Oracle Database Administration Training Session – Day 19

Introduction to Oracle RAC

What is Oracle RAC?

Oracle Real Application Clusters (RAC) is a database clustering solution that allows multiple servers (nodes) to access and operate on a single database instance concurrently. It provides high availability, scalability, and fault tolerance by distributing workloads across multiple interconnected nodes in a cluster.

Key Features:

- **High Availability**: RAC ensures that the database remains available even if one or more nodes fail.
- **Scalability**: Additional nodes can be added to the cluster to handle increasing workloads.
- **Load Balancing**: Workloads are automatically distributed across all available nodes for optimal performance.
- **Fault Tolerance**: Automatic failover to other nodes in case of node or instance failure.

Benefits and Use Cases of Oracle RAC

Benefits:

1. **Continuous Availability**:
    - Ensures 24/7 database availability by eliminating single points of failure.
    - Suitable for mission-critical applications.
2. **Improved Performance**:
    - Distributes workload across multiple nodes for better performance.
    - Ideal for environments with heavy transactional or analytical workloads.
3. **Scalability**:
    - Seamlessly add nodes to the cluster without downtime.
    - Supports growing application demands.
4. **Cost-Effective**:
    - Runs on commodity hardware, reducing infrastructure costs.
5. **Transparent Application Failover (TAF)**:
    - Automatically redirects users to a surviving node during failures, minimizing disruption.

Use Cases:

- **E-Commerce Platforms**:
    - Supports millions of simultaneous transactions with high availability.

- **Banking and Financial Systems**:
    - Ensures reliable operations for transactional databases.
- **Airline Ticketing Systems**:
    - Handles high volumes of real-time bookings with zero downtime.
- **Healthcare Systems**:
    - Maintains critical medical records and patient data availability.

Architecture Overview

Oracle RAC architecture consists of the following core components:

1. Shared Storage

- Centralized storage (SAN/NAS or Oracle ASM) is accessible by all nodes in the cluster.
- Stores database files, redo logs, control files, and archive logs.

2. Interconnect

- A high-speed private network that connects all nodes in the cluster.
- Facilitates node-to-node communication and synchronization.
- Uses protocols like **Gigabit Ethernet** or **InfiniBand**.

3. Nodes

- Physical or virtual servers that run Oracle Database instances.
- Each node has its own memory and CPU resources.

Key Components

1. Oracle Clusterware

- Manages the cluster environment, including node membership, failover, and cluster communication.
- Components:
    - **Voting Disk**: Tracks active nodes in the cluster.
    - **OCR (Oracle Cluster Registry)**: Stores configuration information for the cluster.

2. Oracle Grid Infrastructure

- Combines Oracle Clusterware and ASM.
- Provides tools and utilities to manage cluster resources.

3. ASM (Automatic Storage Management)

- Oracle's file system and volume manager for shared storage.
- Simplifies database storage management by automatically distributing data across available disks.

- Provides redundancy levels (NORMAL, HIGH) to prevent data loss.

RAC vs. Single Instance Database

Feature	Oracle RAC	Single Instance Database
Availability	High availability with failover capabilities	Limited availability, downtime possible
Scalability	Horizontally scalable by adding nodes	Limited by server hardware
Load Balancing	Automatic workload distribution	Not available
Cost	Higher initial investment	Lower cost
Use Cases	Mission-critical, high-availability systems	Small-scale, standalone applications

---

How Oracle RAC Works in Emirates Airline Ticketing System

In an airline ticketing system, RAC ensures the following:

- **High Availability**:
  - If one server fails during peak booking hours, the system redirects users to another node seamlessly.
- **Scalability**:
  - During seasonal surges, such as holidays, additional nodes can be added to handle the increased workload.
- **Load Balancing**:
  - Transactions like seat selection, booking confirmation, and payment processing are evenly distributed across all nodes.

Practical Example

1. **Scenario**: Emirates Airline ticketing sees a spike in bookings during a flash sale.
   - **RAC Setup**:
     - Shared storage holds customer data, booking details, and transaction logs.
     - Interconnect ensures all nodes are synchronized to prevent double booking.
     - Each node handles a subset of users, maintaining a balanced workload.
   - **Failover**:
     - If a node fails due to hardware issues, active transactions are moved to other nodes automatically.
2. **Outcome**:
   - Continuous uptime during peak traffic.

