

# Oracle Application Server 10g High Availability

*An Oracle White Paper  
January 2004*



# Oracle Application Server 10g High Availability

1. Introduction.....	4
2. High Availability Guidelines.....	6
Redundancy.....	6
Death Detection, Heartbeats, and Auto-Restarts.....	6
State Replication, Recovery, and Routing.....	7
Clustering.....	7
Connection Management.....	8
Virtual Naming and Indirection.....	8
Resource Provisioning and Load Balancing.....	8
Rolling Upgrades.....	8
Error Reporting and Diagnostics.....	9
Systems Management.....	9
Backup and Recovery.....	9
Disaster Recovery.....	9
3. Oracle Application Server High Availability.....	11
Process Level High Availability.....	11
Infrastructure High Availability.....	11
Cold Failover Cluster (CFC).....	11
Middle Tiers in a Cold Failover Cluster environment.....	12
Active Failover Clusters.....	13
Identity Management Service Replication.....	14
Installing Metadata Repository in an existing RAC database.....	14
Middle Tier High Availability.....	15
Web Cache.....	15
Oracle HTTP Server (OHS).....	16
Mod_oc4j.....	16
Mod_plsql.....	16
OC4J – Deployment.....	17
OC4J – HTTP Session State.....	17
OC4J – EJBs.....	17
OC4J – TAF.....	18
Portal, Reports, Forms and other Middle Tiers.....	18
Enterprise Manager and Distributed Configuration Manager.....	19
Disaster Recovery.....	20
4. Summary.....	21
5. References.....	21

## 1. INTRODUCTION

Oracle Application Server 10g (9.0.4) introduces a rich set of features to ensure end-to-end high availability of mission critical applications deployed on Oracle Application Server. It provides an architecture with no single point of failure and a design that ensures minimal planned or unplanned downtime.

Oracle Application Server's clustering and configuration management architecture is designed to enable it to be efficiently deployed on a Grid. Grid computing is a computing architecture that effectively pools large numbers of servers and storage to create a virtual computing resource.

A Grid computing environment is a parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across multiple administrative domains based on their availability, capability, performance, cost, and quality-of-service requirements. Its process management facilities enable Application Servers to be added on-demand as scalability and availability requirements increase; its workload management and load balancing facilities can detect and route requests among a cluster of Application Server instances to most effectively use idle capacity; and its Distributed Configuration Management (DCM) facilities allow new Application Server instances to be very quickly and efficiently provisioned before they are added into a Grid.

Oracle Enterprise Manager (EM), also called Grid Control, is used to manage the Oracle Application Server Grid environment. Grid Control is the primary interface for performing all management tasks in a grid. A reliable and scalable multi-administrator repository offers the option of cooperative management. Using the Oracle Grid Control product family, database administrators and IT managers can increase productivity, deliver better services, and reduce the total cost of information systems.

This paper provides discusses the high availability facilities in Oracle Application Server 10g (9.0.4). The configuration details required for administrators to implement the solutions described herein are not part of this document – but are referenced where appropriate.

It is assumed that the reader has an introductory knowledge of Oracle Application Server 10g (9.0.4) and its various components. Hence, some terms and components may not be defined prior to their usage. For more information about such terms, please read the Oracle Application Server Concepts Guide [1].

This document is broadly divided into two sections: First, it covers the standard traits of any highly available system and describes how Oracle Application Server provides them. Second, it takes an Oracle Application Server view and takes each of the different tiers (install types), components, and services the product provides and discusses how to make them highly available.

High Availability is a way of guaranteeing results. It is a type of insurance, and insurance always comes at a cost. The costs and value delivered by Oracle

Application Server's high availability options are woven throughout their discussion, and a later section of this paper compares and contrasts different high availability solutions.

