An Oracle White Paper
June 2014

WebLogic Active GridLink: Intelligent integration between WebLogic Server and Oracle Database Real Application Clusters

Introduction

High availability and scalability are often the focus in different systems enabling customer applications with cost efficient solutions. Users who want their systems to be always ready to serve them need high availability. A system that is highly available is designed to provide uninterrupted computing services during essential time periods, during most hours of the day, and most days of the week throughout the year; this measurement is often shown as 24x365. Such systems also need a high availability solution for planned maintenance operations such as upgrading a system's hardware or software.

There are various known issues in a heterogeneous complex environment involving both 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. Additional, it is complex for application developers to mask outages of a database session (instance, node, storage or network or any other related component) and as a result errors and timeouts are often exposed to the end users leading to user frustration, lost productivity, and lost opportunities.

Oracle Real Application Clusters (RAC) provides highly scalable and available database solutions for all your business applications. It provides unbeatable fault tolerance, performance, and scalability with no application changes necessary. In Oracle Database 12c, there are solutions introduced not only for high availability, but also for automatic recovery and cloud integration.

Oracle WebLogic Server 12c (12.1.2 and 12.1.3) provides strong support for the features in Oracle Database 12c, minimizing database access time while allowing transparent access to rich pooling management functions that maximizes both connection performance and availability. There are also improvements for better user experience when handling both unplanned outages and planned outages; and guaranteed transaction outcome.

Oracle WebLogic Active GridLink for RAC 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 recommended by Oracle[1]. It represents the best possible middleware and database integration with features that are not available from other vendors. WebLogic Server is the only Application Server who has been fully integrated and certified with Oracle Database RAC 11g and 12c without losing any capabilities in Java EE implementation with security, transaction, connection pooling, etc management.

The combination of Oracle WebLogic Server Data Source and Connection Pooling solutions and Oracle Database, 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.

[1] Please refer to the Oracle Fusion Middleware Licensing Information documentation for Active GridLink features

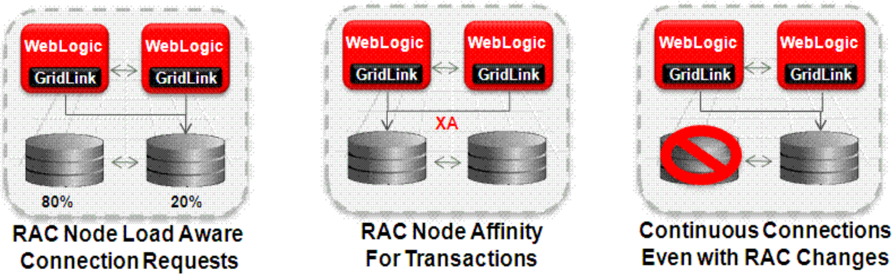

WebLogic
GridLink
WebLogic
GridLink
80%
20%
RAC Node Load Aware Connection Requests
WebLogic
GridLink
WebLogic
GridLink
XA
RAC Node Affinity For Transactions
WebLogic
GridLink
WebLogic
GridLink
Continuous Connections Even with RAC Changes

Figure 1: WebLogic Active GridLink for RAC

Beginning with Oracle Database Release 12c Release 1 (12.1), Oracle Database supports Application Continuity for Java. This feature supports the concept of commit outcome and recovery of active requests allowing JDBC-based applications in high availability (HA) infrastructures to avoid application errors from system failures by providing an immediate implicit recovery and resubmission of all active requests as soon as failover occurs and connectivity is restored to database services.

- Replays in-flight work on recoverable errors
- Masks many hardware, software, network, storage errors and outages
- Improves end user experience and developer productivity

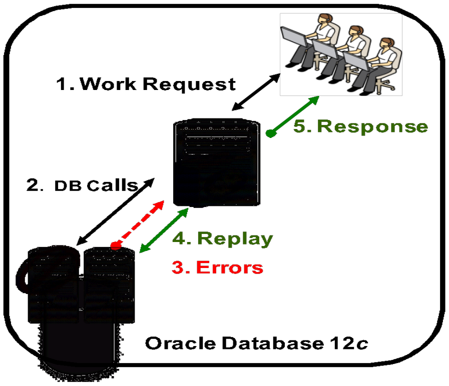


Figure 2: User experience when using Application Continuity

The Transaction Guard has also been introduced in Oracle Database 12c which is a reliable protocol and tool that returns the outcome of the last in-flight transaction after an outage that makes the database session unavailable.

As another cutting edge high performance solution, Database Resident Connection Pool has shown great performance results in different enterprise and mission critical scenarios that simply are a connection pool in the server and it can be shared across many clients.

Oracle WebLogic Server 12c (12.1.2 and 12.1.3) and Oracle Database 12c are fully certified to work together providing high-availability, scalability and high performance. WebLogic Server is the #1 Application Server as shown in the performance testing with SpecJ; and it has even greater performance results with Oracle Database end-end scenarios.

In this article, we start with a brief introduction to Oracle RAC and an overview of the Oracle RAC and Database features supported in Oracle Database 12c and Oracle WebLogic Server 12c (12.1.2 and 12.1.3). We then focus on details on each integrated scenarios with Oracle WebLogic 12c (12.1.2 and 12.1.3) and Oracle Database 12c. All contents covered in this paper also apply to Oracle WebLogic Server 12.1.3. The technical details about Runtime Connection Load Balancing, XA Affinity, Web Session Affinity, Fast Connection Failover, and how to remove and add the additional RAC node with zero-downtime will be discussed with different use cases. All the new Oracle Database 12c capabilities integrated with WebLogic Server, such as Application Continuity, Transaction Guard, Database Resident Connation Pool and Pluggable Database, Global Database Services will be covered in details as well.

.

Oracle Real Application Clusters

Oracle RAC enables you to cluster Oracle databases. Single-instance Oracle databases have a one-to-one relationship between the Oracle database and the instance. Oracle RAC environments have a one-to-many relationship between the database and the instance.

The key of high availability and scalability is how to manage workloads in Oracle Real Application Clusters for your applications. The most critical element to manage workloads or a group of applications is to define services that you assign to a particular application or to a subset of an application's operations. The work also can be grouped by type under services. For example, online users can use one service, while batch processing can use another and reporting can use yet another service to connect to the database.

Figure below shows how Oracle RAC is the Oracle Database option that provides a single system image for multiple servers to access one Oracle database. In Oracle RAC, each Oracle instance usually runs on a separate server.

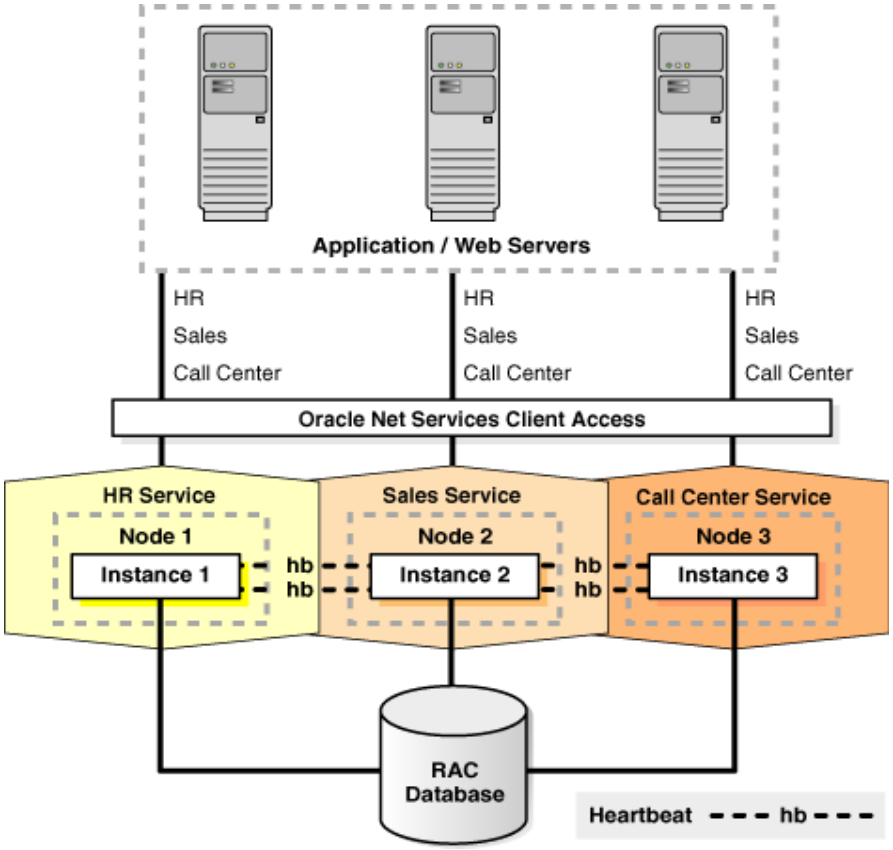


Figure 3: Oracle Real Application Clusters

New Features in Oracle Database 12c

Oracle Database 12c is one of most successful major release with great feature set that not only is the key component for Oracle private and public cloud, but also it brings the Data solution to global! Oracle WebLogic Server 12c (12.1.2 and 12.1.3) has tight integration with Oracle Database 12c with the advantages of all new net capabilities.

- Application Continuity, Application Continuity is invoked for outages that result in recoverable errors, typically related to underlying software, foreground, hardware, communications, network, or storage layers. Application Continuity is used to improve the user experience when handling both unplanned outages and planned outages. .
- Transaction Guard, Transaction Guard with Oracle Database 12c is a reliable protocol and tool that returns the outcome of the last in-flight transaction after an outage that makes the database session unavailable. Without Transaction Guard, applications and users who attempt to retry operations following an outage can cause logical corruption by committing duplicate transactions or committing transactions out of order.
- Database Resident Connection Pool, The Database Resident Connection Pool implementation creates a pool on the server side, which is shared across multiple client pools. It significantly lowers memory consumption on the server because of reduced number of server processes on the server and increases the scalability of the Database server.
- Oracle Database consolidation and Pluggable Database, Database consolidation provides the ability to host multiple applications or databases on the same system platform or within the same database. Pluggable Database is the capability that enables an Oracle database to contain a portable set of schemas, objects, and related structures that appears logically to an application as a separate database.
- Global Data Services, Global Data Services enables administrators to automatically and transparently manage client workloads across replicated databases that offer common services. A database service is a named representation of one or more database instances. Services enable you to group database workloads and route a particular work request to an appropriate instance. A global service is a service provided by multiple databases synchronized through data replication.
- Rolling upgrade with Oracle Data Guard, Rolling Upgrades using Oracle Active Data Guard provides new PL/SQL packages that automate much of the process of performing a database rolling upgrade (to a later Oracle Database release or to a new patch set, or when performing other database maintenance) using a physical standby database. You input an upgrade plan and PL/SQL packages automate three phases of the upgrade according to that plan: start, switchover, and finish.

Oracle WebLogic Server 12c (12.1.2 and 12.1.3) and Oracle Database 12c are fully certified to work together providing high-availability, scalability and high performance. The Oracle Database solutions, such as Application Continuity, Transaction Guard, Database Resident Connection Pool, etc are available via Oracle WebLogic Server JDBC data source and connection pool implementations.

Oracle WebLogic Server and Oracle Database RAC

In Java EE Application Servers, database interactions are typically handled by data source implementations. You configure and expose a connection to databases as JDBC data sources.

There are two data source implementations in Oracle WebLogic Server to support Oracle Real Application Clusters (RAC): the multi data source solution which has been used successfully in customer production deployments and the new implementation in Oracle WebLogic starting with 11*g* Release 1 (10.3.4) and continually with more enhancements in 12c (12.1.1, 12.1.2 and 12.1.3), called Oracle WebLogic Active GridLink for RAC which is the market-leading mid-tier integration solution leveraging the latest and greatest Oracle Database and RAC advances.

With Active GridLink for RAC solution, Oracle WebLogic Server 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 Web Session Affinity is supported within a HTTP session scope.

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.

With the key foundation for providing deeper integration with Oracle Database 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 RAC Oracle Notification Service (ONS) subsystem that advises the “client” of any state changes within the RAC cluster.

The Universal Connection Pool Java library has been integrated with WebLogic Server and been utilized by WebLogic GridLink data source implementation to provide the Fast Connection Failover, Runtime Connection Load Balancing and Affinity features.

Oracle Database services are logical abstractions for managing workloads in Oracle Database. Services divide workloads into logically disjoint groupings. Each service represents a workload with common attributes, service-level thresholds, and priorities. Services are built into the Oracle Database, providing a single system image for workloads, prioritization for workloads, performance measures for real transactions, and alerts and actions when performance goals are violated. Services enable database administrators to configure a workload, administer it, enable/disable it, and measure workload as a single entity.

The GridLink Data Source is associated with a connection pool, which contains a set of heterogeneous connections to the RAC instances that are hidden behind the database service. When an application requests a connection from the data source, a suitable connection is borrowed from the pool and supplied to the application based on the load balancing information the connection pool has received and the current distributions of connections in use from the pool.

Active GridLink for RAC simplifies the use of Oracle RAC database with WebLogic Server through the single Data Source approach, which in turn reduces the configuration and management complexity required to use Oracle RAC. Note that utilization of Multi Data Source configurations for RAC environments will continue to be supported. Upgrades from RAC Multi Data Sources to Grid Link Data Sources are straight-forward and involve creating a single Grid Link Data Source with the same JNDI name as the Multi Data Source, which reduces the number of configuration artifacts to maintain.

In WebLogic Serer 12c (12.1.2 and 12.1.3), Active GridLink has added new capability of Oracle Database 12c integration, including cutting edge functionalities of Application Continuity, Transaction Guard, Database Resident Connection Pool, Pluggable Database and Global Data Services, etc.

Fast Connection Failover

The Fast Connection Failover (FCF) feature is a Fast Application Notification (FAN) client implemented through the Universal Connection Pool. The feature requires the use of an Oracle JDBC driver and an Oracle RAC database or use of Oracle Restart on a single instance database.

WebLogic GridLink Data Source has been integrated with FCF from Universal Connection Pool implementation and uses FCF to:

- Provide rapid failure detection
- Abort and remove invalid connections from the connection pool quickly
- Perform graceful shutdown for planned and unplanned Oracle RAC node outages
- Adapt to changes in topology, such as adding or removing a node
- Distribute runtime work requests to all active Oracle RAC instances, including those rejoining a cluster

ONS is used by the Oracle RAC database to broadcast events that describe a change of state. GridLink Data Sources can register to receive notifications from ONS and therefore quickly become aware of any state changes in a RAC database. Using these state change notification events, GridLink Data Sources can intelligently adapt its connection pools so that it provides continuous, reliable and efficient access to the RAC database as changes happen.

An adaptive response to state changes in the RAC cluster allows WebLogic Server to handle outages by immediately retracting, closing and discarding connections to RAC instances that have been stopped or taken out by an unplanned outage, without needing to periodically poll the connections to ensure they are valid, or affecting uninvolved connections to surviving nodes. This eliminates the need to test connections to ensure applications are not given dead connections and quickly removes dead connections from RAC node failures, which in some failure modes, might otherwise hang for minutes.

Further, it allows WebLogic Server to proactively reapportion its set of connections to support scenarios where new RAC instances are added or are restarted after an outage. This results in WebLogic Server being able to make full use of the resources within the RAC database. Furthermore, using the database service model, this allows database administrators to make changes to the RAC service/instance allocations, which are then seamlessly applied through the affected WebLogic connection pools without needing to make configuration changes to the connection pool configuration. It also removes the need to create complex arrangements of multiple data sources to represent a dedicated instance of the RAC database.

The WebLogic GridLink Data Source provides Fast Connection Failover capabilities and responds to RAC database service and node events {UP, DOWN} to ensure that the reserve of physical connections in the pool are always pointing to a valid database node; and it ensures that the reserve of physical connections are well distributed across the available database nodes. The Fast Connection Failover behavior is enabled as a configuration setting on the GridLink Data Source.

With the Fast Connection Failover capability enabled, the following scenarios are supported:

- Planned down Event - Planned outages are defined as database maintenance or other activities that are needed to perform at a known point in time. Support for these events is available where an Oracle RAC service can be gracefully shutdown. In such scenarios, any borrowed or in-use connections are not interrupted and closed until work is completed and control of the connection is returned to the pool. This provides an extremely efficient way in large heterogeneous customer environments to manage planned outages.
- Unplanned down Event - Support for unplanned outages is provided by detecting and removing stale connections to an Oracle RAC cluster. Stale connections include connections that do not have a service available on any instance in an Oracle RAC cluster due to service-down and node-down events. Borrowed connections and available connections that are stale are detected, and their network connection is severed before removing them from the pool. These removed connections are not replaced by the pool. Instead, the application must retry connections before performing any work with a connection.

  The primary difference between unplanned and planned shutdown scenarios is how borrowed connections are handled. Stale connections that are idle in the pool (not borrowed) are removed in the same manner as the unplanned shutdown scenario.
- Up Event - Oracle RAC Instance Rejoin and New Instance Scenarios - Scenarios where an Oracle RAC cluster adds instances that provide a service of interest are supported. The instance may be new to the cluster or may have been restarted after a down event. In both cases, WebLogic Connection Pool for JDBC recognizes the new instance and creates connections to the node as required.

Runtime Connection Load Balancing

WebLogic GridLink Data Sources and JDBC connection pools leverage the load balancing functionality provided by an Oracle RAC database to provide better throughput and more efficient use of resources. Runtime connection load balancing requires the use of an Oracle JDBC driver and an Oracle RAC database.

Oracle performance analysis has revealed significant performance benefits from the use of runtime connection load balancing vs. a static round-robin algorithm. These benefits are observed even when nodes in the RAC cluster are balanced from a hardware perspective, and when the average load on the nodes on the cluster are expected to be reasonably uniform on average. Transient differences in load characteristics are often sufficient to make runtime connection load balancing the optimal load balancing mechanism for RAC clusters.

The load balancing advisory service issues FAN event that advice client on the current state of the cluster including advice on where to direct connections to. WebLogic Server connection pool receives load balancing advisory events issued by the database, and distributes connections to the RAC nodes accordingly as shown in the diagram below.

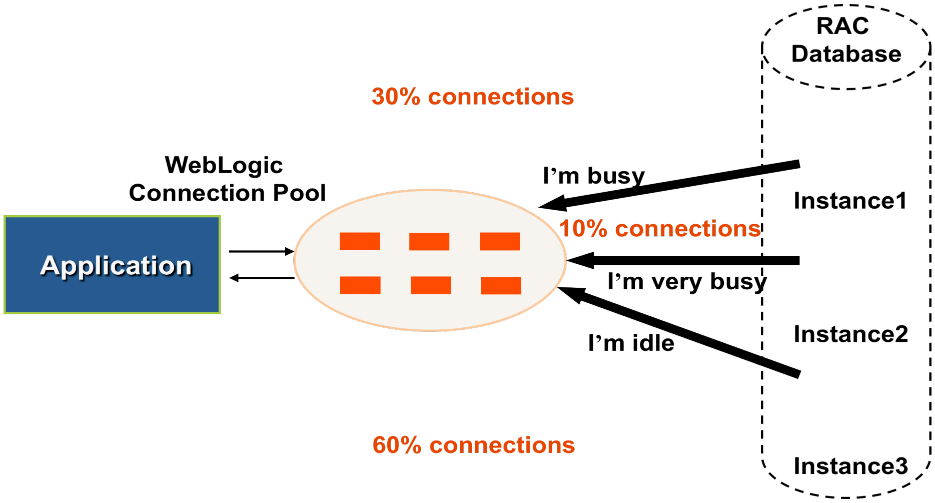


Figure 4: Runtime Connection Load Balancing

Runtime connection load balancing provides the following benefits:

- Manages pooled connections for high performance and scalability
- Receives continuous recommendations on the percentage of work to route to database instances
- Adjusts distribution of work based on different back-end node capacities such as CPU capacity or response time
- Reacts quickly to changes in cluster reconfiguration, application workload, overworked nodes, or hangs
- Receives metrics from the Oracle RAC Load Balance Advisory. Connections to well performing instances are used most often. New and unused connections to under-performing

instances will gravitate away over time. When distribution metrics are not received, connection is selected using a random choice.

Connection Affinities

WebLogic GridLink Data Sources leverage affinity functionality provided by an Oracle RAC database. Connection affinity requires the use of an Oracle JDBC driver and an Oracle RAC database version 11.1.0.6 or higher.

Connection affinity allows a connection pool to select connections that are directed at a specific Oracle RAC instance to provide the best performance for the customer applications. The pool uses run-time connection load balancing to select an Oracle RAC instance to create the first connection and then subsequent connections are created with an affinity to the same instance.

WebLogic GridLink Data Sources supports transaction-based affinity, Web Session Affinity.

Transaction Affinity

Transaction-based affinity is an affinity to an Oracle RAC instance that can be released by either the client application or a failure event. Applications typically use this type of affinity when long-lived affinity to an Oracle RAC instance is desired or when the cost (in terms of performance) of being redirected to a new Oracle RAC instance is high. WebLogic XA connections that are enlisted in a distributed transaction keep an affinity to the Oracle RAC instance for the duration of the transaction. In this case, an application would incur a significant performance cost if a connection is redirect to a different Oracle RAC instance during the distributed transaction.

The affinity will be established based on the global transaction id, instead of by individual data source, to ensure that connections obtained from different data sources that are configured for the same RAC cluster are all associated with the same RAC instance. The LLR two-phase commit optimization will be supported by the RAC data source and will also participate in XA affinity.

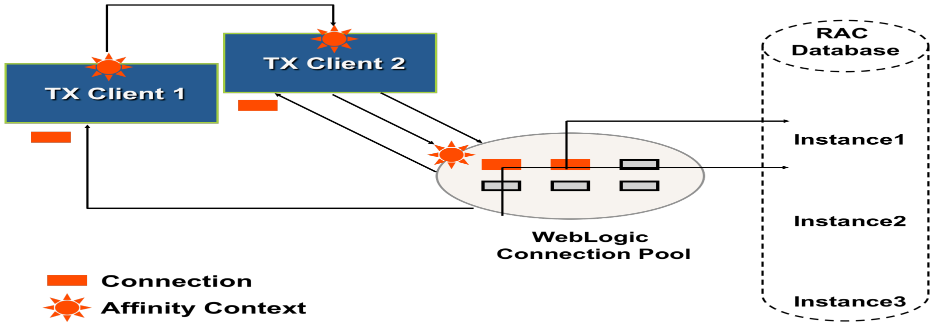


Figure 5: XA Affinity

Web Session Affinity

WebLogic Web Session Affinity is an affinity to an Oracle RAC instance that can be released by the instance, a client application, or a failure event. Applications typically use this type of affinity when short-lived affinity to an Oracle RAC instance is expected or if the cost (in terms of performance) of being redirected to a new Oracle RAC instance is minimal. For example, a mail client session might use Web session affinity to an Oracle RAC instance to increase performance and is relatively unaffected if a connection is redirected to a different instance. The typical use cases using this type of affinity are where a user session has back-to-back online transaction processing (OLTP) which could have better performance when repeated operations against the same set of records are processed by the same RAC instance. Business applications such as online shopping and online banking are typical examples of this pattern.

A GridLink data source uses the Session Affinity policy to ensure all the data base operations for a web session, including transactions, is directed to the same Oracle RAC instance of a RAC cluster. The first connection request within the same HTTP session scope uses RCLB to select a connection; the subsequent requests are enforced by the Affinity. The connection selection falls back to RCLB after the Affinity ends.

The Web Session context is stored in the HTTP session cookie and accessed via Affinity Callbacks. It's cleared by the application after session completes. And it can be propagated within the HTTP session. The Affinity Callbacks are used by the connection pool to store and retrieve the Affinity Contexts.

The Web Session Affinity scope is determined by the HTTP session lifecycle and the database failure events.

Although the Session Affinity policy for a GridLink data source is always enabled by default, a Web session is active for Session Affinity if:

- Oracle RAC is enabled, active, and the service has enabled RCLB. RCLB is enabled for a service if the service `GOAL` (NOT `CLB_GOAL`) is set to either `SERVICE_TIME` or `THROUGHPUT`.
- The database determines there is sufficient performance improvement in the cluster wait time and the Affinity flag in the payload is set to `TRUE`.

If the database determines it is not advantageous to implement session affinity, such as a high database availability condition, the database load balancing algorithm reverts to its default work allocation policy and the Affinity flag in the payload is set to `FALSE`.

WebLogic Web Session Affinity provides extreme high performance by reducing the RAC cluster wait time as shown in below.

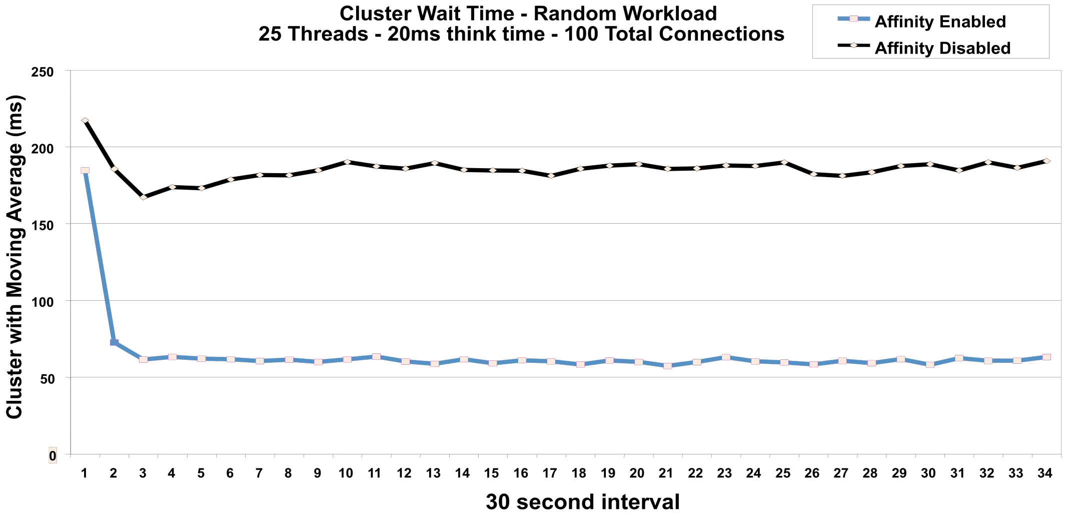
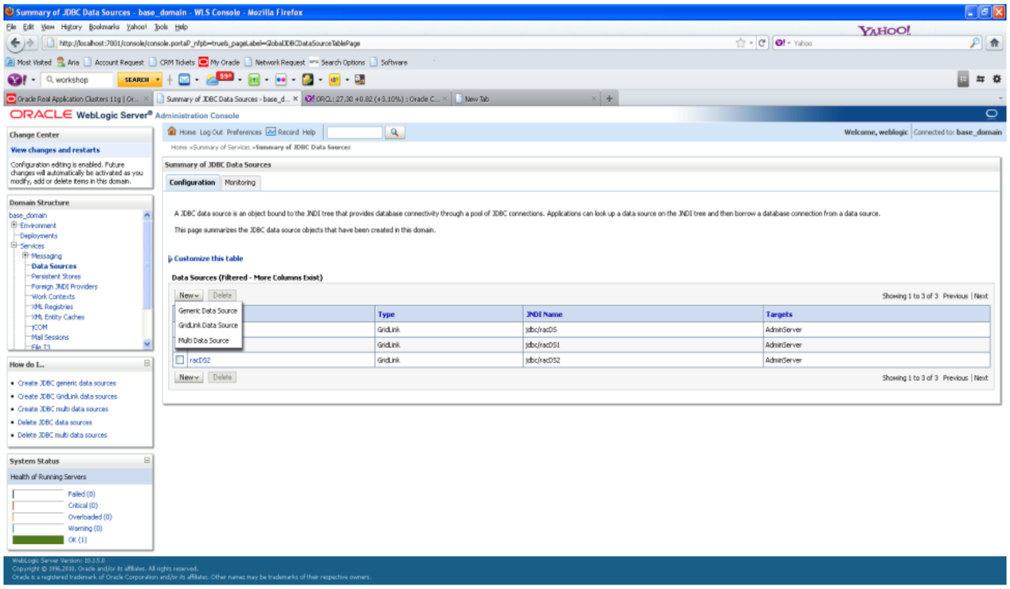


Figure 6: Web Session Affinity

WebLogic GridLink Data Source Configuration

The WebLogic Administration Console or the WebLogic Scripting Tool (WLST) can be used for creating a GridLink Data Source. GridLink Data Source is the implementation for Active GridLink functionalities.

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 doesn't leverage Oracle Notification Service. The GridLink Data Source is the new Active GridLink implementation which takes advantages of Oracle RAC supporting FCF, RCLB, Affinities and all the Database 12c new capabilities. It supports using SCAN for the configuration.


Summary of JDBC Data Sources - base_domain - WLS Console - Mozilla Firefox
ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic | Connected to: base_domain
Change Center
View changes and restarts
Domain Structure
How do I...
System Status
Configuration
Monitoring
New
Delete
Generic Data Source
GridLink Data Source
Multi Data Source
Showing 1 to 3 of 3 Previous | Next

Home >Summary of Services >Summary of JDBC Data Sources

Summary of JDBC Data Sources

A JDBC data source is an object bound to the JNDI tree that provides database connectivity through a pool of JDBC connections. Applications can look up a data source on the JNDI tree and then borrow a database connection from a data source.

This page summarizes the JDBC data source objects that have been created in this domain.

Customize this table

Data Sources (Filtered - More Columns Exist)

		Type	JNDI Name	Targets
		GridLink	jdbc/racDS	AdminServer
		GridLink	jdbc/racDS1	AdminServer
□	racDS2	GridLink	jdbc/racDS2	AdminServer

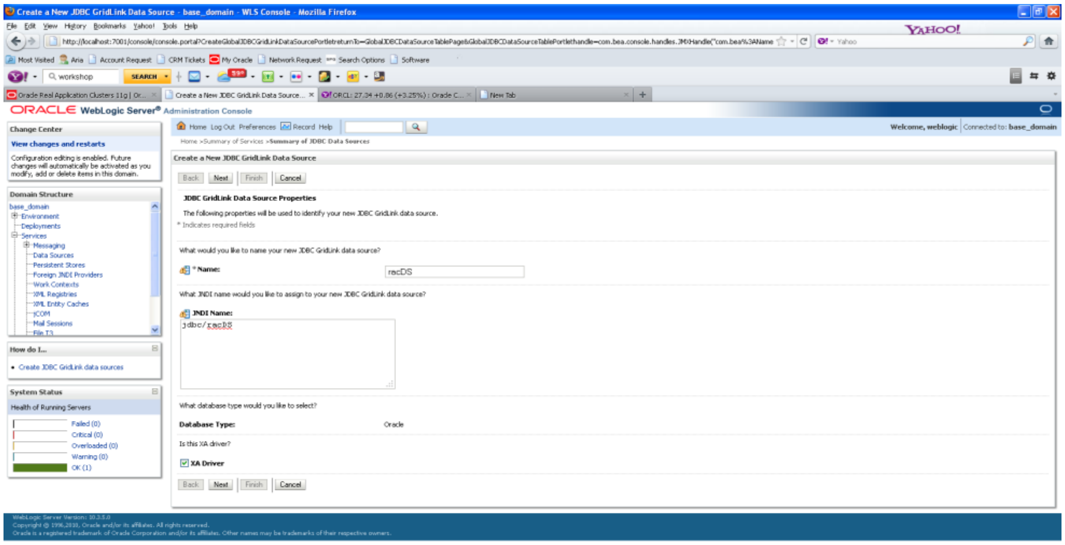
Configuration editing is enabled. Future changes will automatically be activated as you modify, add or delete items in this domain.

- Create JDBC generic data sources
- Create JDBC GridLink data sources
- Create JDBC multi data sources
- Delete JDBC data sources
- Delete JDBC multi data sources

Health of Running Servers: Failed (0) Critical (0) Overloaded (0) Warning (0) OK (1)

Figure 7: WebLogic Data Source configuration

The GridLink Data Source can be configured simply using a Data Source name and JNDI location as it's shown below.


Create a New JDBC GridLink Data Source - base_domain - WLS Console - Mozilla Firefox
ORACLE WebLogic Server® Administration Console
Home Log Out Preferences Record Help
Welcome, weblogic | Connected to: base_domain
Change Center
View changes and restarts
Domain Structure
How do I...
System Status
Back
Next
Finish
Cancel

Home >Summary of Services >Summary of JDBC Data Sources

Create a New JDBC GridLink Data Source

JDBC GridLink Data Source Properties

The following properties will be used to identify your new JDBC GridLink data source.

* Indicates required fields

What would you like to name your new JDBC GridLink data source?

* Name: racDS

What JNDI name would you like to assign to your new JDBC GridLink data source?

JNDI Name: jdbc/racDS

What database type would you like to select?

Database Type: Oracle

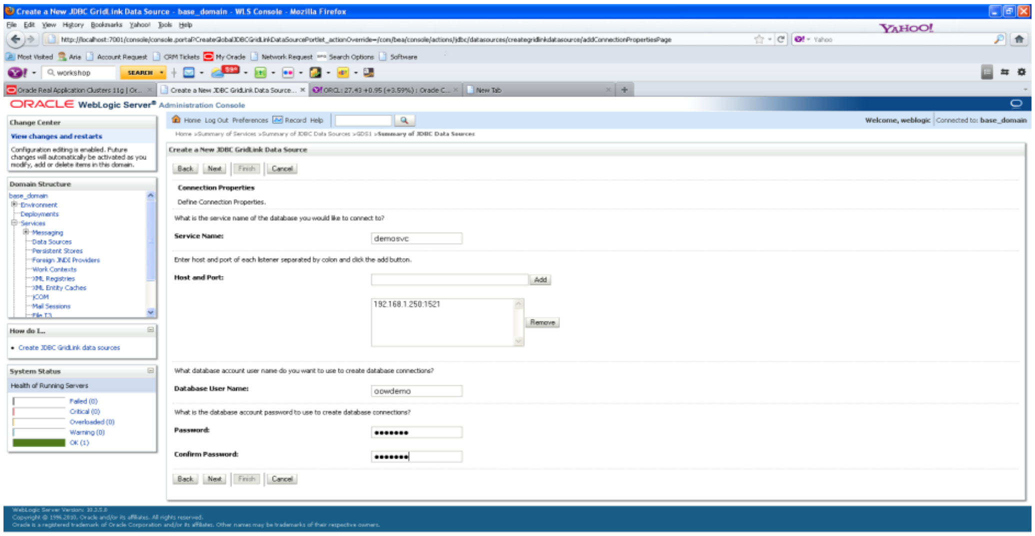
Is this XA driver?

☑ XA Driver

- Create JDBC GridLink data sources

Figure 8: GridLink Data Source configuration

To be able to leverage Oracle Notification Service, both listener and ONS will need to be configured. The SCAN address could be used for both. The Service Name is the service created in your RAC database. The Host:Port would be Your_SCAN:1521. The ONS configuration will use the same SCAN address with ONS port, such as Your_SCAN:6200.



Create a New JDBC GridLink Data Source

Connection Properties

Define Connection Properties.

What is the service name of the database you would like to connect to?

Service Name: demosvc

Enter host and port of each listener separated by colon and click the add button.

Host and Port: 192.168.1.250:1521

What database account user name do you want to use to create database connections?

Database User Name: oowdemo

What is the database account password to use to create database connections?

Password: •••••••

Confirm Password: •••••••

Figure 9: ONS configuration

All the Active GridLink performance features, such as Fast Connection Failover, Runtime Connection Load Balancing, Affinities, are enabled by one flag, i.e., 'FAN Enabled'. Different performance or HA features will kick in automatically depending upon the application scenarios. For example, FCF will kick in when a failure case happens. If the application has Global Transaction opened, the XA Transaction Affinity will kick in based on the RCLB event for the first connection. For the Web Session scenarios, Session Affinity will apply first, then RCLB. The goal is to provide the best high availability, scalability and performance solution for the heterogeneous systems and applications.

With the integration of Oracle Database 12c, features such Global Database Service, are also enabled by the flag, i.e.s 'FAN Enable'.

When the listener and ONS are configured, connections can be tested easily with the WebLogic Server Administration Console.

WebLogic Server 12c (12.1.2 and 12.1.3) and Oracle Database 12c

In most enterprise and mission critical environments, the customer applications and infrastructure require not highly available, but also fully recoverable with guaranteed transaction outcome. With

Oracle Database 12c, new solutions such as Application Continuity, Transaction Guard have arrived under the extreme demand from the industry for decades.

Oracle WebLogic Server 12c (12.1.2) and Oracle Database 12c are fully certified to work together providing high-availability, scalability and high performance. Let's look at some exciting details.

Application Continuity

It is complex for application developers to mask outages of a database session (instance, node, storage or network or any other related component) and as a result errors and timeouts are often exposed to the end users leading to user frustration, lost productivity, and lost opportunities. Application Continuity masks outages from end users and applications by recovering the in-flight work for impacted database sessions following outages. Application Continuity performs this recovery beneath the application so that the outage appears to the application as a slightly delayed execution.

Application Continuity is invoked for outages that result in recoverable errors, typically related to underlying software, foreground, hardware, communications, network, or storage layers. Application Continuity is used to improve the user experience when handling both unplanned outages and planned outages.

Introduced in Oracle Database 12c, Application Continuity strengthens the fault tolerance of systems and applications that use an Oracle database. Application Continuity is available for Oracle WebLogic Server Active GridLink, Oracle Universal Connection Pool and Oracle JDBC-Thin Driver.

Application Continuity enables replay, in a non-disruptive and rapid manner, of a database request when a recoverable error makes the database session unavailable. The request can contain transactional and non-transactional calls to the database and calls that are executed locally at the client or middle tier. After a successful replay, the application can continue where that database session left off instead of leaving users in doubt not knowing what happened to their funds transfers, flight bookings, and so on, and for administrators' avoiding the need to reboot mid-tier machines to recover from logon storms. With Application Continuity, the end user experience is improved by masking many outages, planned and unplanned, without requiring the application developer to attempt to recover the request.

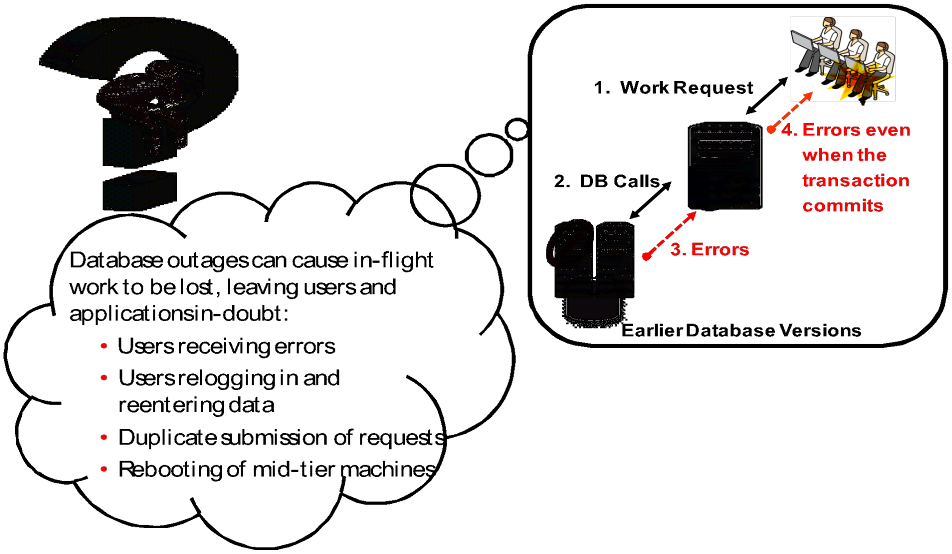
Without Application Continuity, it can be almost impossible for an application to mask outages in a safe way, for reasons that include the following:

- The state at the client remains at present time, with entered data, returned data, and variables cached while the state changes reflected in the database session are lost.
- If a commit has been issued, the commit message is not durable Furthermore, checking the status of a lost request is no guarantee that it will not commit in the future.
- Non-transactional database session state that the application needs to operate is lost.
- If the request can continue, the database and the database session must be in the right state.

Application Continuity, however, does this work for the application developer, thus masking many outages in a safe way. Application Continuity improves developer productivity by attempting to mask outages that can be safely masked.

Experience before Application Continuity

Without Application Continuity, database recovery does not mask outages that are caused by network outages, instance failures, hardware failures, repairs, configuration changes, patches, and so on.

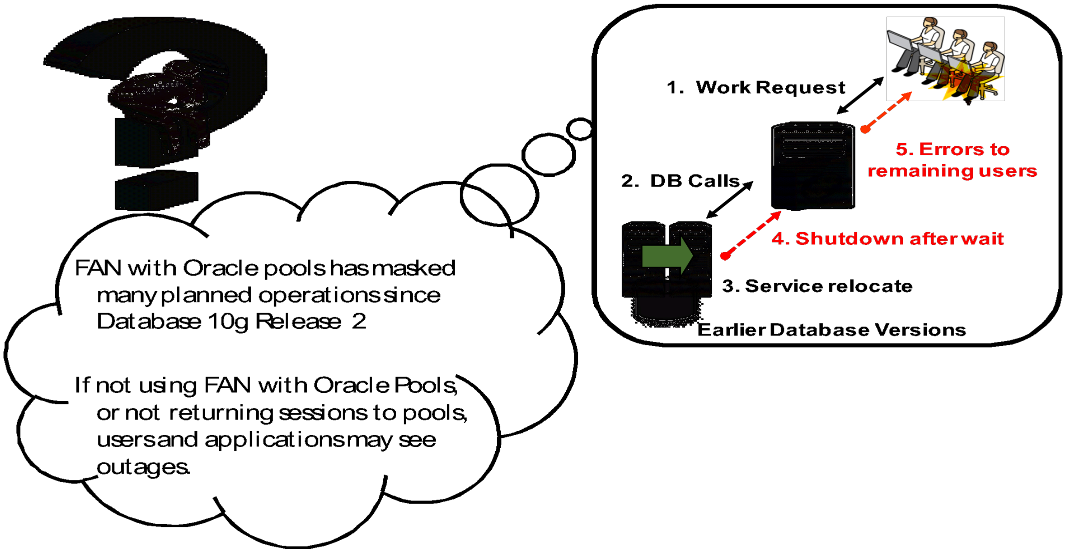

Database outages can cause in-flight work to be lost, leaving users and applications in-doubt:

- Users receiving errors
- Users relogging in and reentering data
- Duplicate submission of requests
- Rebooting of mid-tier machines

1. Work Request
2. DB Calls
3. Errors
4. Errors even when the transaction commits
Earlier Database Versions

Figure 10: Earlier experience for unplanned outages

Figure 7 illustrates the earlier experience where errors can be returned to the end user, even when the request completed.


FAN with Oracle pools has masked many planned operations since Database 10g Release 2

If not using FAN with Oracle Pools, or not returning sessions to pools, users and applications may see outages.

Figure 11: Earlier experience for planned outages

Planned outages are more frequent than unplanned outages. For applications using RAC, RAC One, Data Guard or Oracle Restart, with FAN in conjunction with Oracle connection pools – Oracle WebLogic Server Connection Pool, Oracle Universal Connection Pool, Oracle JDBC Implicit Connection Cache (ICC), ODP.NET, OCI Session Pool – planned outages have been masked since Oracle Database 10g Release 2.

Oracle WebLogic 12c (12.1.2 and 12.1.3) and Database 12c Experience with Application Continuity

Figure 9 below illustrates the improved user experience possible with Oracle Database 12c for applications using Application Continuity.

