" + dbURL);
     pds.setURL(dbURL);
```

2. The JDBC client must set the `oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR` property. This property enables the JDBC client to quickly traverse an `ADDRESS_LIST` in the

event of a failure. For example, if the client attempts to connect to a host that is unavailable, the connection attempt will be bounded to the time specified by the `SQLnetDef.TCP_CONNTIMEOUT_STR` property after which the client attempts to connect to the next host in the `ADDRESS_LIST`. The behavior continues for each host in the `ADDRESS_LIST` until a connection is made. Setting the property to a value of 3 seconds will suffice in most environments. It is important to note that the `SQLnetDef.TCP_CONNTIMEOUT_STR` property should be set on the data source and not on the implicit connection cache.

```
    Properties prop = new Properties();
    prop.put(oracle.net.ns.SQLnetDef.TCP_CONNTIMEOUT_STR, ""+5000);
// 5000ms
    pds.setConnectionProperties(prop);
```

3. Enable Fast Connection Failover (FCF) and configure the JDBC application to connect to all ONS daemons for both the primary and standby clusters using the `setONSConfiguration` property. The `setONSConfiguration` property should point to all primary and standby ONS daemons.

```
pds.setONSConfiguration("nodes=hasun05:6200,hasun06:6200,hasun07:6200,
hasun08:6200");
pds.setFastConnectionFailoverEnabled(true);
```

4. By default the JDBC application will randomly pick three hosts from the `setONSConfiguration` property and create connections to those three ONS daemons. This default must be changed so that connections are made to all ONS daemons. This is done by setting the following property when the JDBC application is invoked to the total number of ONS daemons in the configuration:

```
java -Doracle.ons.maxconnections=4
```

OLE DB and ODP .Net Clients

The following sections describe how to configure OLE DB and ODP .Net client applications to enable FAN support and to receive HA notifications. This allows application clients to receive notification that a failure has occurred, break them out of TCP timeout, and reconnect to database services running on the new primary database.

If your application client does not support FAN, then please refer to the section of this paper titled *Automatic Failover for Applications that Do Not Support FAN.*

To configure automatic client failover for OLE DB and ODP .Net clients, first perform the same configuration steps described above for OCI Clients and then perform the steps provided below.

Configuring Automatic Failover for OLE DB clients

1. Set the OraOLEDB connection string attribute: `DBNotifications = true`

This can also be set via the registry.

2. Set the OraOLEDB connection string attribute: `DBNotificationPort = [unsigned integer]`

Setting the `DBNotificationPort` attribute allows a specific port to be assigned for use. If this attribute is not set then the port is randomly selected.

Configuring Automatic Failover for ODP .Net clients

Follow the configuration steps described in the section "Connection Optimization for Oracle RAC and Data Guard" in the *Oracle Data Provider for .NET Developers Guide*[15].

Configuring PeopleSoft for Automatic Client Failover

PeopleSoft PeopleTools version 8.50.09 and higher supports FAN. This enables PeopleSoft applications to automatically failover database connections to a surviving instance in an Oracle RAC cluster or to a new primary database in a Data Guard configuration should its database connection be lost. In the event of a RAC instance failure, primary database failure, or a shutdown/restart of the Oracle Database, PeopleSoft servers and clients continue running and users are not required to login a second time.

The same OCI configuration steps described earlier in this paper are used for PeopleSoft applications.

---

[15] http://download.oracle.com/docs/cd/E11882_01/win.112/e18754/featConnecting.htm#sthref202

Refer to My Oracle Support Note 876292.1for important prerequisites for automatic client failover for PeopleSoft

When properly configured, the failover process is as follows:

- When either a manual or automatic database failover is initiated, the standby database transitions to the primary database role.
- The PeopleSoft database service appropriate to the primary role is automatically started using Oracle Database 11.2 role-specific services, making the new primary database immediately available to accept new connections.
- As part of the database service startup, the `AQ_HA_NOTIFICATIONS` parameter causes a FAN event to be sent to all previously connected clients.
- Upon receipt of the FAN event, the clients will break their existing TCP connections to the failed database instance and automatically failover by going to the next host in the `TNSNAMES.ORA` connect alias address list until they establish a connection to the new primary database.

Automatic Failover for Clients Applications that do not Support FAN

The following sections describe how to configure automatic failover for applications that do not support FAN events. This includes a number of applications from Oracle (e.g. Siebel and Oracle E-Business Suite), and application servers that have not yet implemented FAN support. Even though FAN events cannot be used in such cases, applications can still be configured for efficient failover by using timeouts and application retries

OCI Applications:

For OCI applications follow the steps described in the above section titled *Configuring Automatic Failover for OCI Clients* with the exception of the steps relating to enabling FAN support (steps 1 through 4). Given that your client does not support FAN, you will use TCP timeouts at either the OS or Oracle Net level to break client connections out of a TCP hang.

