Oracle Maximum
Availability Architecture

An Oracle White Paper
August 2015

Oracle WebLogic Server and Highly Available Oracle Databases: Oracle Integrated Maximum Availability Solutions

Executive Overview

High availability and scalability are often the focus in different systems enabling customer applications with cost efficient solutions. There are various known issues in a heterogeneous complex environment involving Java EE middle tier and backend databases. For example, an application request may get blocked for a long period when a database node dies. There is no easy way to tell whether to obtain a fresh new connection after an application received the SQLException. The middle tier applications are unaware of new or restarted database nodes or it executes the work on a slow, hung or dead database node and they often have to rely on waiting for TCP/IP time-outs. Additionally, enterprise deployments need protection from unforeseen disasters and natural calamities.

Oracle WebLogic Server 11g and 12c provide strong support for integrating with the High Availability (HA) features of the Oracle database minimizing database access time while allowing transparent access to rich pooling management functions that maximizes both connection performance and application availability. Oracle Database offers HA capabilities such as Real Application Clusters (RAC), Oracle Data Guard, Oracle Restart, and etc., along with complementary products such as Oracle Golden Gate. As essential component of Oracle Maximum Availability Architecture (MAA)[1], Oracle WebLogic Server and Oracle Database's integrated high availability capabilities together deliver a robust solution that prevents, detects and recovers from unplanned outages within a small mean time to recovery.

Within the MAA solution-set, Oracle WebLogic Server and Oracle Data Guard work together enabling a comprehensive data protection solution that involves setting up a standby site at a geographically different location than the production site. The standby site may have equal or fewer services and resources compared to the production site. All data including application data, metadata, configuration data, and security data are replicated to the standby site. The standby site is normally in a passive mode; it is started when the production site is not available.

The Active Data Guard Option available with Oracle Database 11g Enterprise Edition enables you to open a physical standby database for read-only access for reporting, for simple or complex queries, sorting, or Web-based access while Redo Apply continues to apply changes from the production

[1] http://www.oracle.com/technology/deploy/availability/htdocs/maa.htm

database to the standby database. All queries reading from the physical standby database execute in real time, and return current results. With Active Data Guard, you can offload any operation that requires up-to-date, read-only access to the standby database, enhancing and protecting the performance of the production database without any compromise in Recovery Point or Recovery Time objectives.

In this paper, we investigate configuration best practices for Oracle WebLogic Server deployments against a highly available database.

Oracle WebLogic Server is a scalable, enterprise-ready Java Platform, Enterprise Edition application server and database interactions are typically handled by the middle tier's data source implementations. In Oracle WebLogic Server, configuring WebLogic Generic data sources, GridLink data source and multi data sources exposes database connectivity. These JDBC resources are then deployed, targeting servers or clusters in a WebLogic Server domain. Each data source that is configured contains a pool of database connections that are created when the data source instance is created, deployed, or targeted, or that are created at server startup.

Oracle WebLogic Server 11g and 12c RAC integration solutions have been jointly designed implemented and tested by Oracle MAA Development team. Oracle WebLogic Server is the only Application Server who has been fully integrated and certified with Oracle Database RAC 11g without losing any rich functionalities in Java EE implementation with security, transaction, connection pooling, etc management.

There are two data source implementations in Oracle WebLogic Server to support Oracle Real Application Clusters (RAC) and Oracle Data Guard:

- The multi data source solution which has been used successfully in customer production deployments since WebLogic Server 8.1 SP5

- The new implementation in Oracle WebLogic 11g Release 1 and 12c called Oracle WebLogic Active GridLink which is the market-leading mid-tier integration solution leveraging additional Oracle RAC advancements.

Oracle WebLogic Server Active GridLink for RAC is the strategic solution for supporting Oracle Real Application Clusters and Oracle Data Guard, as recommended by Oracle's MAA best practices. It represents the best possible middleware and database integration with features that are not available from other vendors.

The combination of Oracle WebLogic Server Data Source and Connection Pooling solutions and Oracle RAC provides a high-end mission-critical environment offering performance, high scalability and availability features. Load-balancing and Affinity capabilities offer significant performance improvement for online transaction processing scenarios, as well as improving throughput and total response time. Failover solution gives end-to-end rapid failure detection supporting graceful shutdown for planned and unplanned Oracle RAC node outages.

WebLogic Active GridLink fully supports Data Guard with the integration solution implemented using Oracle Universal Connection Pool.

This paper is based on Oracle WebLogic Server 11g (10.3.6) and Oracle Database 11g Release 2 (11.2), and describes the best practices to automatically transition application connections from a failed primary database to a new primary database. The configuration details with both Web Logic Active GridLink and Multi Data Source for working with Data Guard are included in this paper.

Introduction

Oracle WebLogic Server JDBC Data Sources

In WebLogic Server, you configure database connectivity by adding data sources to your WebLogic domain. WebLogic JDBC data sources provide database access and database connection management. Each data source contains a pool of database connections that are created when the data source is created and at server startup. Applications reserve a database connection from the data source by looking up the database source on the JNDI tree or in the local application context and then calling getConnection(). When finished with the connection, the application should call connection.close() as early as possible, which returns the database connection to the pool for other applications to use.

Types of WebLogic Server JDBC Data Sources

WebLogic Server provides three types of data sources:

- Generic Data Sources – Generic data sources and their connection pools provide connection management processes that keep the system running efficiently. The configuration options could be set in the data source to suit the applications and environment.
- GridLink Data Sources – Single data sources that adaptively respond to state changes based on Oracle Event Notification from Oracle RAC or Data Guard.
- Multi data sources – A multi data source is an abstraction around a group of generic data sources that provides load balancing or failover processing.

For additional information on WebLogic Server JDBC Data Sources, see *Oracle Fusion Middleware Configuring and Managing JDBC Data Sources for Oracle WebLogic Server* 11g Release 1 (10.3.5).[2]

Oracle WebLogic Server and RAC

Oracle WebLogic Server 10.3.4 introduced a single data source implementation to support an Oracle RAC cluster. It responds to FAN events to provide Fast Connection Failover (FCF), Runtime Connection Load-Balancing (RCLB), and RAC instance graceful shutdown. XA affinity is supported at the global transaction Id level. The new feature is called WebLogic Active GridLink for RAC; which is implemented as the GridLink Data Source within WebLogic Server.

The RAC integration capabilities of Universal Connection Pool (UCP) have been utilized by the WebLogic Server GridLink Data Source implementation to provide the FCF, RCLB and Affinity features.

[2] http://docs.oracle.com/cd/E21764_01/web.1111/e13737/jdbc_datasources.htm#i1204742

With the key foundation for providing deeper integration with Oracle RAC, this single data source implementation in Oracle WebLogic Server supports the full and unrestricted use of database services as the connection target for a data source. The active management of the connections in the pool is based on static settings configured on the connection pool itself (min/max capacity, timeouts, etc.) and real time information the connection pool receives from the Oracle Notification Service (ONS) subsystem that advises the "client" of any state changes within the RAC cluster.

Oracle WebLogic Server Database Persistence of Transaction Logs (TLOGs)

In WebLogic Server 10.3.6 and 12c, the capability of Database persistence of WebLogic transaction logs (TLOGs) is supported. This new feature adds additional flexibility that allows users to choose another persistent file store for primary TLOG instead of the default persistent file store. The user will have the flexibility to configure where their WebLogic server will store the Java Transaction API (JTA) primary TLOG. This configuration is per server. If the user selects the database store option, all JTA primary TLOG information, including transaction commit decision without Logging Last Resource (LLR) participant, resource check point, and server check point etc, will be stored in the database.

This provides a number of benefits:

1. Exploits the replication and other HA aspects inherent from the underlying database system.
2. Enhances handling of disaster recovery scenarios. With JMS, TLOGs and application in the same database and replication handled by Data Guard, there is no need to worry about cross site synchronization when compared to previous releases when additional care/solutions are required when one or more of this is on the storage systems such as a NAS.
3. Alleviates the need for a shared storage sub-system such as a NAS or a SAN. Usage of the database also reduces overall system complexity since in most cases a database is already present for normal runtime/application work.[3]

[3] This feature is similar to one in OC4J 10.1.3.0.

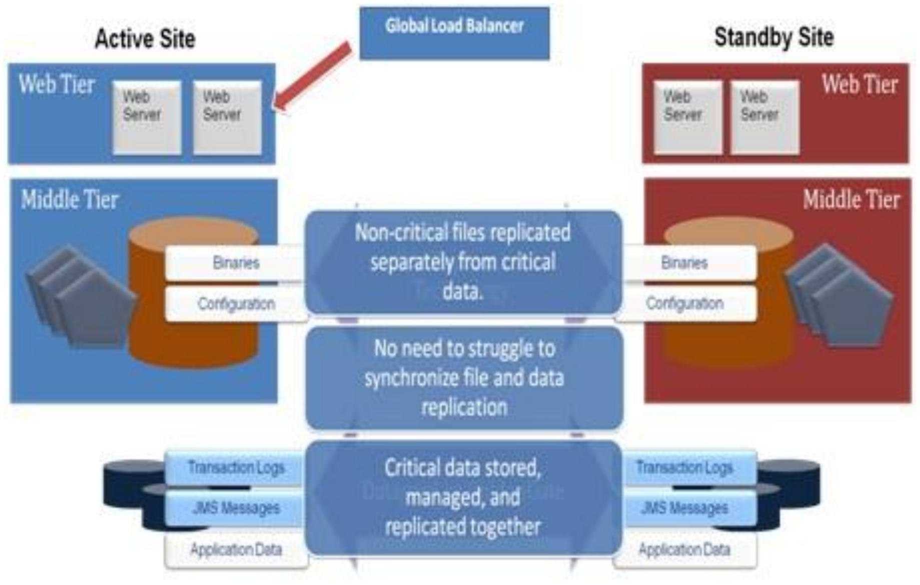

Global Load Balancer
Active Site
Standby Site
Web Tier
Web Server
Web Server
Middle Tier
Binaries
Configuration
Non-critical files replicated separately from critical data.
No need to struggle to synchronize file and data replication
Critical data stored, managed, and replicated together
Transaction Logs
JMS Messages
Application Data

Figure 2: Disaster Recovery Architectures with WebLogic DB TLOG and Data Guard

Oracle Database High Availability Solutions for Unplanned Downtime

Oracle Restart

Oracle Restart is a new feature in Oracle 11g Release 2 (11.2) that enhances the availability of a single-instance (non-clustered) Oracle Database and its components. Oracle Restart is used in single-instance environments only.

If you install Oracle Restart, it automatically restarts the database, the listener, and other Oracle components after a hardware or software failure or whenever the database's host computer restarts. It also ensures that the Oracle components are restarted in the proper order, in accordance with component dependencies.