- No loss of booking or transaction data.
- Users experience seamless service.

By leveraging Oracle RAC, organizations can build robust, high-performing database environments suitable for dynamic, real-time applications like airline ticketing systems.

In an Oracle RAC environment, **Load Balancing** is achieved through a combination of Oracle's built-in features and external configurations that distribute the workload across multiple nodes in the cluster. Here's what is used:

1. Oracle RAC Load Balancing Features

Client-Side Load Balancing

- Distributes connection requests from client applications across all available RAC nodes.
- Configured using:
  - **Easy Connect Naming Method**: Automatically resolves and distributes connections to available instances.
  - **TNSNAMES.ORA**: Includes LOAD_BALANCE=ON in the connect string.

Example TNS Entry for Load Balancing:

```
MY_RAC_DB =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = node1-vip)(PORT = 1521))
   (ADDRESS = (PROTOCOL = TCP)(HOST = node2-vip)(PORT = 1521))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = my_service)
   (LOAD_BALANCE = ON)
  )
 )
```

- **VIPs (Virtual IP Addresses)** are used for failover and routing.

Server-Side Load Balancing

- **Oracle Net Listener** dynamically redirects incoming client connections to the least loaded instance.
- Uses:
  - **Service Metrics**: Monitors response times and system metrics to balance workload.
  - **Load Balancing Advisory (LBA)**:
    - Continuously evaluates workload on each instance.
    - Provides client applications with connection routing information.

2. Load Balancing Advisory (LBA)

- A component of Oracle RAC that dynamically routes client requests based on:

- **Response Time**: Directs requests to the fastest node.
- **Service Level Agreements (SLAs)**: Ensures critical workloads are prioritized.

Configuration:

- Enabled at the service level:

```
BEGIN
  DBMS_SERVICE.MODIFY_SERVICE(
    service_name => 'my_service',
    goal => DBMS_SERVICE.GOAL_THROUGHPUT,
    clb_goal => DBMS_SERVICE.CLB_GOAL_SHORT
  );
END;
/
```

- **Goals**:
  - **Throughput**: For transaction-heavy workloads (e.g., airline ticketing systems).
  - **Response Time**: For time-sensitive applications (e.g., real-time seat selection).

3. Third-Party Load Balancers (Optional)

For additional control, enterprises may use **external load balancers** to distribute incoming connections:

- **F5 BIG-IP** or **Cisco Load Balancer**:
  - Distributes client traffic across Oracle RAC nodes.
  - Provides advanced features like session persistence and SSL termination.

4. Connection Pooling

- **Oracle Universal Connection Pool (UCP)**:
  - Manages and balances connections to RAC nodes.
  - Reuses existing connections for optimal performance.
- Example configuration:

```
PoolDataSource pds = PoolDataSourceFactory.getPoolDataSource();
pds.setConnectionFactoryClassName("oracle.jdbc.pool.OracleDataSource");
pds.setURL("jdbc:oracle:thin:@//myrac-db-scan:1521/my_service");
pds.setInitialPoolSize(10);
pds.setMaxPoolSize(50);
```

5. Scan (Single Client Access Name)

- Introduced in Oracle RAC 11g, **SCAN** simplifies load balancing and failover:
  - A single hostname (e.g., myrac-scan) is used for all client connections.
  - SCAN resolves to multiple IP addresses, enabling distribution across nodes.

Benefits of SCAN:

- Automatic node discovery and failover.
- No need to reconfigure clients when nodes are added or removed.

Use Case in an Emirates Airline Ticketing System

Scenario:

- Thousands of users are booking tickets during a flash sale.
- Workloads include:
    - Seat selection queries.
    - Real-time payment processing.
    - Ticket generation and confirmation.

Load Balancing in Action:

- **Client-Side**:
    - Users are directed to less busy nodes using LOAD_BALANCE=ON and SCAN.
- **Server-Side**:
    - Oracle Listener redirects queries based on node availability and response time.
- **LBA**:
    - Ensures that transactions requiring faster response times (e.g., payment) are prioritized.
- **Outcome**:
    - No server is overwhelmed, ensuring seamless ticket booking for users.

By combining these Oracle RAC features and tools, load balancing ensures an optimal, highly available, and efficient environment, even during peak usage.