1. Configure your operating system for efficient TCP timeouts on the hosts that run the application layer. The OS TCP timeouts should be set to the amount of time it takes for the database layer to failover and the database services to be started. Consult your operating system manuals for how to properly configure TCP timeout.

2. Application retries are automated by configuring server-side TAF as described in the above section titled *Configuring OCI Database Services*. If the application cannot use TAF then the application must be configured with reconnection logic in the event of an exception. For example, when a session from the connection pool receives any exception which results in a

disconnect (such as an ORA-3113 error) the application should automatically attempt to reconnect that session. The reconnection attempts should be configured such that they will continue for the length of time that it takes to failover the database layer and bring the application services online.

3. It is important to note that TAF can only automate retries for an existing session. To automate retries when a new connection attempt receives on error use the `RETRY_COUNT` parameter that is part of an Oracle Net `DESCRIPTION_LIST`. This parameter specifies the number of times an `ADDRESS` list is traversed before the connection attempt is terminated.

JDBC Applications:

1. Configure your operating system for efficient TCP timeouts on the hosts that run the application layer. The OS TCP timeouts should be set to the amount of time it takes for the database layer to failover and the database services to be started. Consult your operating system manuals for how to properly configure TCP timeout.

2. Configure reconnection logic within the application to respond appropriately in the event of an exception. For example, when a session from the connection pool receives an exception that results in disconnect (such as an ORA-3113 error), the application should automatically attempt to reconnect that session. The reconnection attempts should be configured such that they will continue for the length of time that it takes to failover the database layer and bring the application services online.

Additional Considerations for Active Data Guard Connections

Oracle Active Data Guard 11g extends basic Data Guard functionality by allowing read-only access to a physical standby database while the standby continuously applies changes received from the primary database. Thus a physical standby database can provide disaster protection while it also increases performance, scalability, and return on investment by offloading ad-hoc queries, Web-based access, reporting, and backups from the primary database. This creates additional considerations should a failover occur. For example, a decision must be made if there is enough capacity to support both read-only work load that was running while it was in standby role and the additional work load will inherited should a failover occur and it becomes the primary production database.

Initial Connections:

Read-only queries and reporting applications can be directed to an Active Data Guard standby by creating a database service description that will start a read-only service anytime a database functions in the standby role. Because role-based services are controlled by the Data Guard Broker and CRS (Oracle Restart in the case of single instance databases), services will also be started/stopped appropriate for the database role whenever a Data Guard role transition occurs.

All applications that connect to a Data Guard configuration should connect using an Oracle Net alias that contains SCAN names for both the primary and standby databases in a `DESCRIPTION_LIST`. The Oracle Net alias used by read-write applications that connect to the database in the primary role should list the primary SCAN name in the first `DESCRIPTION` while the Oracle Net alias used by read-only applications that connect to the database in the standby role should have the standby SCAN name listed in the first `DESCRIPTION`.

Managing Existing Connections for Unplanned Outages

During steady state, the primary application will have connections to the primary database and the read-only reporting application will have connections to the active standby database. When a role transition occurs, reporting users connected to an active standby database will be disconnected as part of the failover process. Once the standby is transitioned to the primary database role, the primary database service will be started and users will automatically begin to connect to the new primary databases using the process described earlier in this paper.

Failover of the reporting application connections and services running on the standby database is performed manually. Following a failover, you first determine which database you wish to direct the reporting application. Your choices are as follows:

- If the new primary database has enough capacity to support both the primary application connections and the reporting application connections, then:
  - Manually start the reporting database service on the primary database.
  - Once the service is available, restart the reporting application to get connections established.
- If the primary database does not have enough capacity to support both the primary application and the reporting application, then start the reporting database service on other Active Data Guard standby databases.
  - If you do not have multiple Active Data Guard standbys in your Data Guard configuration, then reinstate the old primary database as a new synchronized standby database. This is a fast and simple process if Flashback Database was enabled prior to the failover and the cause of the outage does not prevent its use for re-instating the failed primary . Once the new standby is operational, start the read-only database service and connect your reporting application.

Additional Considerations for Data Guard SQL Apply

There are two types of Data Guard standby databases. The first type is a physical standby database that uses Redo Apply to maintain a block for block, exact replica of the primary database. Physical standby databases are the most popular type of Data Guard standby for data protection and availability due to the simplicity, high performance of Redo Apply. Active Data Guard also uses Redo Apply.