- Replays in-flight work on recoverable errors
- Masks many hardware, software, network, storage errors and outages
- Improves end user experience and developer productivity

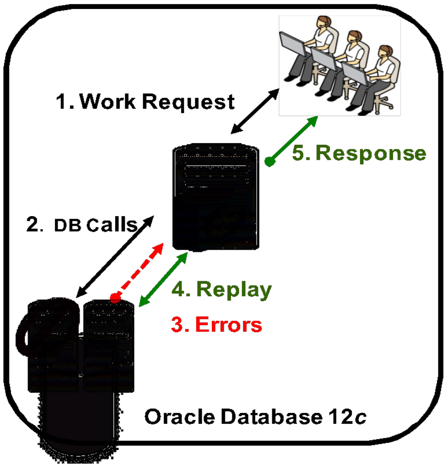


Figure 12: User experience when using Application Continuity

When replay is successful, Application Continuity masks many recoverable database outages from the applications and the users. It achieves the masking by restoring the database session, the full session (including session states, cursors, variables), and the last in-flight transaction (if there is one).

If the database session becomes unavailable due to a recoverable error, Application Continuity attempts to rebuild the session and restore any open transactions to the correct states.

If the transaction is successfully committed and does not need to be re-executed, the successful status is returned to the application.

If the replay is successful, the request continues safely, with no risk of duplication.

If the replay is not successful, the database rejects the replay and the application receives the original error. To be successful, the replay must return to the client the exact same data that the client received previously in the request, and that the application potentially made a decision on.

Application Continuity WebLogic Data Source Configuration

In WebLogic Data Source implementation, the Datasource schema, weblogic-jdbc.xsd, has been enhanced to provide configuration with the following new element.

```
<element name="replay-initiation-timeout"
        type=" j2ee:xsdNonNegativeIntegerType" nillable="true" minOccurs="0">
```

Transaction Guard

Transaction Guard with Oracle Database 12c is a reliable protocol and tool that returns the outcome of the last in-flight transaction after an outage that makes the database session unavailable. Without Transaction Guard, applications and users who attempt to retry operations following an outage can cause logical corruption by committing duplicate transactions or committing transactions out of order.

Transaction Guard avoids the costs of ambiguous errors that lead to user frustration, customer support calls, and lost opportunities. Transaction Guard is safer and performs better, with lower overheads, than home grown solutions for known outcome and at-most-once execution.

Using Transaction Guard, the end user experience is vastly improved by returning to the application and user, following an outage, whether the last submission committed and completed or did not. After submitting a request, it is far better to know whether your funds transfer, bill payment, form submission and so on has executed or, that it was not done and is safe to re-submit. Without using Transaction Guard, users can receive a vague, ambiguous error message following an outage, and are left not knowing what happened to the last in-flight operation. Applications typically display messages that can be frustrating such as the following:

- Please call customer support
- Do not press resubmit or reload
- Do not use the backspace key

Developers embed the Transaction Guard APIs in their application or mid-tier error handling. Following the flow in figure one, the client submits work to the application (step one), that in turn submits calls to the Oracle database (step 2). When a recoverable error occurs on a session (step 3), the error handling invokes Transaction Guard to return the outcome of the last in-flight work on that session (step 4). Transaction Guard's reliable protocol enforces that when the transaction outcome is returned to the application, the outcome persists with the value that is returned. Using Transaction Guard, once a committed or uncommitted result is returned to the application, the result stays this way (step five). This is critically important. A committed result stays committed. An uncommitted result stays uncommitted, and is a green light, for example, to safely resubmit. It can also be a green light for applications to resubmit themselves in case of an outage.

The benefits of using Transaction Guard are:

- For businesses, a much better user experience with fewer support calls and missed opportunities.
- For users, a reliable outcome for the last work submitted following outages.
- For developers, increased productivity handling outages.
- Overall, increased performance and higher safety over home built solutions handling idempotence and resubmission.

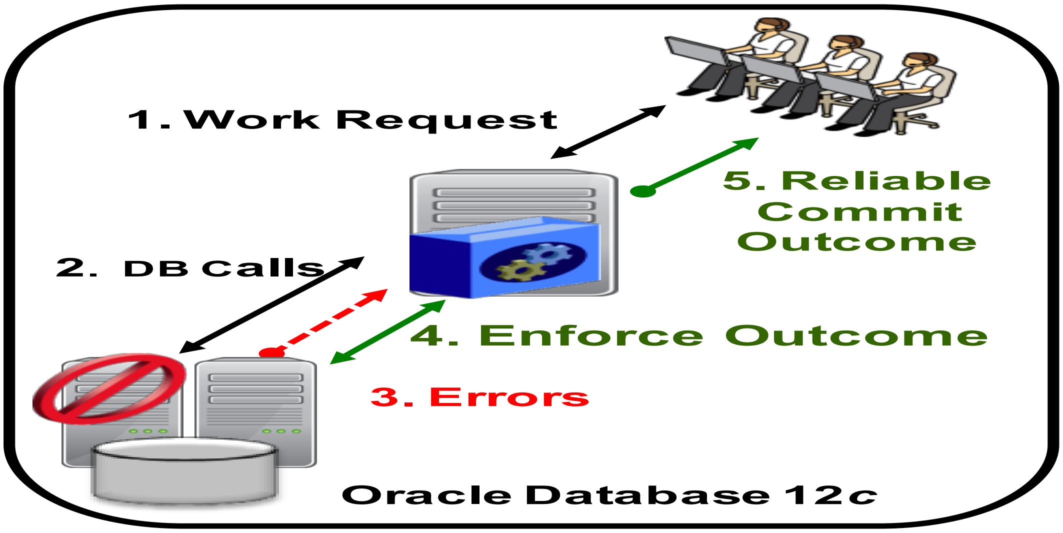


Figure 13: Transaction Guard Provides a Reliable Commit Outcome

Database Resident Connection Pool (DRCP)

In middle-tier connection pools, such as WebLogic Sever JDBC connection pool, every connection cache maintains a minimum number of connections to the server. Each connection represents used up resources at the server. All these open connections are not utilized at any given time, which means that there are unused resources that unnecessarily take up server resources. In a multiple middle-tier scenario, such as WebLogic cluster, these connections are not shared with different instances and are retained in the cache even if some of these are idle. However, a large number of WebLogic connection pools increase the number of inactive connections to the Database server significantly and waste a lot of Database resources because all the connections do not remain active simultaneously.

For example, in a middle-tier connection pool, such as WebLogic JDBC connection pool, if the minimum pool size is 200, then the connection pool has 200 connections to the server, and the Database server has 200 server processes associated with these connections. If there are 30 WebLogic instances with a connection pool of minimum size 200, then the server has 6000 (200 * 30) corresponding server processes running. Typically, on an average only 5% of the connections, and in turn, server processes are in use at any given time. So, out of the 6,000 server processes, only 300 server processes are active at any given time. This leads to over 5,700 unused server processes on the server. These unused processes are the wasted resources on the server.

The Database Resident Connection Pool implementation creates a pool on the server side, which is shared across multiple client pools, such as WebLogic JDBC connection pools. This significantly lowers memory consumption on the server because of reduced number of server processes on the server and increases the scalability of the Database server.

The Database Resident Connection Pool needs to be enabled in both client and server sides.