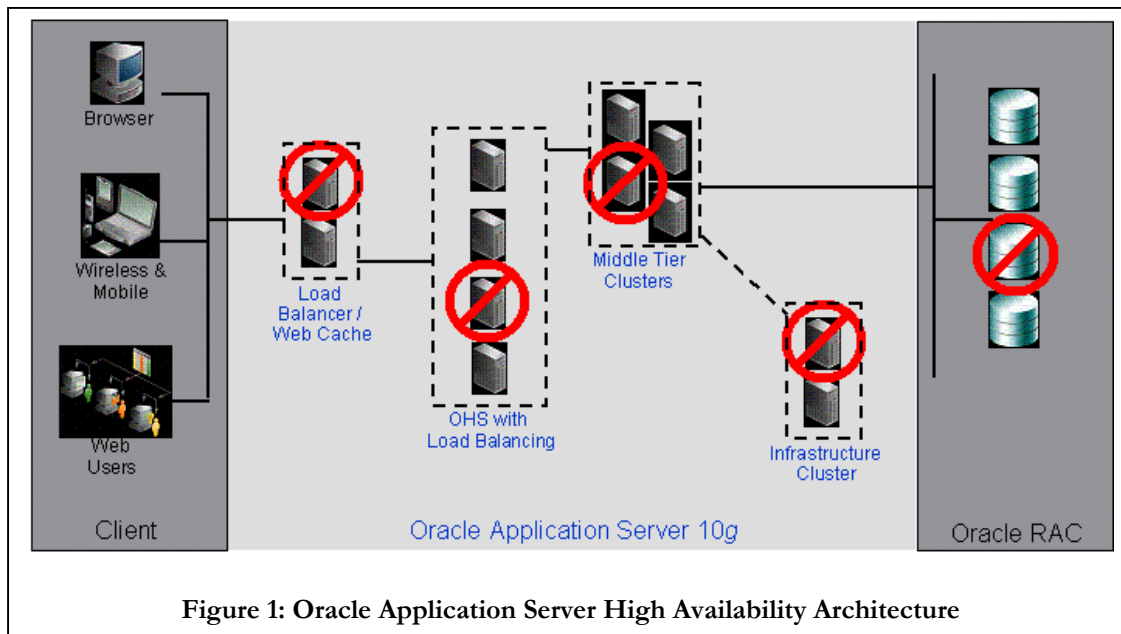
## 2. HIGH AVAILABILITY GUIDELINES

There is a generally recognized set of standard attributes used to discuss high availability in computing systems [2], [3].

This section walks through these attributes and provides a perspective of the different facilities that Oracle Application Server uses to deliver on these high availability criteria.

### Redundancy

Any highly available system has to have redundant components to mask failures in individual components. All Oracle Application Server components can be deployed in a redundant fashion to make their services more available (see Figure 1). In some cases, the systems provide redundancy with an active-passive configuration where the passive components are only used when the active component fails (e.g., Infrastructure Cold Failover Cluster). In other cases (e.g., Middle Tier or Web Cache clusters), the components are active-active, where all instances of the



components are able to service the requests simultaneously, adding scalability to redundancy

Installation flexibility allows different levels of redundancy for different components. In addition, these redundant components can talk to each other while thus maintaining a group

state, such as Web Cache Cluster or OC4J islands.

Making the core components redundant automatically makes applications and other components, such as Portal or Reports highly available.

### Death Detection, Heartbeats, and Auto-Restarts

Failures are inevitable. Systems need a way to detect them and react to them.

All Oracle Application Server processes – middle tier and infrastructure– are “self healing” – any failure in the process is detected under a central process management umbrella. Any time heartbeats are missed beyond certain threshold, or

a crash detected, the component is restarted. Certain other system metrics can also be leveraged to determine restart conditions.

Since no system administrator intervention is required, the failure window is extremely small – and thus largely transparent from the end users.

During startup or restart of any component, the system ensures any components required for dependent functionality are all running. If any of these required components are not available, Oracle Application Server can start these components – even if they are on another machine and part of a different Oracle Application Server instance.

### **State Replication, Recovery, and Routing**

When processes are started or the application's control is transferred to a redundant process, it is necessary that the client application state be adequately recreated. In general, this operation requires the individual processes (and applications) to persist their state and read and recreate that state on startup. Oracle Application Server provides two facilities for J2EE applications: (a) replicating state declaratively, and (b) replicating it using Java Object Cache, which does require programmatic object replication across processes.

In the cases when state is not an issue, stateless routing suffices. State based routing using *cookies* is supported by all relevant components. Additionally, smart replication set based routing is supported by mod\_oc4j – if an OC4J process cannot be routed to, it determines which other process has the replicated state and tries routing to one of those processes. This capability allows for transparent failover.

On the Infrastructure tier, the services do not retain any state in the server. Thus, the cookies or persisted data are sufficient to recreate state.