The second type of standby is a logical standby database. Logical standby uses SQL Apply and contains the same logical information as the primary database, although the physical organization and structure of the data can be different. SQL Apply is very useful for executing database rolling upgrades as it can support replication from a lower Oracle Database release or patchset, to a higher Oracle Database release or patchset, beginning from Oracle Database 10.1.3 onward. With the introduction of Transient Logical Database in Oracle Database 11g, SQL Apply can be used to execute database rolling upgrades using a physical standby database.[16]

Unlike Active Data Guard, role transitions for logical standby databases (SQL Apply) do not provide an ability to automatically disconnect attached sessions. When a failover or switchover occurs, read-only connections that existed on the logical standby will still be connected after the logical standby is converted into a primary. If the logical standby was being used to offload read-only reporting from the primary database or for other purposes, you will need to decide if there is sufficient capacity to meet service level objectives for all workloads after a failover or switchover has occurred. If not, the read-only sessions will need to be manually disconnected from the new primary as part of the switchover/failover process.

Client Transition during Switchover Operations

Up until this section, the focus of this paper has been automating client failover for unplanned outages when a Data Guard failover has occurred. In Data Guard, the term 'switchover' is used to describe a planned event where a primary and standby database switch roles, usually for the purpose of minimizing downtime while performing planned maintenance. The configuration best practices described in this paper to address unplanned failovers also address most of what is required for planned switchover as well, except for several additional manual steps that apply to logical standby databases (SQL Apply).

There are no additional considerations for switchovers using Active Data Guard. The following describes the switchover process for Data Guard 11g Release 2:

[16] http://www.oracle.com/technetwork/database/features/availability/maa-wp-11g-upgrades-made-easy-131972.pdf

Physical standby switchover:

4. First the primary database is converted to a standby database. The command to do so disconnects all sessions and brings the database to the mount state. The Data Guard Broker shuts down any read write service.
5. Client sessions receive a ORA-3113 and begin going through their retry logic (TAF for OCI and application code logic for JDBC)
6. The standby database is converted to a primary database and any existing sessions are disconnected. The Data Guard Broker shuts down read-only services
7. Read-only connections receive a ORA-3113 and begin going through their retry logic (TAF for OCI and application code logic for JDBC)
8. As the new primary and new standby are opened the respective services are started for each role and clients performing retries now see the services available and connect.

Logical standby switchover:

1. First insure that the proper reconnection logic has been configured as described in previous sections. For example, configure TAF and RETRY_COUNT for OCI applications and code retry logic for JDBC applications
2. Stop the services used by the primary application and the read only applications enabled on the standby database.
3. Disconnect or shutdown primary and read only application sessions
4. Once the switchover has completed restart the services used by the primary application and the read only application
5. Sessions that were terminated will reconnect once the service becomes available as part of the retry mechanism.
6. Restart the application in the case of an application shutdown

It's important to note that FAN is not needed to transition clients during a switchover operation as long as the application performs retries. FAN is only needed to break clients out of TCP timeout, a state that should only occur during unplanned outages.

Considerations for Prior Oracle Releases

Oracle Database 11g Release 2 greatly simplified client failover configuration and operation compared to previous Oracle releases. All steps needed to failover services to a new primary database, or start/stop services according to database role or notify clients to break them out of TCP timeout have been automated.

Previous to Oracle Database 11g Release 2 the following extra considerations were required:

1. Configure Oracle Net alias to include all node VIP names instead of SCAN names
2. Create database triggers to start/stop database services for the correct database role.
3. Configure a wrapper script and configuration file for the cfo executable for notification of JDBC clients.
4. Create a database trigger based on the DB_ROLE_CHANGE system event to execute the cfo wrapper script.

See Client Failover Best Practices[17] for a complete description of client failover best practices for Oracle Database 10g Release 2 and Oracle Database 11g Release 1.

Other areas to note:

- RETRY_COUNT is only available in Oracle Database 11g Release 2. Previous releases may need to specifically code additional retries for new connection attempts (number of times to go through the ADDRESS_LIST).
- Outbound connect time prior to Oracle Database 11g Release 2 can only be set in the sqlnet.ora. This means that all oracle net aliases inherit the same value. In Oracle Database 11g Release 2 it is set at the Oracle Net alias level

Conclusion

Oracle Database 11g Release 2 enhancements enable the transition of applications from a failed database to a new primary database after a Data Guard failover to be controlled by the same automation framework that exists for moving application connections from a failed node to surviving nodes within an Oracle RAC cluster. By combining this automation with Data Guard Fast-Start Failover, Oracle provide end-to-end automation that can make failover to a database at a remote site a true HA event.

[17] www.oracle.com/technetwork/database/features/availability/maa-wp-10gr2-clientfailoverbestprac-129636.pdf

Client Failover Best Practices for Highly
Available Oracle Databases: Oracle Database
11g Release 2
February 2011
Author: Michael T. Smith

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment