An Oracle and NEC Joint White Paper
June 2014

Active GridLink for RAC: Intelligent Integration Between WebLogic Server and Oracle Real Application Clusters

Contents

1. Introduction

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.

Oracle WebLogic Server 11g and 12c provides strong support for the Real Application Clusters (RAC) features in Oracle Database 11g and the coming 12c, minimizing database access time while allowing transparent access to rich pooling management functions that maximizes both connection performance and availability.

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.

Oracle and NEC have been jointly invested on verifying and testing the Active GridLink for RAC solution implemented within Oracle WebLogic Server which gives the rich functionality for Oracle Database Real Application Clusters integration. It's not only the best High Availability solution in the market, but also Web Logic Server is the only Application Server who has been fully integrated and certified with Oracle

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

Database RAC 11g 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 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.

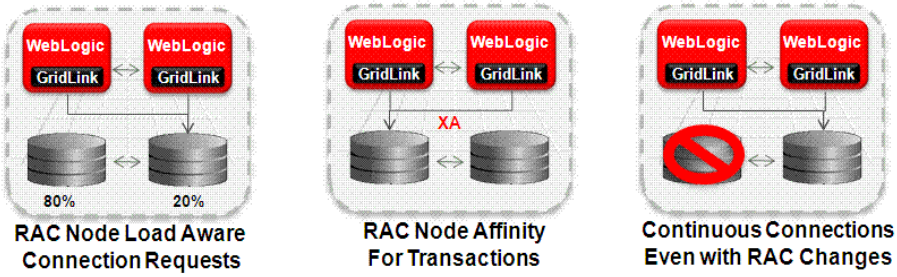


Figure 1

In this article, we start with a brief introduction to Oracle RAC and an overview of the Oracle RAC features supported in Oracle WebLogic Server 11g and 12c. We then focus on details of the effort that have been jointly done with Oracle and NEC with all the detailed testing scenarios and testing results, along with the analysis. The background and overview of NEC's test of Active GridLink for RAC will be covered in details. The technical details about Runtime Connection Load Balancing, 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.

In WebLogic Server 12c (12.1.2), Active GridLink includes Oracle Database 12c integration, including the latest functionalities of Application Continuity, Transaction Guard, Database Resident Connection Pool, Pluggable Database and Global Data Services. All contents covered in this paper also apply to Oracle WebLogic Server 12.1.3. In next version of this white paper, we will be focusing on all the testing scenarios of using Oracle Database 12c.

2. 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.

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.

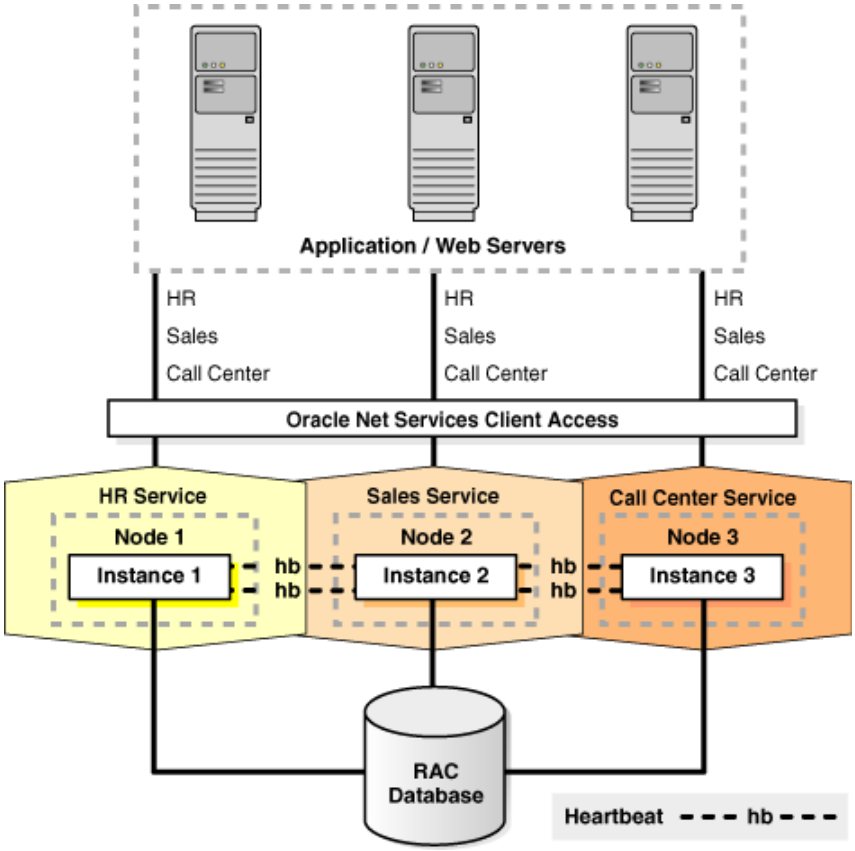


Figure 2

New Support in Oracle RAC 11gR2

The new features in Oracle RAC 11gR2 include:

- Oracle RAC One Node, it provides enhanced high availability for single-instance databases, protecting them from both planned and unplanned downtime.

- Single Client Access Name (SCAN), Oracle Database 11g database clients use SCAN to connect to the database with the capability of resolving to multiple IP addresses, reflecting multiple listeners in the cluster handling public client connections.
- Edition-Based Redefinition, an edition attribute can be specified for a database service using SRVCTL, then all subsequent connections that specify the service use this edition as the initial session edition.
- Enhanced Oracle RAC monitoring and diagnostics in Enterprise Manager.
- Enhanced Oracle RAC Configuration Assistants.
- Enhancements to SRVCTL for Grid Infrastructure Management.
- OCI runtime connection load balancing; which supports for parallel execution processes running on database instances and support for distributed transactions spanning; and the Oracle Database Quality of Service Management Server.

Oracle WebLogic Server 11g, 12c and Oracle 11g RAC are fully certified to work together providing high-availability, scalability and performance. The Oracle RAC services, such as failover, runtime connection load-balancing, Affinity etc, are available via Oracle WebLogic Server JDBC data source and connection pool implementations.

New Features in Oracle Database 12c

Oracle Database 12c is one of most successful releases in Oracle's history, with a comprehensive set of innovative features to make the world's leading database a key component for Oracle private and public clouds. Oracle WebLogic Server 12c is tightly integrated with Oracle Database 12c to allow customers to take advantage of the following capabilities.

- *Application Continuity* – 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* – A reliability tool that returns the outcome of the last in-flight transaction that made a database session unavailable after an outage. Without Transaction Guard, applications that 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* – Creates a connection pool in the server that can be shared across many clients. This architecture significantly lowers memory consumption on the server by reducing the number of server processes and increasing the overall scalability of the database server.
- *Oracle Database Consolidation and Pluggable Database* – Lets customers host multiple applications or databases on the same system platform or within the same database. This unique capability

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* – 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 Active Data Guard* – Provides new PL/SQL packages that automate the process of performing a rolling upgrade using a physical standby database. This feature is useful when you upgrade to a newer version of Oracle Database or to a new patch set or when performing other database maintenance. You simply input an upgrade plan and PL/SQL packages automate the upgrade in a three-phased plan: start, switchover, and finish.

Oracle WebLogic Server 12c and Oracle Database 12c are fully certified to work together to deliver high-availability, scalability and performance. All Oracle Database solutions, including Application Continuity, Transaction Guard, and Database Resident Connection Pool, are available via Oracle WebLogic Server JDBC data source and connection pool implementations.

3. Oracle WebLogic Server and 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), called Oracle WebLogic Active GridLink for RAC which is the market-leading mid-tier integration solution leveraging the latest and greatest Oracle RAC advances.

In WebLogic Server 12c (12.1.2, as well as 12.1.3), Active GridLink includes Oracle Database 12c integration, including the latest functionalities of Application Continuity, Transaction Guard, Database Resident Connection Pool, Pluggable Database and Global Data Services.

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 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 (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.

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 3-1

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 and Data 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.