### **Clustering**

Clustering multiple systems together lets them behave as one to the external clients. Oracle Application Server supports this at multiple levels (see Figure 1):

- (a) Web Cache clusters make a group of Web Cache instances work together to have a *shared distributed cache*
- (b) Oracle Application Server Clusters ensure configuration replication across all member Middle Tier instances. This automation reduces failures caused by mis-configured systems due to human error. It also enables the session replication discussed above.
- (c) Hardware clusters allow Oracle Application Server to leverage industry-standard techniques such as Cold Failover Clusters (CFC) as well as Oracle's own technology such as Active Failover Clusters and Real Application Clusters which provide further enhanced features that run on hardware clusters, and
- (d) J2EE Clusters, which provide J2EE session state replication.

### **Connection Management**

In many situations, clients connect to a service and then reuse the connection. However, if the server process on the other end is restarted, the connection may need to be re-established.

Oracle Application Server components, such as OC4J, make sure that if a reused connection fails, the connection is retried before a failure is announced to upstream components. This feature helps avoid extended impact of a system restart – especially the Oracle Application Server Infrastructure restarts.

### **Virtual Naming and Indirection**

From a high availability perspective, it is a good idea to use names instead of addresses wherever possible. This practice can enable quicker recovery by re-adjusting the name to address mapping in case of machine failures.

Using a virtual name and/or virtual address further helps by allowing multiple systems to be configured to respond – improving availability. This capability is leveraged in the context of a Cold Failover Cluster setup.

### **Resource Provisioning and Load Balancing**

Intelligent routing and process management enables Oracle Application Server to do capacity driven routing – especially for Web Cache and the OC4J (Oracle's J2EE container). In addition, the workload management provisioning algorithms (such as weighted or metric based routing) allow finer grained control in case the application systems need to be loaded differently. This type of dynamic workload management is a key enabler of Grid Computing.

Some components, such as Oracle HTTP Server, Oracle Internet Directory or Web Cache, can increase the number of server processes or threads to respond to a spike in requests and then reduce the number of processes to a lower, steady state number. Thus in general consuming fewer resources unless required by system load. This feature enables the provisioning of “on demand” resources in a grid.

### **Rolling Upgrades**

Every system needs updates or maintenance activity. Minimizing the downtime during this operation is important for supporting highly available systems. Oracle Application Server formally supports the upgrade of infrastructure and middle-tiers from 9.0.2 to 9.0.4 with minimal operational impact [4]. Additionally, the new checkpointing capability [also called archival and retrieval] enables one system configuration to be captured and then re-applied to another system. This feature allows one system to be upgraded and tested before its configuration is cloned. Similarly, Web Cache clusters allow incremental update of configuration to the members.

An external load balancer or the load balancing provided by Web Cache can always be leveraged to hide the system component being upgraded – including

applications. The EJB client proxy classes use dynamic discovery to determine available EJB servers, thus automatically adjusting in case a server is down for upgrade.

### **Error Reporting and Diagnostics**

Monitoring logs is important since they can serve as a precursor to impending failures. Debugging facilities in the product help identify the root cause of problems earlier in the development and deployment cycle.

Oracle Application Server provides several types of logs and debugging levels to support these requirements. The component error logs are now loaded into a central repository that can be mined through Enterprise Manager. This centralization allows correlation of errors across systems and across components enabling quicker resolution – i.e. reduced downtime. Oracle Application Server Dynamic Monitoring System is now integrated into all components, giving several metrics that can be useful in determining the root cause of a slow down.

### **Systems Management**

Enterprise Manager provides quicker monitoring and administration of an individual Oracle Application Server system. In addition, clustering capabilities of Oracle Application Server allow configuration to be easily replicated across clustered instances – resulting in fewer configuration errors.

### **Backup and Recovery**

When a system cannot be repaired, it may be quicker to replace the failed system component instead of taking the time to immediately identify the root cause of the problem. For this purpose, administrators can keep hardware spare parts, or, in the case of Oracle Application Server, a system check pointed state.

The backup and recovery tool delivered with the product makes it easy to create this checkpoint and then restore it if necessary. Also, for localized problems involving only the OHS and OC4J components, Distributed Configuration Manager (DCM) provides a faster check pointing capability. In even simpler cases – such as those involving configuration errors – an *undo* operation is now included for quicker recovery!