Oracle Restart periodically monitors the health of components – such as SQL*Plus, the Listener Control utility (LSNRCTL), ASMCMD, and Oracle Data Guard – that are integrated with Oracle Restart. If the health check fails for a component, Oracle Restart shuts down and restarts the component. The Data Guard Broker requires that Oracle Restart be used by non-RAC databases in a Data Guard configuration to insure that the appropriate database services are active and the appropriate Fast Application Notification (FAN) events are published after role transitions.

Oracle Restart runs out of the Oracle Grid Infrastructure home, which you install separately from Oracle Database homes.

See:

- *Oracle Database Administrator's Guide* [4] for information about installing and configuring the Oracle Restart feature

Oracle Real Application Clusters

Oracle RAC and Oracle Clusterware allow Oracle Database to run any packaged or custom application across a set of clustered servers. This capability provides the highest levels of availability and the most flexible scalability. If a clustered server fails, then Oracle Database continues running on the surviving servers. When more processing power is needed, you can add another server without interrupting access to data.

Oracle RAC enables multiple instances that are linked by an interconnect to share access to an Oracle Database. In an Oracle RAC environment, Oracle Database runs on two or more systems in a cluster while concurrently accessing a single shared database. The result is a single database system that spans multiple hardware systems, enabling Oracle RAC to provide high availability and redundancy during failures in the cluster.

Oracle RAC One Node

Oracle Real Application Clusters One Node (Oracle RAC One Node) is a single instance of an Oracle RAC database that runs on one node in a cluster. Oracle RAC One Node enables better availability than cold failover for single-instance databases because of the Oracle technology called online database relocation, which intelligently migrates database instances and connections to other cluster nodes for high availability and load balancing. Online database relocation is performed using the Server Control Utility (SRVCTL).

Oracle Data Guard

Oracle Data Guard is the solution for Oracle data protection, data availability, and disaster recovery. Data Guard is an included feature of Oracle Database Enterprise Edition that provides the management, monitoring, and automation software to create and maintain one or more synchronized standby databases to protect Oracle data from failures, disasters, human error, and data corruptions.

Data Guard's loosely coupled architecture provides optimal data protection and availability:

- Database changes are transmitted directly from memory, isolating the standby from I/O corruptions that occur at the primary.

4 http://docs.oracle.com/cd/E11882_01/server.112/e25494/restart001.htm#ADMIN12709

- A standby database uses a different software code-path than the primary – isolating it from firmware and software errors that impact the primary database.
- Oracle corruption detection insures that data is logically and physically consistent before it is applied to a standby database.
- Data Guard detects silent corruptions caused by hardware errors (memory, cpu, disk, NIC) and data transfer faults, and prevents them from impacting the standby database.
- A standby database is used to perform planned maintenance in a rolling fashion, minimizing downtime and eliminating the risks inherent with introducing change to production environments.
- Data Guard can support a primary database and up to 30 standby databases in a single configuration. It is not unusual for customers to deploy a Data Guard configuration that includes both a local standby database for HA and zero data loss protection, and a remote standby database to protect against outages that can impact a large geography.

Oracle Active Data Guard

Oracle Active Data Guard 11g is an option of Oracle Database Enterprise Edition that extends basic Data Guard functionality. It allows a physical standby database to be open as read-only while changes are applied to it from the primary database. This increases performance and return on investment by offloading ad-hoc queries, Web-based access, reporting, and backups from the primary database while also providing disaster protection.

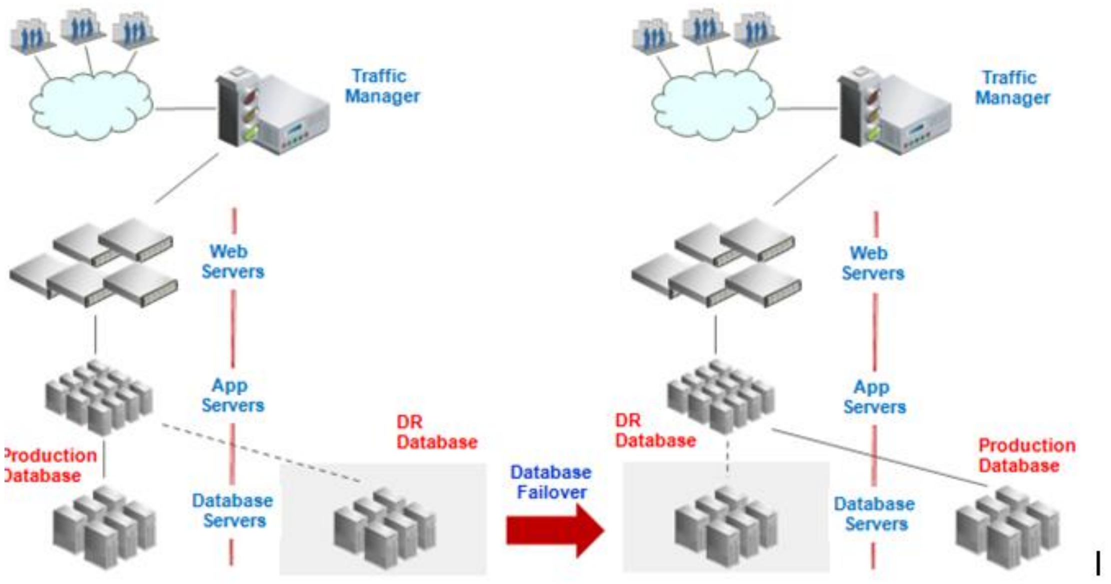
Oracle GoldenGate

Oracle GoldenGate is Oracle's strategic product for data distribution and data integration. It is a high-performance software application that uses log-based bidirectional data replication for real-time capture, transformation, routing, and delivery of database transactions across heterogeneous systems.

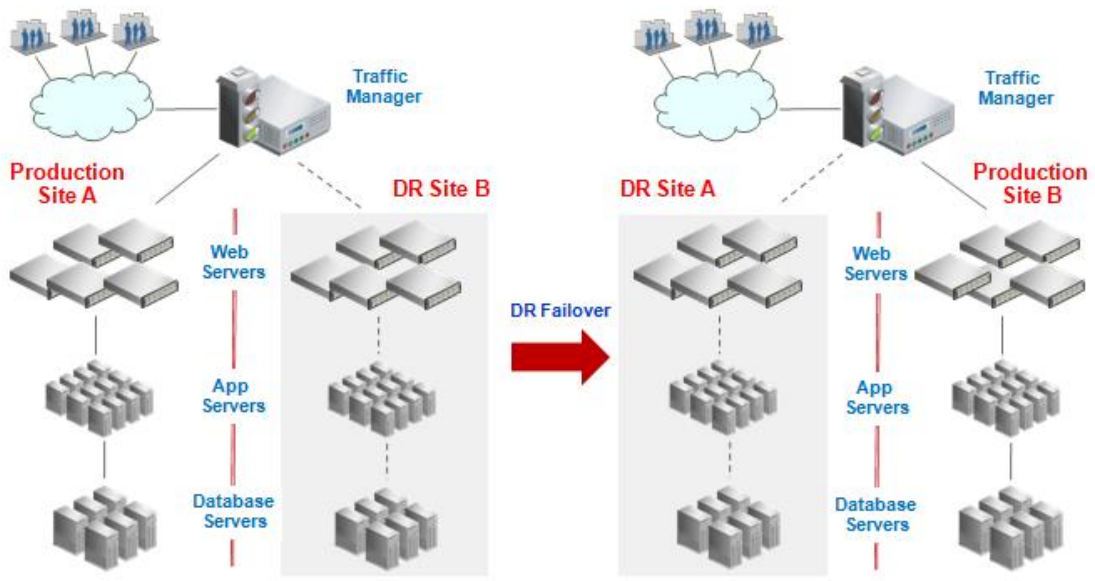
Configuring WebLogic Server Data Sources for Data Guard

Client Failover

There are several considerations when planning for client failover in a Data Guard configuration. In an environment with a local Oracle Data Guard standby database, the WebLogic Server middle tiers can connect to the primary or standby databases without any noticeable response time impact. In addition, a Data Guard database role transition (where a standby database becomes the new primary database) can be performed without restarting WebLogic Servers. The local standby database may be in the same data center, or across campus, or in the next town, as long as the network is able to handle the load and response time requirements of the application. This makes a local standby database ideal for maintaining availability during localized outages that only impact the primary database, or to facilitate planned maintenance with minimal downtime.



Contrast this with a configuration having only a remote standby database where the performance impact of network latency between primary and standby sites require a separate Oracle WebLogic Server middle tier to be co-located with the standby database. In this example, a Data Guard role transition does not require starting the Oracle WebLogic Servers in the standby site.



The two examples above illustrate different strategies for deploying a Data Guard standby database and WebLogic Server middle tiers that will determine how quickly users will be able to resume work after a failover has occurred.

The following sections describe how to configure WebLogic Server data sources for client failover for Oracle Database 11g Release 2 (11.2).

Prerequisites

The best practices described in this paper require that the Data Guard configuration be managed by the Data Guard Broker. The Data Guard Broker is the native Data Guard command line interface included with Oracle Database Enterprise Edition. The Data Guard Broker enables more efficient, centralized management of a Data Guard configuration from any database in a Data Guard configuration. The Data Guard Broker is also 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, eliminating any concern for availability being impacted by TCP/IP timeouts. 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.

Setup Procedures

This section provides step-by-step procedures to configure for client failover and set up WebLogic Server data sources and standby databases to minimize downtime during database outages. The following list presents a high-level description of the steps:

1. Create database services for database connections and configure the appropriate service attributes.
2. Ensure that the client-side Oracle Net configuration points to the service and includes all primary and standby listeners. This ensures that data sources can connect regardless of where the service is started.

Database Services and Application Notification Framework

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 notification framework that Oracle provides for single instance databases using Oracle Restart, for Oracle RAC databases, and for Oracle Data Guard. Please see the following for details on this framework: *Automatic Workload Management with Oracle Real Application Clusters* 11g Release 2[6] and the *Oracle Real Application Clusters Administration and Deployment Guide*[7].

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

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

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

Role-Based Database Services

Beginning with Data Guard 11g Release 2, you can automatically control the startup of database services on primary and standby database by assigning a database role `[ -l {[PRIMARY] | [PHYSICAL_STANDBY] | [LOGICAL_STANDBY] | [SNAPSHOT_STANDBY]}]` to each service. 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 it 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).

Configuring JDBC Database Services

Do not enable TAF on services for JDBC database services that are FAN enabled. The following example illustrates how to create database services for JDBC database services. 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 read/write workload service (oltpworkload) that the WLS data source 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 – JDBC r/w service:
   ```
   srvctl add service –d Austin –s oltpworkload –r ssa1,ssa2
   –l PRIMARY –q FALSE –e NONE –m NONE –w 0 –z 0
   ```

   Standby cluster – JDBC r/w service:
   ```
   srvctl add service –d Houston –s oltpworkload –r
   ssb1,ssb2 –l PRIMARY –q FALSE –e NONE –m NONE –w 0 –z 0
   ```

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

   Primary cluster – JDBC r/o service:
   ```
   srvctl add service –d Austin –s reports –r ssa1,ssa2 –l
   PHYSICAL_STANDBY –q FALSE –e NONE –m NONE –w 0 –z 0
   ```

   Standby cluster – JDBC r/o service:
   ```
   srvctl add service –d Houston –s reports –r ssb1,ssb2 –l
   PHYSICAL_STANDBY –q FALSE –e NONE –m NONE –w 0 –z 0
   ```

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_names => ‘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’);
```

3. Modify the service for the appropriate service goals.
   Run-time connection load balancing requires configuring Oracle RAC Load Balancing Advisory with service-level goals for each service for which load balancing is enabled. The Oracle RAC Load Balancing Advisory may be configured for SERVICE_TIME or THROUGHPUT. The connection load balancing goal should be set to SHORT.

   Oracle RAC Load Balancing Advisory configured for SERVICE_TIME:
   ```
   srvctl modify service –d <database name>  –s <service
   name> –B SERVICE_TIME –j SHORT
   ```

   Oracle RAC Load Balancing Advisory configured for THROUGHPUT:
   ```
   srvctl modify service –d <database name>  –s <service
   name> –B THROUGHPUT –j SHORT
   ```

Configure Oracle WebLogic Server Data Sources

There are three types of data sources available for the configuration. The generic data source is the implementation for single database access. The multi data source is the native WebLogic middle tier implementation for Oracle RAC integration, which does not leverage Oracle Notification Service. The GridLink data source is the new Active GridLink implementation which takes advantages of Oracle RAC supporting Fast Connection Failover (FCF), Runtime Connection Load-Balancing (RCLB), and Affinities. It supports using Single Client Access Name (SCAN) for the configuration.

WebLogic Active GridLink for Data Guard

WebLogic Active GridLink fully supports Data Guard with the integration solution implemented with Oracle Universal Connection Pool. WebLogic Active GridLink has the following characteristics:

- Integrates with usage of FAN
- Can consume Data Guard specific FAN events
- Integrates with best practices for SCAN addresses with Oracle Database 11.2

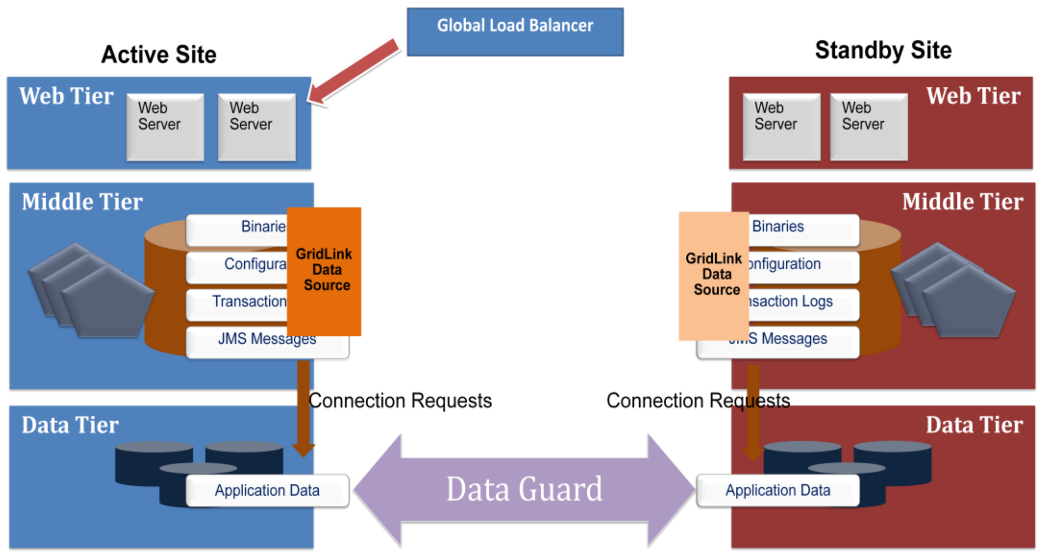


Figure 3: WebLogic Active GridLink for Data Guard

The configuration is performed by specifying two SCAN address in the connection url, one that represents the primary data center and the other representing the standby data center. The backend has the knowledge of which address is for the primary and which is for the standby. The failure events are notified to the middle tier through the Oracle Notification Service (ONS).

Sample Read Write Connection url for Primary and Standby RAC 11gR2:

```
jdbc:oracle:thin:@(DESCRIPTION_LIST = (LOAD_BALANCE = off)
  (FAILOVER = on)(DESCRIPTION = (CONNECT_TIMEOUT =
  10)(TRANSPORT_CONNECT_TIMEOUT = 3)(RETRY_COUNT = 3)
  (ADDRESS_LIST = (LOAD_BALANCE = on)(ADDRESS =
     (PROTOCOL = TCP)(HOST = PRMY_SCAN)(PORT =
      1521)))(CONNECT_DATA = (SERVICE_NAME =
      oltpworkload)))(DESCRIPTION = (CONNECT_TIMEOUT =
      10) (TRANSPORT_CONNECT_TIMEOUT = 3)(RETRY_COUNT = 3)
  (ADDRESS_LIST = (LOAD_BALANCE = on)(ADDRESS = (PROTOCOL =
      TCP)(HOST = STBY_SCAN)(PORT = 1521)))(CONNECT_DATA =
     (SERVICE_NAME = oltpworkload))))
```

When a new connection is made using the above connection url, the following logic is used:

a) Oracle Net contacts DNS and resolves `PRMY_SCAN` to a total of 3 IP addresses.
b) Oracle Net randomly picks one of the 3 IP addresses 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 (the initial attempt plus the `RETRY_COUNT` in the above example).

c) If the connection to the primary site is unsuccessful, it then contacts DNS and resolves `STBY_SCAN` to 3 IP addresses.

d) The same sequence is performed for the standby `STBY_SCAN` that was performed for the `PRMY_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*[8] technical whitepaper.

Sample Read Only Connection url for Primary and Standby RAC 11gR2 and Active Data Guard:

```
jdbc:oracle:thin:@(DESCRIPTION_LIST = (LOAD_BALANCE = off)
  (FAILOVER = on)(DESCRIPTION = (CONNECT_TIMEOUT =
  10)(TRANSPORT_CONNECT_TIMEOUT = 3)(RETRY_COUNT = 3)
  (ADDRESS_LIST = (LOAD_BALANCE = on) (ADDRESS = (PROTOCOL =
    TCP)(HOST = STBY_SCAN)(PORT = 1521)))(CONNECT_DATA =
   (SERVICE_NAME = reports)))(DESCRIPTION = (CONNECT_TIMEOUT =
   10)(TRANSPORT_CONNECT_TIMEOUT = 3)(RETRY_COUNT = 3)
  (ADDRESS_LIST = (LOAD_BALANCE = on)(ADDRESS = (PROTOCOL =
    TCP)(HOST = PRMY_SCAN)(PORT = 1521)))(CONNECT_DATA =
    (SERVICE_NAME = reports))))
```

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

Sample Connection url for Primary RAC 11gR2 and Standby Single Instance with 11gR2 Oracle Restart:

```
jdbc:oracle:thin:@(DESCRIPTION_LIST = (LOAD_BALANCE = off)
  (FAILOVER = on)(DESCRIPTION = (CONNECT_TIMEOUT =
  10)(TRANSPORT_CONNECT_TIMEOUT = 3)(RETRY_COUNT = 3)
  (ADDRESS_LIST = (LOAD_BALANCE = on) (ADDRESS =
    (PROTOCOL=TCP)(HOST = PRMY_SCAN)(PORT =
    1521)))(CONNECT_DATA = (SERVICE_NAME =
    oltpworkload)))(DESCRIPTION = (ADDRESS_LIST =
    (ADDRESS =
    (PROTOCOL = TCP)(HOST = STBYHOST)(PORT =
    1521)))(CONNECT_DATA = (SERVICE_NAME = oltpworkload))))
```

Sample Connection url for Primary and Standby Single Instance with 11gR2 Oracle Restart:

```
jdbc:oracle:thin:@(DESCRIPTION_LIST = (LOAD_BALANCE = off)
  (FAILOVER = on)(DESCRIPTION =(ADDRESS_LIST = (ADDRESS =
  (PROTOCOL = TCP)(HOST = PRMYHOST)(PORT =
  1521)))(CONNECT_DATA = (SERVICE_NAME =
  oltpworkload)))(DESCRIPTION = (ADDRESS_LIST = ADDRESS =
  (PROTOCOL = TCP)(HOST = STBYHOST)(PORT =
  1521)))(CONNECT_DATA = (SERVICE_NAME = oltpworkload))))
```

To create a JDBC GridLink data source:

1. If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit**.
2. In the **Domain Structure** tree, expand **Services → Data Sources**.
3. On the Summary of JDBC Data Sources page, click **New → GridLink Data Source**.
4. On the JDBC Data Source Properties page, enter or select the following information:
   - Name – Enter a name for this JDBC data source. This name is used in the configuration file (config.xml) and throughout the Administration Console when referring to this data source.
   - JNDI Name – Enter the JNDI path to where this JDBC data source will be bound. Applications look up the data source on the JNDI tree by this name when reserving a connection.
   - Database Type – Select **Oracle**.
   - XA Driver – Leave unselected.

   Click **Next** to continue.

5. On the Transaction Options page, follow these steps:
   - Select Global Transaction options:
     - **Supports Global Transactions** – Select this option (the default) to enable global transaction support in this data source. In most cases, you should leave the option selected.
   - If you select Supports Global Transactions, select an option for transaction processing:
     - **One-Phase Commit** – Select this option to enable the non-XA connection to participate in a global transaction as the only transaction participant.

   Click **Next** to continue.
6. On the GridLink data source connection Properties Options page, select **Enter Complete JDBC URL**.
   Click **Next** to continue.
7. On the Connection Properties page, enter values for the following properties
   Complete JDBC URL – Enter the URL.
   - For Primary and Standby RAC 11gR2, enter:

     ```
     jdbc:oracle:thin:@(DESCRIPTION_LIST = (LOAD_BALANCE =
     off)(FAILOVER = on)(DESCRIPTION = (CONNECT_TIMEOUT =
     10)(RETRY_COUNT = 3)(ADDRESS_LIST =  (LOAD_BALANCE =
     on)(ADDRESS = (PROTOCOL = TCP)(HOST =  PRMY_SCAN)(PORT
     = 1521)))(CONNECT_DATA =
     (SERVICE_NAME=oltpworkload)))(DESCRIPTION =
     (CONNECT_TIMEOUT=10)(RETRY_COUNT=3)
     (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST =
     STBY_SCAN)(PORT=1521)))                  (CONNECT_DATA
     = (SERVICE_NAME = oltpworkload))))
     ```

     The JDBC URL should contain SCAN names for both the primary and standby databases in a `DESCRIPTION_LIST`.

   - For Primary RAC 11gR2 and Standby Single Instance, enter:

     ```
     jdbc:oracle:thin:@(DESCRIPTION_LIST = (LOAD_BALANCE =
     off)(FAILOVER = on)(DESCRIPTION =
     (CONNECT_TIMEOUT=10)(RETRY_COUNT=3) (ADDRESS_LIST =
     (LOAD_BALANCE = on) (ADDRESS = (PROTOCOL = TCP)(HOST =
     PRMY_SCAN)(PORT = 1521)))(CONNECT_DATA = (SERVICE_NAME
     = oltpworkload)))(DESCRIPTION =
     (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)(HOST =
     STBYHOST)(PORT = 1521)))(CONNECT_DATA = (SERVICE_NAME
     = oltpworkload))))
     ```
   - For Primary Single Instance and Standby Single Instance with Oracle Restart, enter:

     ```
     jdbc:oracle:thin:@(DESCRIPTION_LIST = (LOAD_BALANCE =
     off)(FAILOVER = on)(DESCRIPTION = (ADDRESS_LIST =
     ```