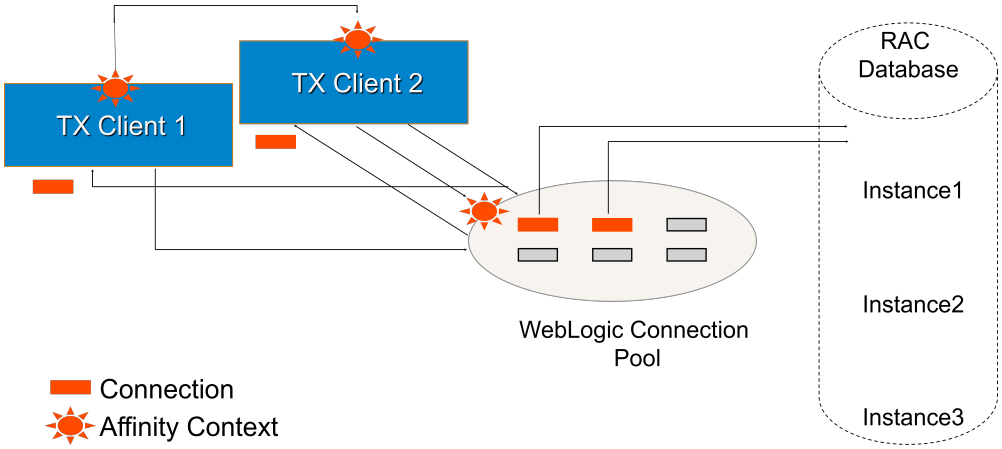


Figure 3-2

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.

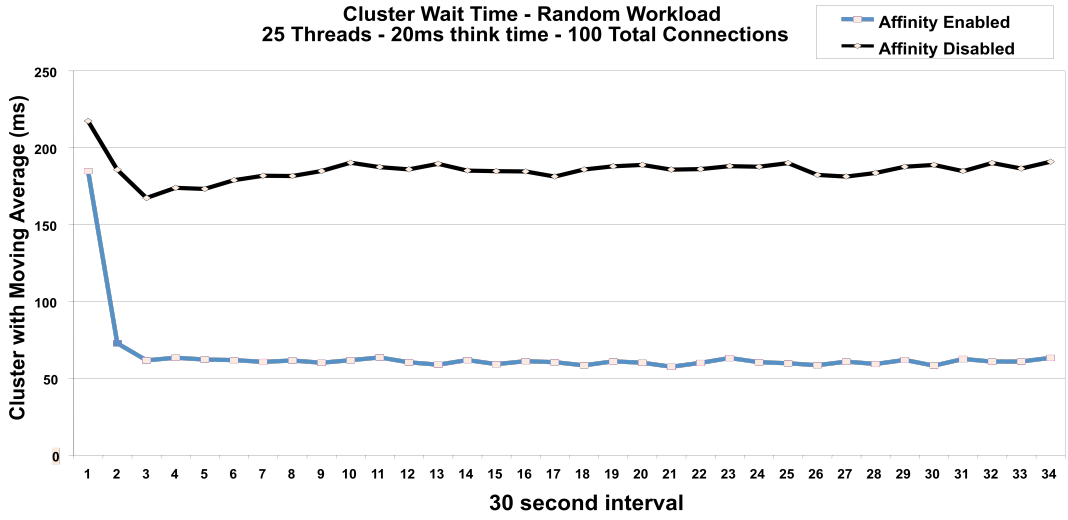


Figure 5

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.

4. NEC Testing Scenario Overview

Purpose

NEC has constructed a lot of high availability systems with WebLogic Server and Oracle Real Application Cluster, and has been storing the know-how based on experiences. NEC investigated advantages and features of the GridLink data source as a new function from standpoint of the system integrator.

NEC focused on high availability and flexibility that GridLink data source achieves by cooperation with RAC. The purpose of this test is to get high serviceability by diverting system integration know-how of JDBC data source already storing, in addition to using easy these functions.

Major items

We tested following functions which GridLink data source has and verified those behavior and effect. We'll describe details of these tests after Chapter 5.

1. Runtime Connection Load Balancing(RCLB)
2. Web Session Affinity
3. Fast Connection Failover(FCF)
4. Performance Improvement by Dynamic Changes in RAC Topology

Results Summary

As results of this test, the following were found.

1. WLS chooses which RAC node to send a request to with notice of load balancing advisory from RAC. Therefore, server resources are used efficiently. As a result, the response time to the busy RAC node is improved, and the average response time in the system is also improved.
2. Cache hit ratio is improved by Web Session Affinity, so response time are improved owing to interconnect traffic decrease. RCLB load-balancing advisory automatically switches over enabling and disabling of Affinity according to the imbalance of load of RAC node and the cluster waiting time.
3. Using FCF, application notices error by notification sent from survivor RAC nodes when public network down or interconnect down occurs. By this notification, the failure which could be detected by relying on waiting for TCP/IP time-outs on client side can be canceled and retried rapidly to continue processing.
4. By combination of runtime connection load balancing and RAC topology change notification of FCF, it is possible to add RAC node to running system and improve the performance.

Using SCAN for a JDBC connection, a RAC node could be added dynamically to a running system without changing WLS data source configuration.

Environment

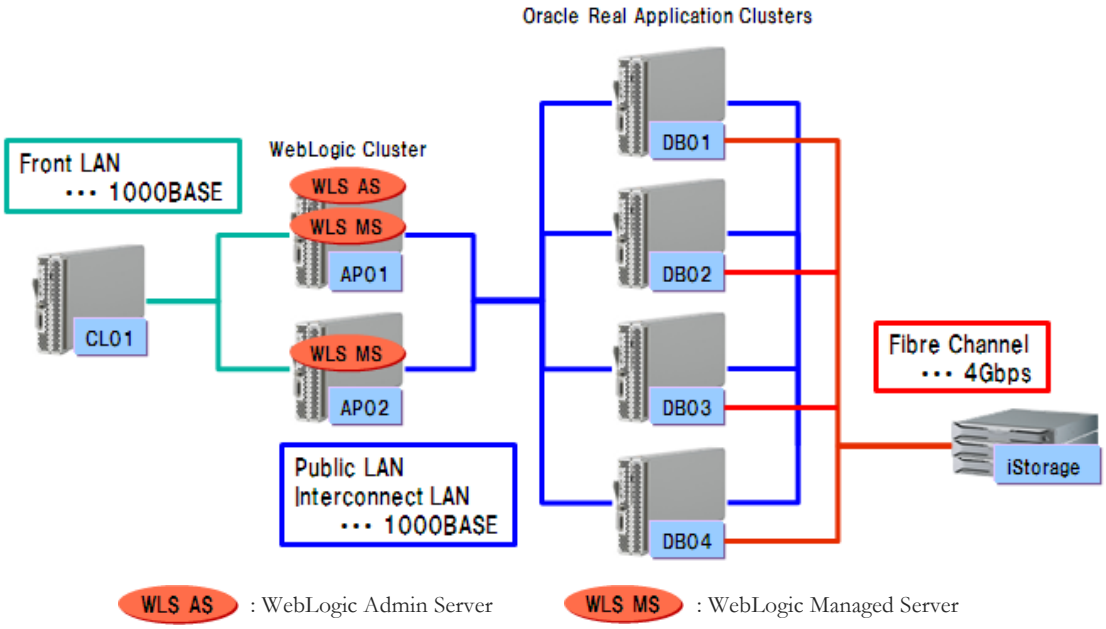


Figure 4-1 System environment in this test

Hardware Specifications

System architecture in this test is below.

Server Role	Host Name
Client	CL01
Application Server(WLS 12.1.1)	AP01, AP02
Database Server (RAC 11.2.0.3)	DB01, DB02, DB03, DB04

All servers in our environment are same type of machine. It is Express5800 which is IA server produced by NEC. The database is deployed on iStorage which is disk storage produced by NEC.

The iStorage is connected to database servers by 4 Fiber Channel (FC) with 4Gbps bandwidth. Oracle WebLogic Server 12c is installed in each application servers, and Oracle Database 11g Release2 is installed in each database server. Details of those hardware specifications are below.

Express5800/B120a-d

CPU	Intel®Xeon® processor X5550(2.66GHz) [core :4] * 2
Memory	48GB
OS	Red Hat Enterprise Linux 5 Update7 x86-64

Storage: iStorage D3-10(FC): 1 node

Hard Disk Drive	SAS 300GB(15000rpm) * 36 (Sum of the main unit and disk enclosure * 2)
Cache Memory	4GB
Host interface	Fiber Channel 4Gbps * 4

The versions of product installed to each node are below.

Application Server: 2 nodes

Oracle WebLogic Server	Oracle WebLogic Server 12.1.1

Database Server: 4 nodes

Oracle Database	Oracle Database 11g Enterprise Edition Release 11.2.0.3

About JDBC connection method

JDBC connection method is constructed by SCAN in this test. All WLS node can get a connection to RAC by using same JDBC connection descriptor. Because of SCAN, It is not necessary to change configuration on WLS side when database topology changed.

5. Runtime Connection Load Balancing (RCLB)

This chapter describes detail of behavior and performance improvement according to test results about Runtime Connection Load Balancing as one of GridLink data source functions.

Setting up RCLB Goal

RAC sends a load balancing advisory to WLS through ONS in an interval of once per 30 seconds. WLS refers to the percentage included in this advisory and follows this percentage of which RAC node to use.

RAC calculates load balancing advisory using service statistics and RCLB goal (RLB_GOAL). RCLB goal and service statistics attribute to calculate percentages are as follows.

In case RCLB goal is SERVICE_TIME:

RAC calculates load balancing advisory based on value of "elapsed time per call". WLS often sends requests to RAC node which has small value of elapsed time per call.

In case RCLB goal is THROUGHPUT:

RAC calculates load balancing advisory based on value of "number of user calls per second". WLS often sends requests to RAC node which has large value of number of user calls per second.

In case RCLB goal is NONE:

RAC doesn't send load balancing advisory. NONE is default value when making RAC service but NONE should be not used usually when using GridLink data source. We recommend changing this parameter to SERVICE_TIME or THROUGHPUT from NONE if using GridLink data source.

Load Balancing Advisory

Distribution of used connections by RCLB

Load balancing advisory has information of percentage which WLS should send requests to which RAC nodes. WLS receives a load balancing advisory and selects a connection to RAC node with its percentage. CurrentWeight in WLS Runtime MBean shows this percentage.

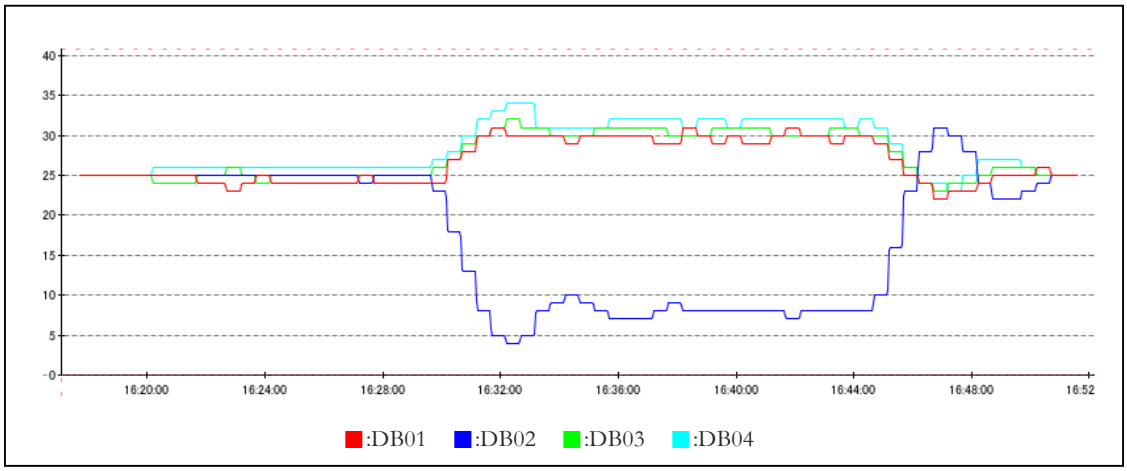


Figure 5-1 CurrentWeight to each RAC node

For example, Figure 5-1 shows CurrentWeight of 4 RAC nodes. In the beginning of the test, Workloads to DB01-DB04 are applied from only WLS. Several minutes after the start of load from WLS, Other process applies additional workload to DB02 while WLS keeps applying load. Then load of DB02 becomes high, the response time of DB02 becomes higher than DB01, DB03, and DB04. So CurrentWeight of DB02 decreases.

Because CurrentWeight of DB02 decreases, WLS doesn't often use connections to DB02. For example, at 16:36:00 in Figure 5-1, Percentage of using connection to DB01 is 30%, DB02 is 7%, and DB03 is 31%, DB04 is 32%.

Performance Improvement

Test Case

When a load of one node becomes high by other process, RCLB balances load of each node and performance of the system improves. We tested 3 patterns as test cases, they are in case of SERVICE_TIME, in case of THROUGHPUT and in case of NONE (RCLB isn't used.)

The testing procedure is as follows:

Step1. WLS applies workload to each RAC node and makes CPU usage of each RAC node become about 50%. Because every RAC node has same workload, CurrentWeight also are equality, WLS sends requests equally to each node.

Step2. Other process applies an additional workload that is 75% of CPU usage to DB02 while WLS are applying workload of Step1. Because DB02 has high load, If RCLB goal is SERVICE_TIME or THROUGHPUT, CurrentWeight of DB02 become lower than other database nodes, and the percentage for which WLS uses DB02 decrease. The percentage for which WLS uses DB01, DB03 and DB04 increases by the values of decrement of the percentage for which WLS uses DB02.

Step3. Only a workload of Step2 is finished. The performance of DB02 returns to degree of Step1. WLS keeps applying a load of Step1, the workload applied to each RAC node becomes about the same.

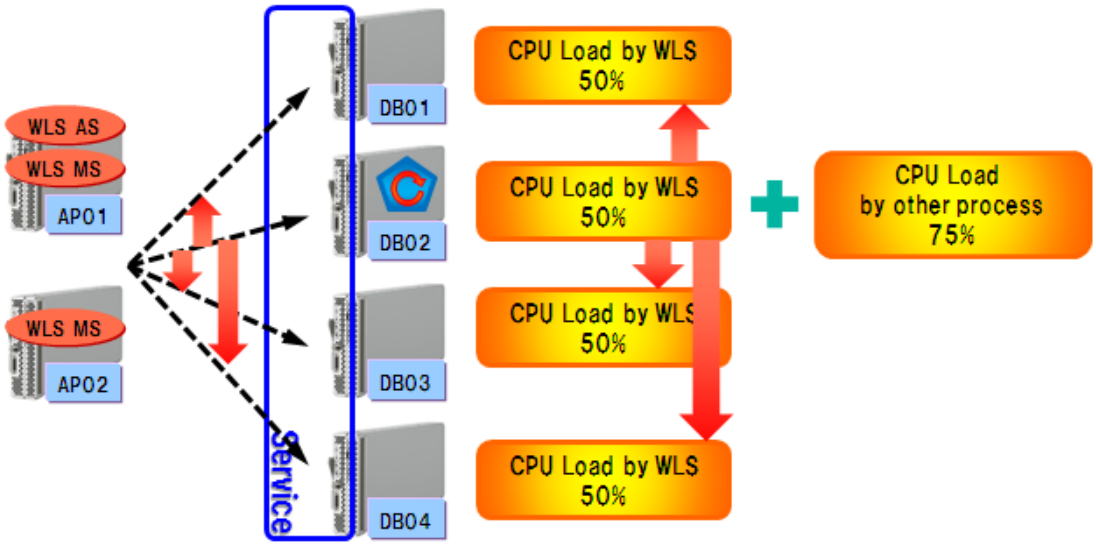


Figure 5-2 Test environment of runtime connection load balancing

Results

The percentage which RAC node is used by WLS (CurrentWeight)

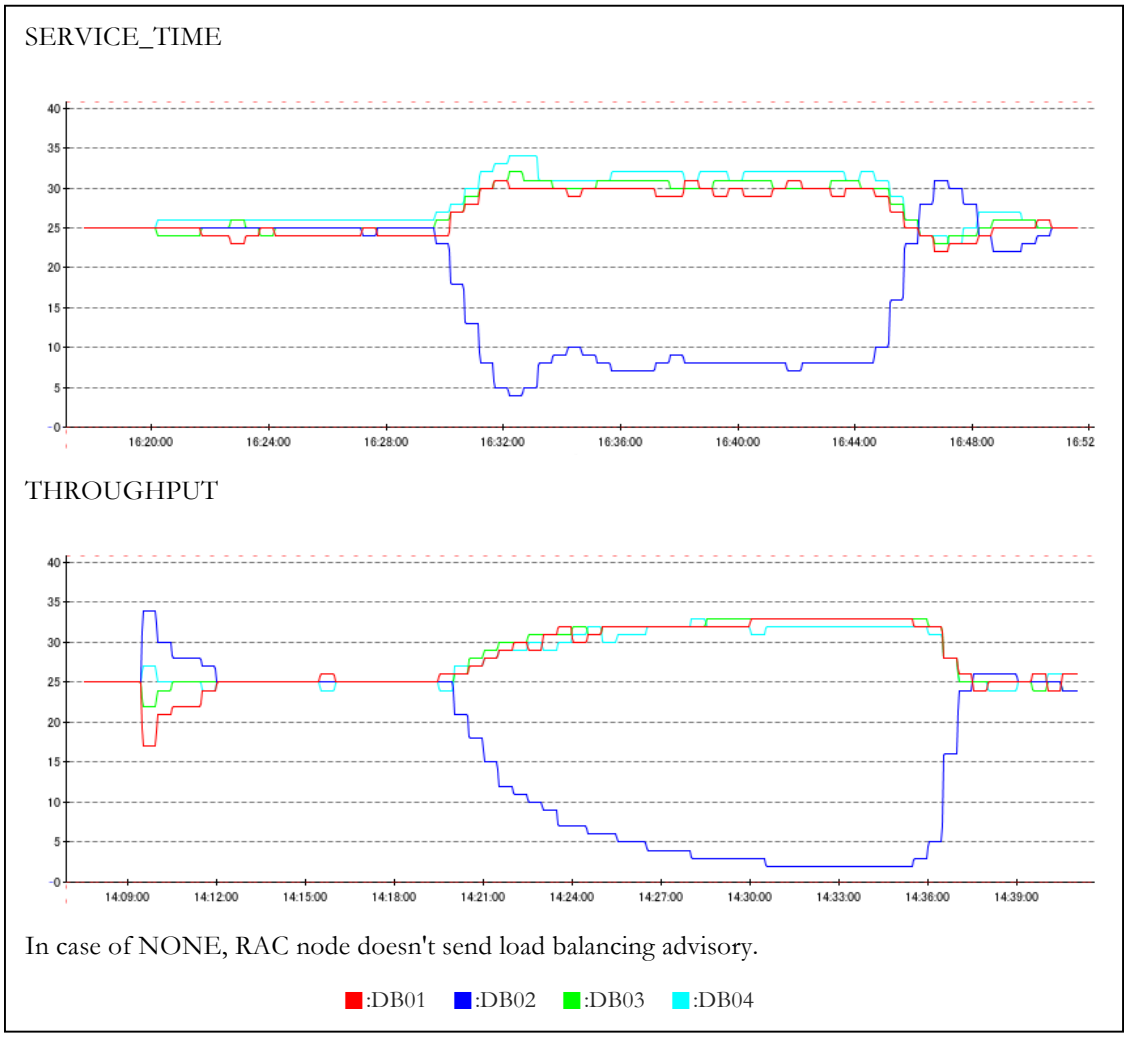


Figure 5-3 The percentage which RAC node is used by WLS

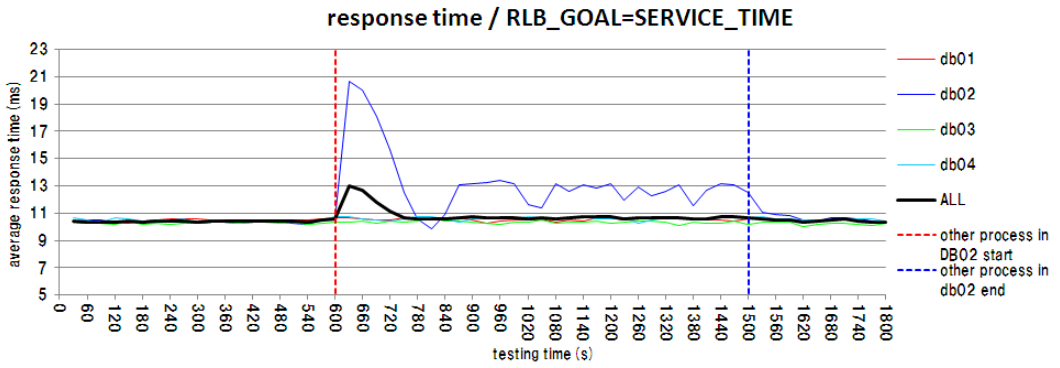
Figure 5-3 shows the percentage which RAC node is used by WLS, This percentage is equal to CurrentWeight value in WLS MBean. CurrentWeight of DB02 is lower than that of other nodes in both case of SERVICE_TIME and case of THROUGHPUT.

After a load of step2 is finished, CurrentWeight of each node returns equally.

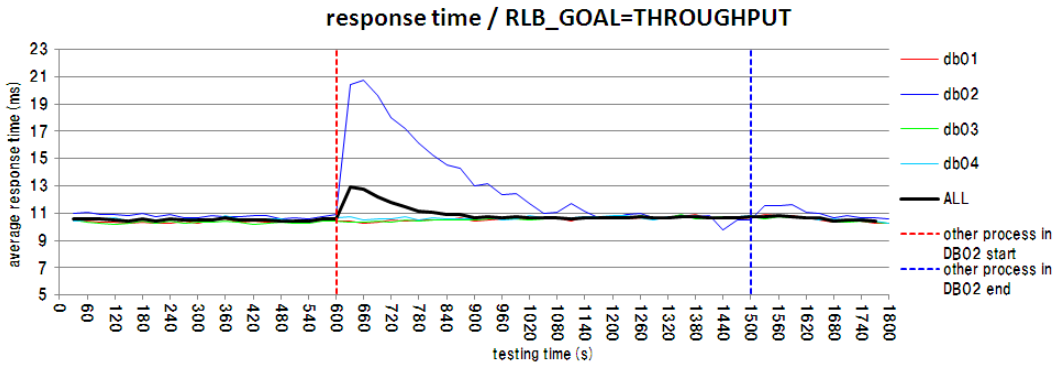
Response Time

("ALL" is average of response time in all RAC nodes)

SERVICE_TIME



THROUGHPUT



NONE

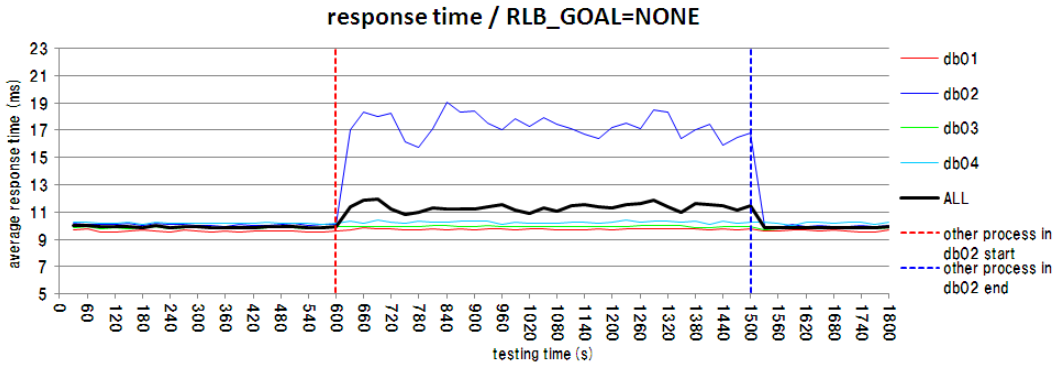


Figure 5-4 Response time by RCLB goal

Figure 5-4 shows how the response time changes by RCLB goal. In case of SERVICE_TIME or THROUGHPUT, Just after other processes applied a load to DB02 at Step2, the response time becomes high, but a system can perform the response time as much as Step1 after several minutes.

Details of results in each RCLB goal are below.

In case of SERVICE_TIME:

When a load of DB02 became heavy by Step2, runtime load balancing begins to function, and the percentage of requests sent to DB02 from WLS decreases. The response time is improved because workload of DB02 decreases and was distributed to the other nodes. The average response time of the system is nearly the same as that in Step1. Therefore, it can be said that a resource can be used efficiently.

In case of THROUGHPUT:

When a load of DB02 became heavy by Step2, runtime load balancing begins to function, and the percentage of requests sent to DB02 from WLS decreases like the case of SERVICE_TIME. It takes longer time to improve response time than case of SERVICE_TIME, but the response time of DB02 became as short as that of the other nodes, the response time of all nodes was about the same. The average response time of the system is nearly the same as that in Step1. Therefore, it can be said that a resource can be used efficiently.

In case of NONE:

The percentage of request from WLS to DB02 has no change because runtime connection load balancing isn't used. High load of DB02 wasn't improved; response time of DB02 in Step2 was 170% of that of the other nodes. The average response time of the system in Step2 was 120% of Step1.

RCLB effectiveness and comparison of response time

Table 1 shows average of the response time of period of time that runtime connection load balancing became stable in Step2.

It is the case of SERVICE_TIME that the average of response time of 4 nodes RAC improves most. It is the case of THROUGHPUT that the response time of DB02 with an additional load most improved. The response time of DB02 becomes as low as other nodes.

Because the response time in case of THROUGHPUT and the response time in case of SERVICE_TIME are also improved than a case of NONE, It can be said that the response time of high load RAC node and the whole system were improved by RCLB.

Table 1 Average of response time (milliseconds)

RLB_GOAL \ Node	Average of 4 Nodes	DB02
SERVICE_TIME	10.65	12.61
THROUGHPUT	10.70	10.65
NONE	11.47	17.12

Issue in case of SERVICE_TIME

If WLS keep applying a remarkable high load to each RAC node in case of SERVICE_TIME, the issue that makes CurrentWeight unstable and makes the response time large may occur.

Our test case that this issue occurred is below.

- WLS applied a high load so that CPU utilization of each RAC node becomes over 80%.
  - This issue was found in the test case of adding RAC node to improve performance. We expected that a response time improves by adding a new RAC node to the high load RAC cluster. However, this issue occurred before new node is added.
- If CPU load is low, this issue doesn't occur. In case of a load of Figure 5-2 which makes CPU utilization percentage 50% or less, this issue didn't occur.
- The SQL query in application we used for test repeats a search to a big table. That's very simple. There are little Disk I/O by this process. Therefore, CPU time has the majority of response time in each RAC node.
- In case of short SQL, it takes longer time that CurrentWeight becomes imbalance than case of long SQL. However, CurrentWeight becomes imbalance finally. Therefore the response time in the system eventually becomes long.

NEC and Oracle are now investigating this issue. Please consider using THROUGHPUT as workaround of this issue at present.

6. Web Session Affinity

This chapter describes detail of behavior and performance improvement according to test results about Web Session Affinity as one of GridLink data source functions. WebLogic GridLink Data Sources supports transaction-based affinity, Web Session Affinity. However, this chapter describes Web Session Affinity by test result. This is because configuration of Affinity is Web Session Affinity in default and transaction-based affinity is similar to web-session affinity in that they select RAC node with some contexts.

Main Behavior

GridLink data source has two algorithms for getting connection from a connection pool. That's RCLB and Affinity. Both are enabled by default, and work as follows.

1. If HTTP request connects to RAC node at first with a HTTP session ID, that request borrows a connection from pool by RCLB.
2. If HTTP request connects to RAC node from the next time with a same HTTP session ID, that request borrows a connection to the same RAC node by Web Session Affinity.

Performance Improvement

Test Case

The performance of the case of using Web Session Affinity is compared with the case of not using Web Session Affinity. Which RAC node receives requests is determined by workings of GridLink data source in this test case.

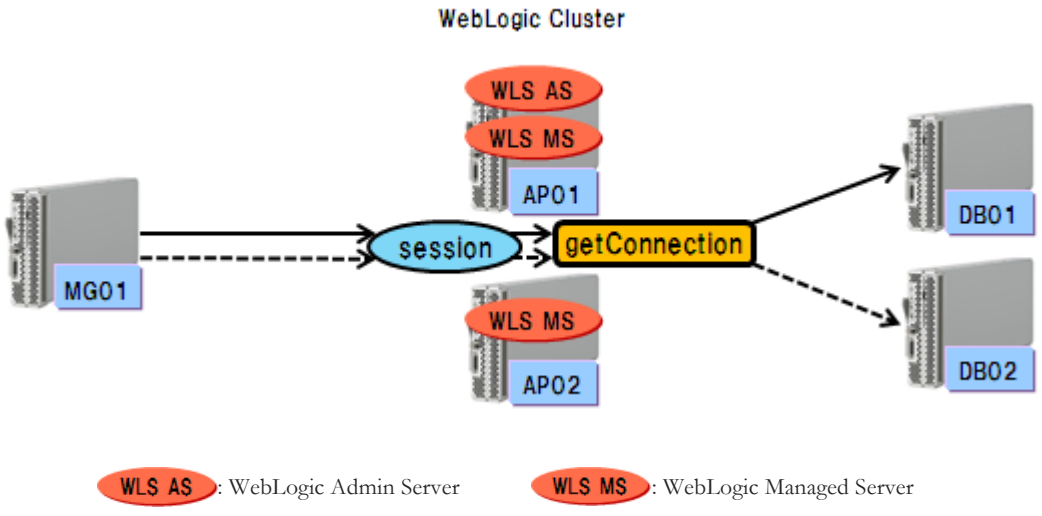


Figure 6-1 test environment for Web Session Affinity

In a case of using Web Session Affinity, HTTP session associated with RAC node sends request to the same RAC node always until its session is expired. Amount of data block transfer between RAC nodes by cache fusion is decreased because database local cache is often used.

In case of not using Web Session Affinity, HTTP session isn't associated with connection to RAC node. Therefore, the request may be sent to different RAC node each time that request get connection to RAC. The response time may increase because data block transfer between RAC nodes by cache fusion may occur.

Results

Left chart in Figure 6-2 shows comparison of response time. Right chart is comparison of interconnect traffic between RAC nodes. Average response time decreased by half, and interconnect traffic was largely decreased by Web Session Affinity.

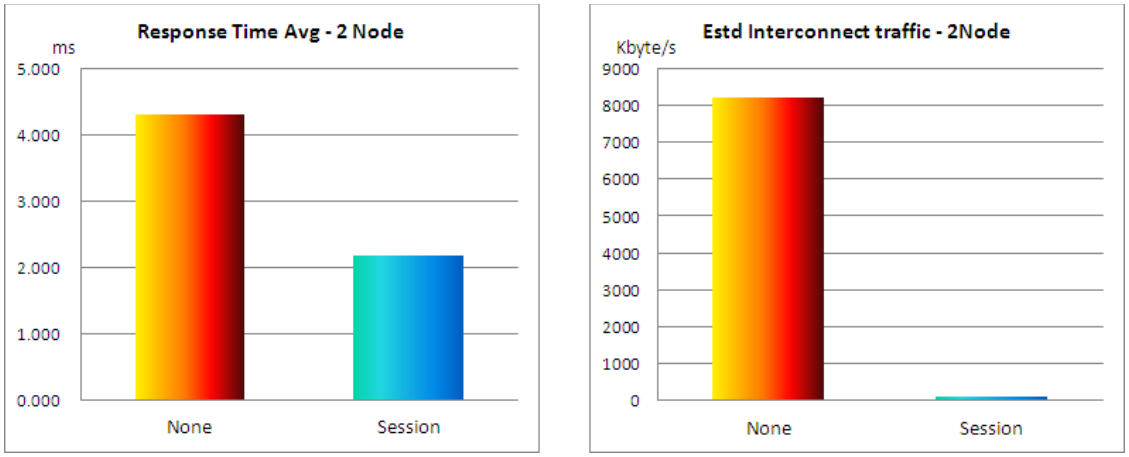


Figure 6-2 Improvement by Web Session Affinity in 2 node RAC

Figure 6-3 is as a result of the case increased in 4 nodes. The case of 4 nodes is similar to the case of 2 nodes, average response times decreased to half, and interconnect traffic largely decreased by Web Session Affinity.

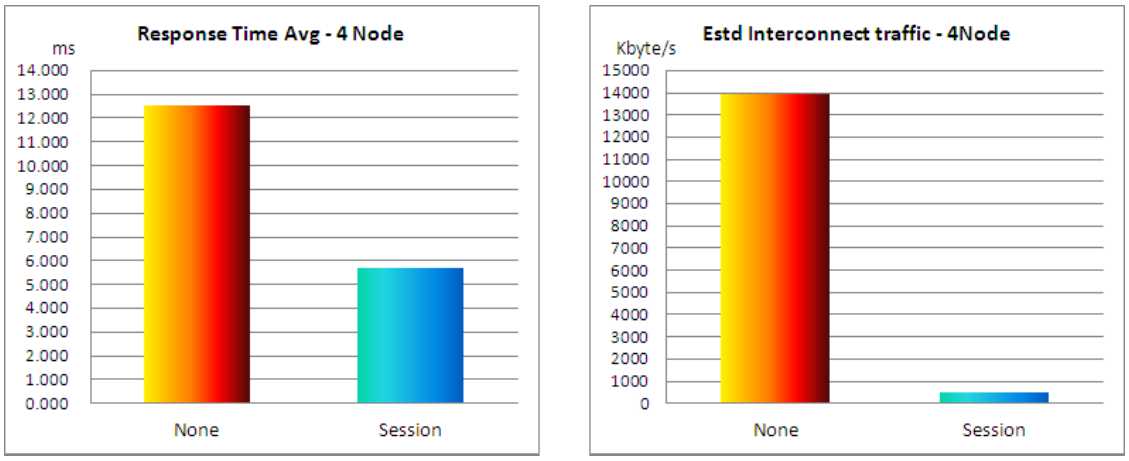


Figure 6-3 Improvement by Web Session Affinity in 4 node RAC

In our application of this test, the improvement effect in the response time and the interconnect traffic was obtained by web session affinity.

Intelligent Switching of Affinity Flag

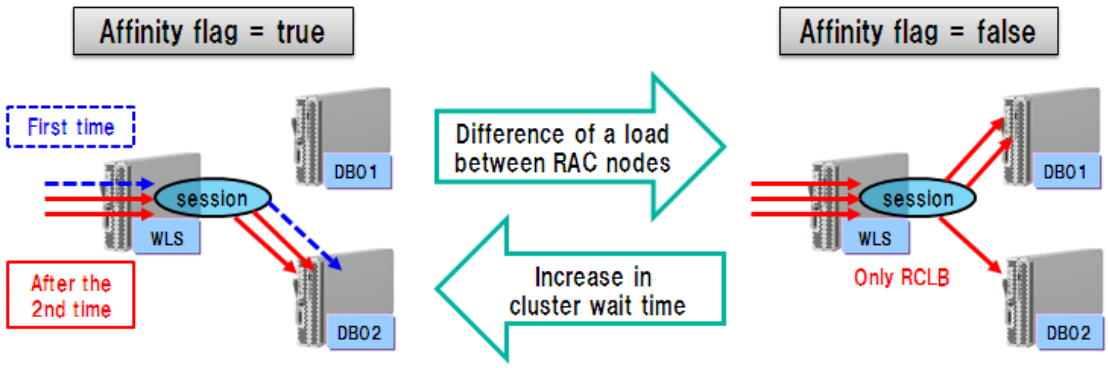


Figure 6-4 Auto switch of Affinity flag

TRUE or FALSE of Web Session Affinity flag is switched automatically according to cluster wait time and RCLB imbalance. If affinity flag is TRUE, Web Session Affinity works to get connection from pool as "Main Behavior" in Page.26. If affinity flag is FALSE, Only RCLB works to get a connection from a pool.

Continuity of Affinity Context

If session is continued by session replication of WebLogic Cluster function, A HTTP session accesses a RAC node which was accessed before failure in WLS. In other words, Affinity is also continued after failure in WLS, and the effect which decreases a cash fusion is continued. Because of this behavior, affinity even reduces a lot of cache fusion which is occurred by access to DB just after failure in WLS.

Figure 6-5 shows that HTTP session accesses RAC node which was accessed before a WLS failure in case session replication is used.

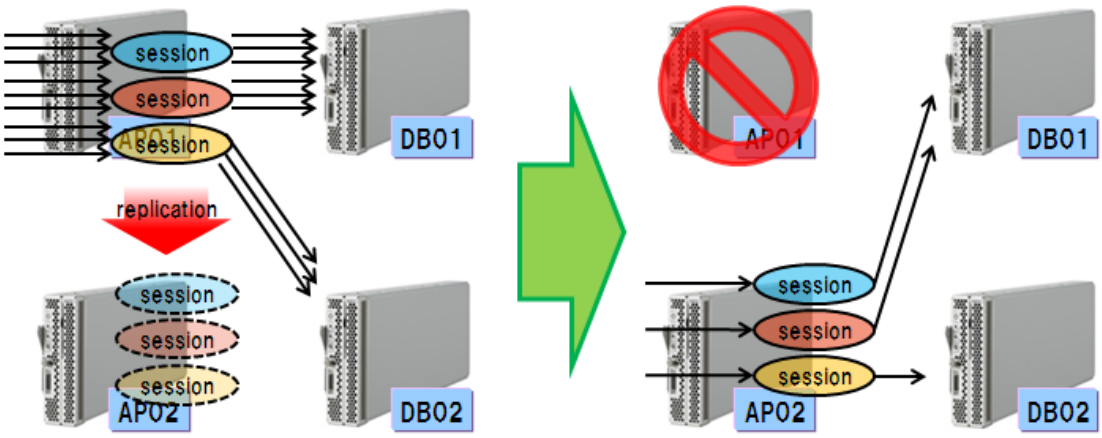


Figure 6-5 continuity of Affinity by session replication

In case of using "Coherence*Web" for continuity of HTTP session, affinity after failure in WLS is also continued because information of Web Session Affinity is included in HTTP session.

7. Fast Connection Failover (FCF)

Rapid Failure Detection and Kind of Detectable Failure

This chapter describes what kind of failure can be detected rapidly in FCF. In addition, time to recover a request is also described.

Depending on the situation or failure timing, a request needs to wait timeout on client side in order to notice failure database. If FCF is used, a survived RAC node sends FAN event to WLS so that a hanging request receives an event and can notice an error rapidly.

Test Case

The objective of this test case is that hanging request can be recovered rapidly by FCF. A case of getting connection after VIP failover or a case of failure detection by WLS connection test are not covered in this test.

Figure 7-1 shows parts that made failure simulation occur by this test.

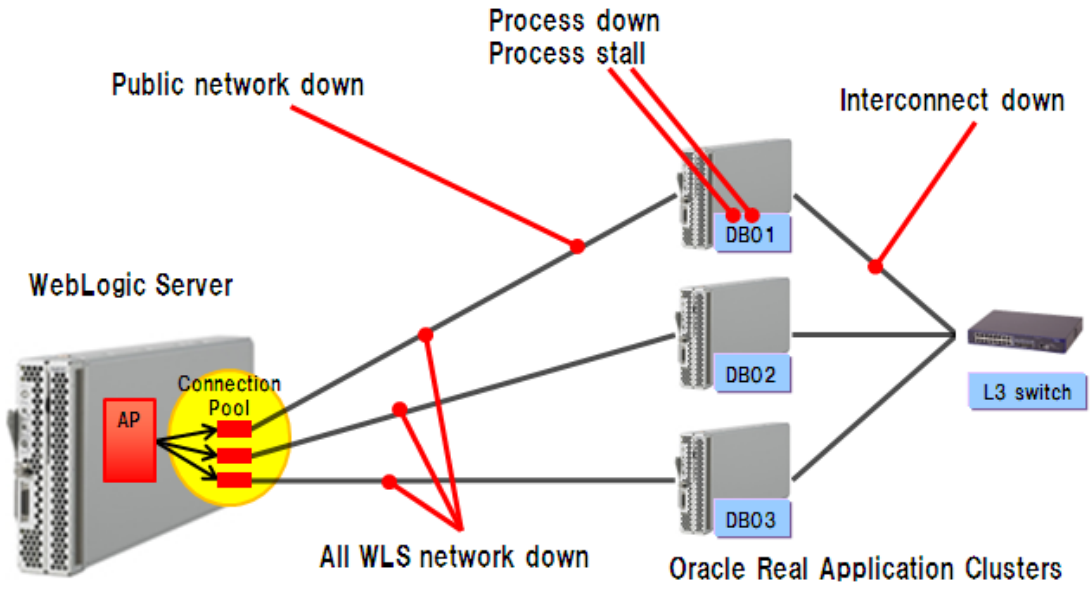


Figure 7-1 kinds and points of failure

Test shows if time that executing SQL returns an error to application is reduced by FCF. In order to simulate several kind of failure, the following ways were taken.

Process down:

SIGKILL is sent to LGWR process. This test case is simulation of Oracle instance down.

Process stall:

SIGSTOP is sent to selected process so that SQL hang up long time. This test case is simulation which Oracle can't properly process according to machine failure or OS fault.

Public network down:

A public network cable is disconnected physically.

Interconnect down:

Only one of the three interconnect cables is disconnected physically.

All physical networks down:

All public network cables are disconnected physically.

Results

Table 2 shows kinds of failure and how long hang up time can be shorted by FCF.

Table 2 kind of failure and client hang up time

kinds of failure	No FCF	FCF
Process down	Immediately	Immediately
Process stall	6min 20sec	6min 10sec
Public network down	9min 30sec TCP keep-alive timeout	15sec FAN event from an available Node
Interconnect down (RAC 3 node)	5min The 1st probe of TCP keep-alive timeout	33sec FAN event from all available Node
All public networks down	9min 30sec TCP keep-alive timeout	9min 30sec TCP keep-alive timeout

The TCP keep-alive time-out was changed to 9min 30sec for test from 2 hours of default.

Hung up time on public network down and interconnect down can be shorted, so it can be said that FCF is effective by kinds of failure.

Detailed results are following.

Public network down:

When a public network is downed, VIP failover occurs and service on DB01 is downed. An available RAC node notifies WLS of service down on DB01 through an available public network.

Interconnect down:

RAC nodes send heart beat to other node through interconnect for the alive monitoring. If interconnect of DB01 is downed, DB01 which can't receive a heartbeat is excluded from the cluster after a while. A surviving node notifies WLS of DB01 down through a surviving public network.

The following are kinds of failure which can't be detected rapidly by FCF.

Process stall:

In this test case, it takes long time for RAC node to detect other RAC node failure. RAC node don't send FAN event unless RAC node can detect the failure. Therefore, application can't notice RAC node failure rapidly by FCF.

All public networks down:

There is no network route to send FAN event to WLS from RAC, so application can't notice RAC node failure rapidly by FCF.

Solution for process stall by CLUSTERPRO MC ApplicationMonitor

If the function of CLUSTERPRO X HA/ApplicationMonitor which is HA clustering software produced by NEC is used, RAC node which has stall database instance is killed to send FAN event so that application can detect a failure rapidly.

Please see the link below for further details.

http://www.nec.co.jp/pfsoft/clusterpro/mc_ha/index.html (Japanese Only)

Behavior of Failback

Figure 7-2 shows the number of physical connections in period which process failover and failback. This test environment is configured by 2 nodes RAC, Connection load balancing goal is LONG, and 10 physical connections to each RAC node are generated.

After a failback, the number of the physical connection becomes equal to each RAC node by the rebalancing process.

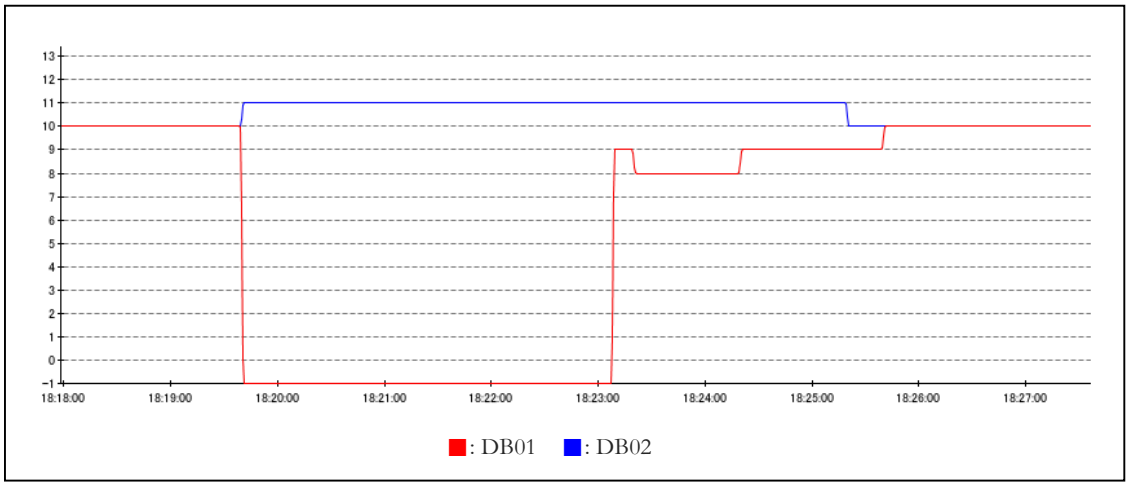


Figure 7-2 the number of physical connections in period which process failover and failback

The number of physical connection to failback node is changed by connection pool size or borrowed connection count. Therefore, the number of connection to each RAC node may not be equal when a failback occurs. However, after a failback, rebalancing process equalizes the number of connections to each RAC node.

Figure 7-2 shows the number of physical connections. Detail of process flow is below.

Failover (18:19:40-)

WLS receive DOWN event in FAN from RAC node; physical connections to failure node (DB01) are closed.

Failback (18:23:10-)

WLS receive UP event in FAN from RAC node; physical connections to recovered server (DB01) are generated. The number of generated connections is changed by connection pool size or borrowed connection count.

Rebalancing process

There is imbalance of physical connections to each RAC node when failback occurred, but the connection number to each RAC node becomes equal by rebalancing process.

8. Performance Improvement by Dynamic Changes in RAC Topology

FCF can't notify only RAC node failure but also the topology change in RAC. New RAC node can be added dynamically to running system by using notification of topology change and run-time load balancing.

Test case

We confirmed a performance improve by adding a RAC node dynamically to a system which has high CPU load.

Figure 8-1 shows this test environment. At first, the RAC cluster consists of 3 RAC nodes. A workload from WLS is applied to each RAC node so that the CPU utilization of each RAC node will be 80%. DB04 node is added to RAC while applying a load to 3RAC nodes from WLS. Because WLS access to RAC by SCAN, It's possible to access DB04 without changing configuration of WLS.

WLS receives the notice to which DB04 was added by FCF, and generates physical connections to DB04.