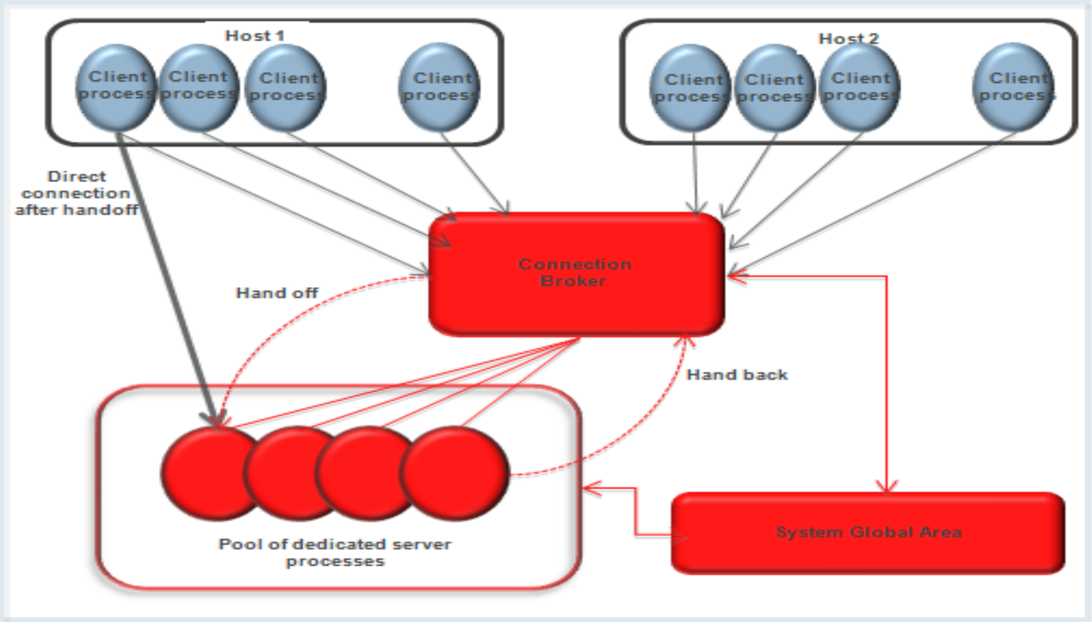


Figure 14: Database Resident Connection Pool

With integration of WebLogic Server, Database Resident Connection Pool provides following benefits:

- Pooled dedicated servers shared across client systems and processes
- Low connect/disconnect costs
    - Server "locked" on connect
    - Server "released" on disconnect

- Low-latency performance of dedicated servers
- Extreme scalability with a DRCP-capable client driver

Pluggable Database

In Oracle Database 12c, Database consolidation provides the ability to host multiple applications or databases on the same system platform or within the same database.

Pluggable Database is the capability that enables an Oracle database to contain a portable set of schemas, objects, and related structures that appears logically to an application as a separate database. This self-contained collection is called a pluggable database (PDB). A container database (CDB) contains PDBs.

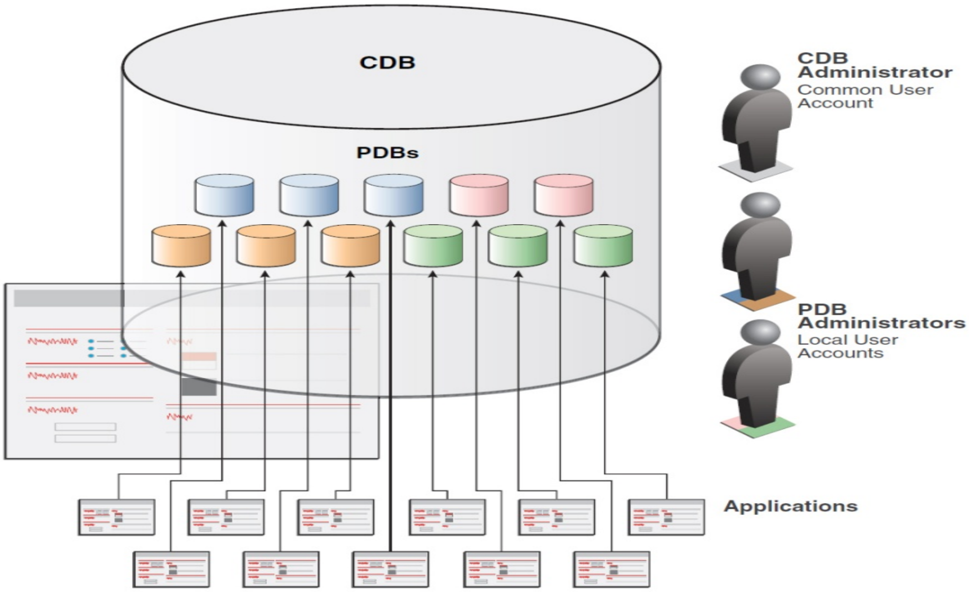


Figure 15: Pluggable Database

Pluggable database is the most cost-effective form of database consolidation. By consolidating multiple physical databases on separate computers into a single database on an optimized engineered platform such as Exadata Database Machine, you gain the following benefits:

- Cost reduction for hardware
- Portability of an application's database back end
- Ease of database and system administration
- Centralized management of database accounts and privileges

- Easier and faster upgrade paths

Global Database Services

Figure 16: Global Database Services

Global Data Services enables administrators to automatically and transparently manage client workloads across replicated databases that offer common services. A database service is a named representation of one or more database instances. Services enable you to group database workloads and route a particular work request to an appropriate instance. A global service is a service provided by multiple databases synchronized through data replication.

- Enables database services to be deployed anywhere within a globally distributed configuration
- Transparently supports existing Fast Connection Failover (FCF), Runtime Load Balancing (RLB), Database Affinity
- Databases may include Oracle RAC, single-instance Oracle databases interconnected through Data Guard, GoldenGate, or any other replication technology

It's easy and straightforward to enable Global Database Service with WebLogic Datasource Configuration.

- Enable Fast Connection Failover (FCF)
- Automatic ONS configuration – no need to call setONSConfiguration!
- Specify global service name and region in connect URL

```
(DESCRIPTION= (ADDRESS_LIST= (LOAD_BALANCE=ON) (FAILOVER=ON)
(ADDRESS=(GDS_protocol_address_information))
(ADDRESS=(GDS_protocol_address_information)))

(CONNECT_DATA= (SERVICE_NAME=global_service_name)
(REGION=region_name)))
```

Summary

Oracle WebLogic Server 12c (12.1.2 and 12.1.3) and Oracle Database 12c are fully certified to work together providing high-availability, scalability and high performance. As the only certified solution in the industry, WebLogic Active GridLink meets the customer system highly available requirements for planned downtime and meets the end user always ready to serve demands for unplanned downtime with Fast Connection Failover implementation; and it brings in unbeatable high performance with Runtime Connection Load Balancing and Database Affinities for customer enterprise and mission critical applications and services.

With integrating with all the new capabilities with Oracle Database 12c, WebLogic Active GridLink provides solutions for auto-recovery with Application Continuity, guaranteed transaction outcome with Transaction Guard and extreme performance with Database Resident Connection Pool. It enables database services to be deployed anywhere within a globally distributed configuration with Global Database Services, and it leverages the most cost-effective form of database consolidation with Pluggable Database services.

Reference

http://www.oracle.com/technetwork/database/database-cloud/private/application-continuity-wp-12c-1966213.pdf

http://www.oracle.com/technetwork/database/database-cloud/private/transaction-guard-wp-12c-1966209.pdf

http://www.oracle.com/technetwork/database/options/clustering/overview/maxappavailabilitywithdatabase12c-2008418.pdf

http://www.oracle.com/technetwork/database/options/clustering/appconndemo-1967318.zip

White Paper Title
March 2013
Author: Frances Zhao-Perez, Carol Colrain
Contributing Authors: David Baum

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