```
(ADDRESS = (PROTOCOL = TCP)(HOST =
PRMYHOST)(PORT=1521)))(CONNECT_DATA = (SERVICE_NAME =
oltpworkload)))(DESCRIPTION = (ADDRESS_LIST = (ADDRESS
= (PROTOCOL = TCP)(HOST = STBYHOST)(PORT =
1521)))(CONNECT_DATA = (SERVICE_NAME =
oltpworkload))))
```

- **Database User Name** – Enter the database user account name that you want to use for each connection in the data source.
- **Password/Confirm Password** – Enter the password for the database user account.

Click **Next** to continue.

8. On the Test Database Connection page, review the connection parameters and click **Test All Listeners**.
    - **Test Table Name** - For Active Data Guard, the SQL statement that we would like to use to test database connections should be such that it fails for the current standby database. A lightweight statement that performs an update on a single row, single column table is best.

    Click **Next** to continue.
9. On the ONS Client Configuration page, enter values for the following properties:
    - Select **Fan Enabled** to subscribe to Oracle FAN Events.
    - Enter the host and port for each ONS node separated by colon and click the add button. For example:
      ```
      PRMYHOST:6200
      STBYHOST:6200
      ```
      Click **Next** to continue.
10. On the Test ONS client configuration page, review the parameters and click **Test All ONS Nodes**. To test individual nodes, click **Test ONS Node** for an ONS host and port.
    Click **Next** to continue.
11. On the Select Targets page, select the servers or clusters on which you want to deploy the data source.
12. Click **Finish** to save the JDBC data source configuration and deploy the data source to the targets that you selected.
13. To activate these changes, in the Change Center of the Administration Console, click **Activate Changes**.

Multi Data Source

WebLogic Multi Data Source supports Oracle Data Guard as follows:

- WebLogic Multi Data Source does not use FAN, it uses a polling mechanism instead. It automatically re-enable the recovery of a failed data source within a multi data source. Frequency of these tests is controlled by the Test Frequency Seconds attribute of the multi data source. The default value for Test Frequency is 120 seconds. If a specific value is not set, then the Multi Data Source will test disabled data sources every 120 seconds. The desired effect can be achieved by using the Administration Console to set the Multi Data Source Test Frequency interval to 5 seconds after the Multi Data Source is created.

- It contains data sources for both primary and secondary sites.
- This support requires database service available at primary site only.
- The connect timeout at each data source level needs to be configured.

JDBC Multi Data Source with Load Balancing Policy

- In the Algorithm Type – Select Load-Balancing for RAC database and Failover for Single Instance database.
- Change the Test Frequency interval for the Multi Data Source from 120 to 5 seconds.

1. Create JDBC data sources, as follows:
   a) If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit**.
   b) In the **Domain Structure** tree, expand **Services → Data Sources**.
   c) On the Summary of JDBC Data Sources page, click **New → Generic Data Source**.
   d) On the JDBC Data Source Properties page, enter or select the following information:
      - Name – Enter a name for this JDBC data source. This name is used in the configuration file (config.xml) and throughout the Administration Console when referring to this data source.
      - JNDI Name – Enter the JNDI path to where this JDBC data source will be bound. Applications look up the data source on the JNDI tree by this name when reserving a connection.
      - Database Type – Select **Oracle**.
      - Oracle Driver – Select the JDBC driver you want to use to connect to the database.
      - For Single Instance Database – select **Oracle's Driver (Thin) for Service connections; Versions: 9.0.1 and later**.
      - For RAC Database – select **Oracle's Driver (Thin) for RAC Service-Instance connections; Versions: 10 and later**.

      Click **Next** to continue.
   e) On the Transaction Options page, follow these steps:
      - Select Global Transaction options:
        - **Supports Global Transactions** – Select this option (the default) to enable global transaction support in this data source. In most cases, you should leave the option selected.

          If you select Supports Global Transactions, select an option for transaction processing:

- **One-Phase Commit** – Select this option to enable the non-XA connection to participate in a global transaction as the only transaction participant.

Click **Next** to continue.

f) On the Connection Properties page, enter values for the following properties:
   - **Service Name (**Only displayed if **Oracle's Driver (Thin) for RAC Service-Instance connections; Versions: 10 and later** was selected for **Oracle Driver**) – Enter the database service (service_names initialization parameter) to which you want to connect.
   - **Database Name**
     - For Single Instance Database – enter the database service (service_names initialization parameter) to which you want to connect.
     - For RAC Database – enter the database instance (instance_name initialization parameter) to which you want to connect.
   - **Host Name** – Enter the DNS name of the server that hosts the database.
   - **Port** – Enter the port on which the database server listens for connection requests.
   - **Database User Name** – Enter the database user account name that you want to use for each connection in the data source.
   - **Password/Confirm Password** – Enter the password for the database user account.

   Click **Next** to continue.

g) On the Test Database Connection page, review the connection parameters and click **Test Configuration**.
   - **Test Table Name** - For Active Data Guard, the SQL statement that we would like to use to test database connections should be such that it fails for the current standby database. A lightweight statement that performs an update on a single row, single column table is best.

   Click **Next** to continue.

h) On the Select Targets page, select the servers or clusters on which you want to deploy the data source.

i) Click **Finish** to save the JDBC data source configuration and deploy the data source to the targets that you selected.

j) To activate these changes, in the Change Center of the Administration Console, click **Activate Changes**.

2. Create a JDBC multi data source, as follows:
   a) If you have not already done so, in the Change Center of the Administration Console, click **Lock & Edit**.
   b) In the Domain Structure tree, expand **Services → Data Sources**.
   c) On the Summary of JDBC Data Sources page, click **New → Multi Data Source**.

d) On the Configure the Multi Data Source page, enter or select the following information:
   - Name – Enter a unique name for this JDBC multi data source. This name is used in the configuration file (config.xml and the JDBC module) and throughout the Administration Console when referring to this data source.
   - JNDI Name – Enter the JNDI path to where this JDBC data source will be bound. Applications look up the data source on the JNDI tree by this name when reserving a connection.
   - Algorithm Type – Select an algorithm option: Select **Load-Balancing for RAC database and Failover for Single Instance database**.
     - Failover – The multi data source routes connection requests to the first data source in the list; if the request fails, the request is sent to the next data source in the list, and so on.
     - Load-Balancing – The multi data source distributes connection requests evenly to its member data sources.

e) On the Select Targets page, select the servers or clusters on which you want to deploy the multi data source.

   The targets you select will limit the data sources that you can select as part of the multi data source. You can only select data sources that are deployed to the same targets as the multi data source.

f) On the Select Data Source Type page, select **Non-XA Driver**. The multi data source will only use data sources that use a non-XA JDBC driver to create database connections.

g) On the Add Data Sources page, select the data sources that you want the multi data source to use to satisfy connection requests.
   - For a Single Instance Database – select the data sources for all databases (SSA1 and SSB1).
   - For a RAC Database – select the data sources for all databases (SSA1, SSA2, SSB1, and SSB2).

h) Click **Finish** to save the JDBC multi data source configuration and deploy the data source to the targets that you selected.

i) To activate these changes, in the Change Center of the Administration Console, click **Activate Changes**.

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 fail over services to a new primary database, start/stop services according to database role, or notify clients to break them out of TCP timeout, have been automated.

Prior to Oracle Database 11g Release2, the following extra considerations were required:

1. Create database triggers to start/stop database services for the correct database role.
2. Configure a wrapper script and configuration file for the cfo executable for notification of JDBC clients.

3. Create a database trigger based on the DB_ROLE_CHANGE system event to execute the cfo wrapper script.

For a complete description of previous releases, see Client Failover Best Practices for Oracle Database 10g Release 2 and Oracle Database 11g Release 1[9].

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 (the number of times to go through the ADDRESS_LIST).
- The outbound connect timeout prior to Oracle Database 11g Release 2 can only be set in the SQLNET.ORA file. 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 WebLogic and Oracle High Available Database are designed to work together enabling mission critical solutions for high availability, scalability and enterprise deployment protection involving disaster recovery; as well as high performance business requirements.

Oracle WebLogic Server Active GridLink for RAC provides the best available support for the Real Application Clusters (RAC) features in Oracle Database 11g. It's the recommended solution for working with Oracle Data Guard.

References

Oracle Database 11gR2 documentation[10]

Oracle WebLogic Server 11g documentation on Data Source configuration[11]

Single Client Access Name White Paper[12]

Oracle RAC One Node White Paper[13]

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

[10] http://www.oracle.com/pls/db112/homepage

[11] http://docs.oracle.com/cd/E21764_01/web.1111/e13737/toc.htm

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

[13] http://www.oracle.com/technetwork/database/clustering/overview/twp-rac1nodev1-1-130698.pdf

How-to Configure Oracle WebLogic Server with GridLink Data Source[14]

Client Failover Best Practices for Highly Available Oracle Databases: Oracle Database 11g Release 2[15]

Oracle WebLogic Server Active GridLink for Real Application Clusters[16]

Oracle WebLogic Server Active GridLink Demo[17]

[14] http://www.oracle.com/technetwork/middleware/weblogic/wls-jdbc-gridlink-howto-333331.html

[15] http://docs.oracle.com/cd/E21764_01/web.1111/e13737/toc.htm

[16] http://www.oracle.com/technetwork/middleware/weblogic/gridlink-rac-wp-494900.pdf

[17] http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/513398.mp4

ORACLE®

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

Oracle WebLogic Server Data Sources for Highly Available Databases
August 2015
Author: Susan Kornberg, Frances Zhao
Contributing Authors: Michael T. Smith, Pradeep Bhat

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

Hardware and Software, Engineered to Work Together