### **Disaster Recovery**

All the discussion so far has concentrated on providing high availability for systems within a single site containing multiple redundant and self-healing components. However, in case of a disaster – e.g., an earthquake – all systems at the site might be unusable. For situations where the entire site is lost, Oracle Application Server Disaster Recovery provides site-to-site failover. This is discussed in more detail later in the paper.

The discussion above has been focused on a general discussion of high availability attributes and how Oracle Application Server follows the same. The next section,

will take a look at the different Oracle Application Server install types and components, and how to deploy these various components for better availability.

### **3. ORACLE APPLICATION SERVER HIGH AVAILABILITY**

Oracle Application Server deployment typically involves deployment of two tiers for the Application Server – an Infrastructure and multiple Middle Tiers.

The Infrastructure provides certain key services like Identity Management and storage and access to commonly used Metadata, while the Middle Tier typically runs the applications – J2EE or Portal, and provides the key run time services such as HTTP, JMS etc.

The following high availability discussion covers how to make these different services highly available.

#### **Process Level High Availability**

All Oracle Application Server processes in the Infrastructure or middle tier are monitored by Oracle Process Manager and Notification Server (OPMN) – and are restarted when a failure is detected. They are thus “self healing”. Additionally, any components on which these processes are dependent are re-started as well. The impact of a restart or detection of the death of a process is immediately communicated to routing components (ex. mod\_oc4j), so they do not route to a dead process. Moreover, a shadow process monitors the OPMN process itself.

Together, the OPMN instances across multiple Oracle Application Server instances form a communication network and provide network wide start, stop and status, akin to a system grid.

The process management infrastructure can also be extended to custom non-Oracle Application Server processes as well.

The next few sections focus on availability beyond this basic, but crucial process monitoring and restart capability.

#### **Infrastructure High Availability**

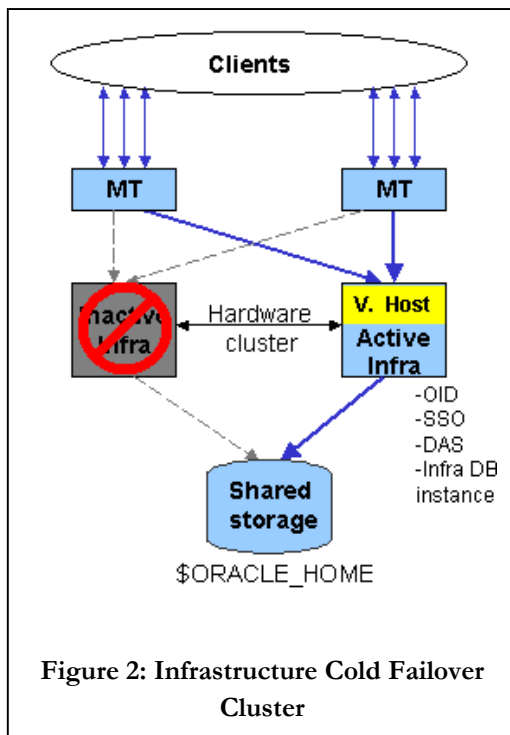
The Oracle Application Server Infrastructure provides three critical services: (a) Identity Management Service, which includes Single Sign-On (SSO) and Oracle’s LDAP Directory, Oracle Internet Directory (OID) and (b) Metadata Repository Service, which includes the metadata used by different Middle Tier components and is stored and managed in an Oracle database, and (c) Management Services, which include metadata that is used by Oracle Application Server Enterprise Manager to monitor and manage the system. The Infrastructure can be made highly available through a variety of techniques discussed below.

#### **Cold Failover Cluster (CFC)**

Multiple machines (typically two) can be clustered together using clusterware such as HP MC/Service Guard or Sun Cluster. This type of clustering is a commonly used solution for making a system highly available. As the name suggests, one node of the cluster is “cold”, passively waiting to take over in the event of a failure, while

the other is “hot”, or actively running the software. When the “hot” or “active” node fails, the clusterware restarts the software on the cold node to bring the system back online. So that both nodes of the cluster are able to run the software, it must be installed on a disk that can be shared (mounted) by either node. Also, the system must use a virtual hostname (and IP address). This virtual hostname is associated by the clusterware with the active node. External systems need never be reconfigured – they always access the software by virtual hostname addressing and no matter which node the software runs on, it can always be accessed by the virtual hostname. Note that the “cold” node need not be kept idle – it could be running other applications, but not the application that is running on the “hot” node.