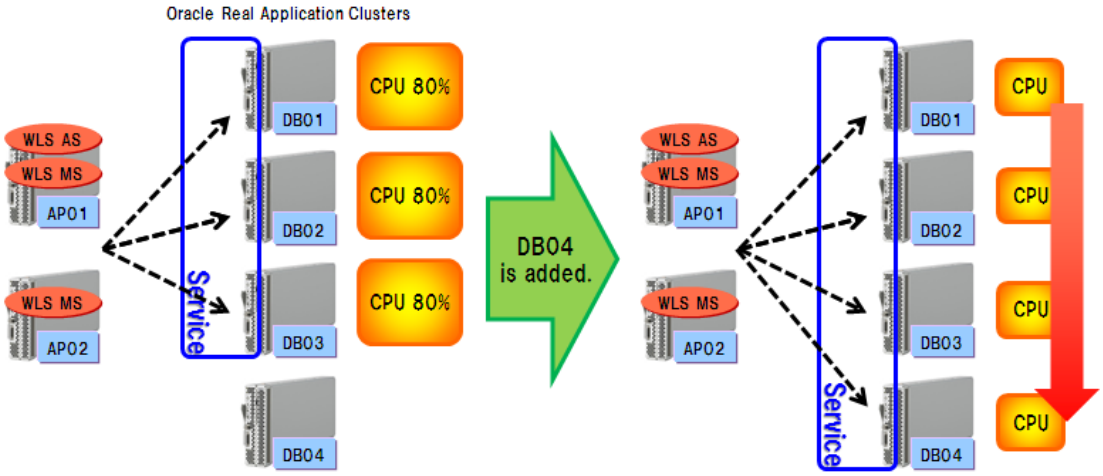


Figure 8-1 Test environment of adding a RAC node dynamically

Results

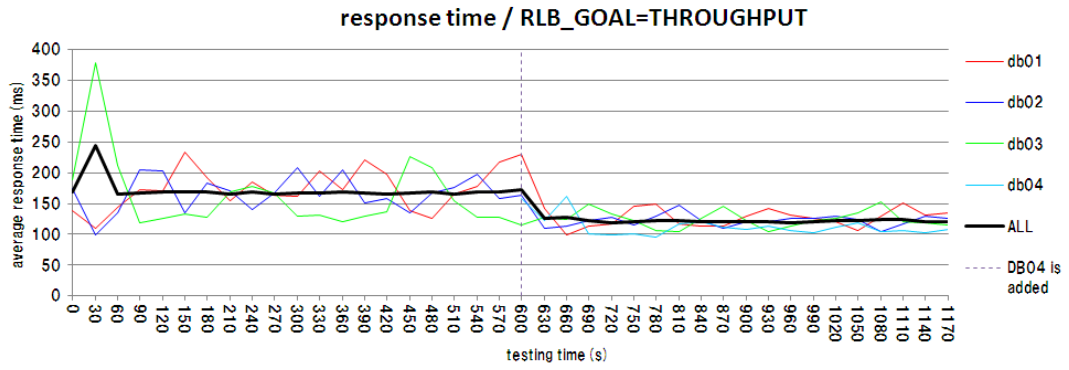
Figure 8-2 shows the response time and the percentage of each RAC node.

The average response time when WLS apply a load to 3 RAC nodes was about 170 ms, but after DB04 node was added, the response time is reduced to about 120 ms because a load is dispersed.

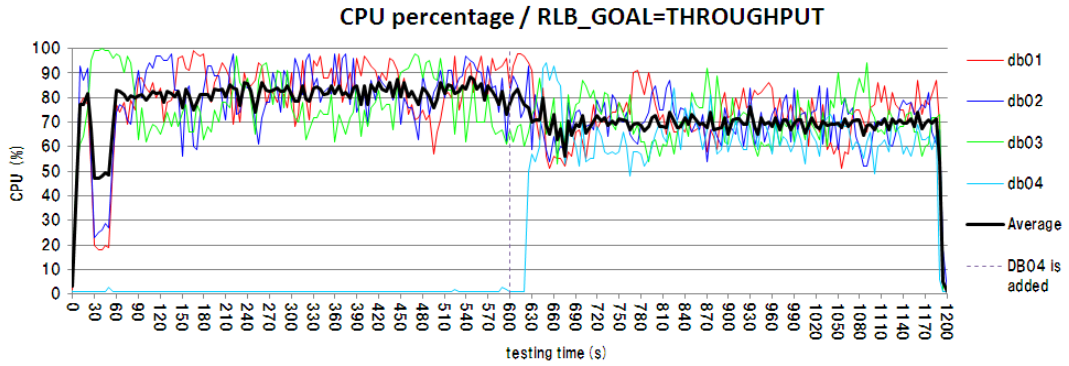
About the percentage of each RAC node, the percentage of the distribution of DB04 which was added later changed to same degree of the other nodes. A load also balances after node addition.

("ALL" is average of response time in all RAC nodes)

Response time



CPU percentage



The percentage which RAC node is used by WLS (CurrentWeight)

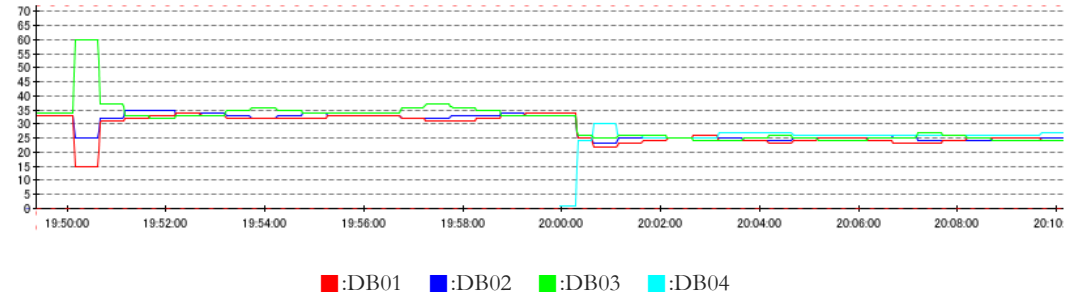


Figure 8-2 Response time and CurrentWeight when RAC node is added

The above is a result of a case which RCLB goal is THROUGHPUT. In case of SERVICE_TIME, the issue was occurred before DB04 was added and RCLB balance became unstable.

We don't comment the result in case of SERVICE_TIME in here because RCLB balance became unstable due to this issue before DB04 is added. The detail of this issue was described in "Issue in case of SERVICE_TIME" in Chapter 5.

9. Conclusion

Our test results indicated the effect of the GridLink data Source which is an integration solution of Oracle WebLogic Server 12c and Oracle Real Application Cluster 11gR2.

Advantage as the product

- Difficult mechanisms which consist of application function or complicated system operation can be achieved easily as product function by using GridLink data source.

Commonality of integration know-how

- Test and environment in this paper was planned based on system integration know-how of WebLogic Server which NEC has. A GridLink data source is a new function. But we found in this test that recommended configuration, detail of JDBC functions, monitoring method and other know-how can be diverted from these of JDBC data source. Much of the know-how used in NEC's system integration in the past can be used effectively even if using GridLink data source.

We tested each function of GridLink data source with different use cases. A GridLink data source can achieve the following.

High performance:

When a workload converges to a certain RAC node, dynamic load balancing by RCLB decreases a percentage of requests to high load RAC node, and assign a percentage of requests to low load RAC node. RCLB solves a long response time in high RAC node, and improves the average of response time in system. So even if high load occurred in a certain RAC node, service can be offered stably.

Web Session Affinity improves local cache hit rate and decreases cache fusion occurrence. So response time and interconnect traffic can be decreased.

Availability:

Even if a failure occurs in a RAC node, a request can be canceled by FCF without waiting for TCP keep-alive timeout. Time until the detection of failure becomes short, so it's possible to return an error reply to end user early.

Scalability and Serviceability:

Using Dynamic Changes in RAC Topology by FCF, RAC node can be added to running system or removed from running system. Additionally, if SCAN is used, it isn't need that WLS data source configuration change even if RAC topology changes.

This is a strong point which improves the serviceability. Combining FCF and know-how of multi data source, RAC topology change requirements to need complex operations will be able to be achieved in short term.

Empowered by Innovation NEC

Author: Yuki Makita

NEC Corporation
7-1, Shiba 5-chome. Minato-ku, Tokyo

ORACLE

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

White Paper Title
June 2013
Author: Frances Zhao-Perez

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