Oracle Application Server Infrastructure is “cluster-aware” and is able to detect a cluster during installation. If detected, the Infrastructure can be installed on a



storage device that is shared by these nodes. This type of install also requires that a virtual hostname be used for the installation so that the Infrastructure can be failed over from one node to the other (see figure 2). When installing the Middle Tier instances, the hostname provided for referring to the Infrastructure node is the virtual hostname. Hence, Middle Tiers are able to address the Infrastructure regardless of which physical host the Infrastructure runs on.

This type of clustering is an easy, cost-effective way to achieve higher availability of the infrastructure services even though there is short downtime window during failover while the clusterware plumbs the virtual hostname (and IP) from one node to the other, mounts the shared disk on the previously cold node and restarts the Infrastructure on that node.

There is some impact on middle tier instances due to the infrastructure failover, especially in components such as mod\_plsql where the connections are cached. However, Oracle Application Server Middle Tier components and applications retry to establish the connection when a usage attempt fails, which keeps the failure from being visible to the

end-user in most cases.

### **Middle Tiers in a Cold Failover Cluster environment**

Middle Tiers can be independently clustered (even without a hardware cluster), so it seems counter-intuitive to want to run them on a hardware cluster. However, recognizing customer demand, Oracle has, in Oracle Application Server 10g (9.0.4) certified a configuration that allows Middle Tiers to be installed (using physical hostname, not virtual hostname) on *local* disks of each of the nodes of a hardware cluster. Requests across the two Middle Tiers are distributed using an external load balancer. While this configuration doesn't allow for the failover of Middle Tiers upon node failure, it does allow high availability because even if one of nodes fails,

the Middle Tier on the other node can continue servicing incoming requests. The infrastructure, in this scenario, is installed just as described above in the “Cold Failover Cluster” configuration.

For even higher availability of the Infrastructure, Oracle Application Server provides “Active Failover Clusters”.

**Active Failover Clusters**

Cold Failover clusters are easy to setup and manage, but have two drawbacks: (a) since only one system is active, multiple-system scalability cannot be achieved, (b) during transfer to the other node, the downtime can run into several minutes – which in some cases may be unacceptable.

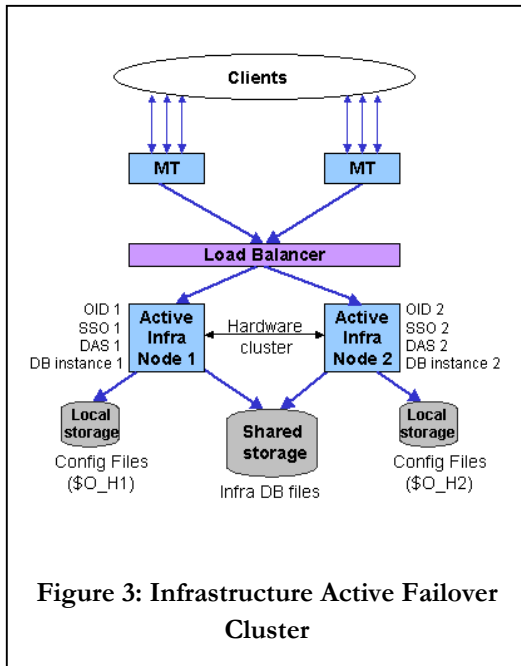
In Oracle Application Server 10g (9.0.4), Oracle introduces an exciting new feature: “Active Failover Cluster”. In contrast with Cold Failover Clusters, Active Failover Clusters allow Oracle Application Server Infrastructure to run on multiple nodes *simultaneously*. In this “Active/Active” setup, all Infrastructure processes are configured to run simultaneously on each node of the cluster (see Figure 3). A Load

Balancer sits in front of the hardware cluster to distribute incoming requests across each of the nodes, since each node is capable of handling any incoming request. If one of the nodes goes down, further incoming requests are immediately re-distributed by the load balancer among the surviving nodes of the cluster, thus eliminating any down time.

This configuration leverages the Real Application Cluster (RAC) feature of the Oracle Database for running the Infrastructure database. However, AFC also provides high availability for the mission-critical Identity Management Services of the Infrastructure. The Identity Management Services include several processes including, Single Sign-On, Oracle Internet Directory (OID), and Delegated Administration Service (DAS). Active Failover Clusters extend the type of high availability provided by a RAC database – they enable high availability of the entire Infrastructure.

Just like Cold Failover Cluster, an Active Failover Cluster does require a hardware cluster. Each node of the cluster contains a separate, local \$ORACLE\_HOME, which contains the configuration and binaries for running the Infrastructure on that node. In addition, all nodes access a set of shared data files used by the RAC database. The Middle Tiers are configured to point to the Load Balancer (as opposed to any individual physical node of the cluster). The complete setup is done with a single installation – the system administrator need not repeat the install on each and every node of the cluster, which makes the installation process very simple.

In the initial release of Oracle Application Server 10g, the Active Failover Cluster feature will be available only as a limited release feature: customers will not be able



**Figure 3: Infrastructure Active Failover Cluster**

to deploy this in a production environment. This feature will be made generally available in the near future.

### Identity Management Service Replication

One of the key services provided by Oracle Application Server Infrastructure is the Identity Management Service. Identity Management (IM) provides SSO and OID (Oracle's LDAP Directory) for user authentication and authorization. Hence, IM availability is very crucial to all applications leveraging that require these features.

The IM Service availability depends on the availability of its individual components:

(a) IM Middle Tier which provides SSO and is composed of the OHS and OC4J

with special IM related configuration, (b) OID LDAP process, and (c) the database, which contains identity data and stored procedures used by the IM Service for identity verification and management.

*IM Middle Tier* is stateless – hence, the standard technique of installing this component on multiple machines and front ending these servers with an external Load Balancer works appropriately. The IM Middle Tier processes can also be configured to go against multiple IM databases. In case one database fails, the IM Middle Tier automatically routes to another one, without requiring RAC for the database.

*OID* receives the incoming LDAP requests, which can all be routed through an IP level load balancer, routing to multiple OID deployments.

*IM data* can be replicated across different backend database instances using Oracle Database replication. With this configuration, multiple IM systems can be kept synchronized with respect to the IM data and metadata, with acceptable delays.

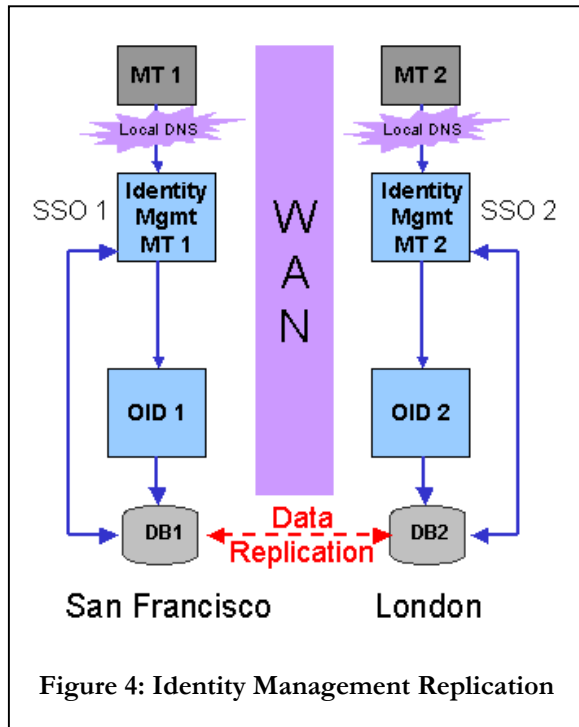


Figure 4: Identity Management Replication

This approach to IM service availability has a couple of

advantages: (a) it does not require expensive clustered hardware, (b) IM scaling can be tuned better depending on which component is required to be tuned, and (c) it provides a way to do geographically distributed active-active deployments of IM, a fairly common requirement for login performance and disaster recovery reasons.

### Installing Metadata Repository in an existing RAC database

If the administrator wishes to avoid installing the database as part of the Infrastructure, they have the option to install the Metadata Repository (MR) schemas in an existing Oracle database. Furthermore, this pre-existing database can be a RAC database, thus making the MR highly available. Note that, this feature only makes the MR data highly available. The IM services still need to be installed in an \$ORACLE\_HOME and thus may or may not be highly available.

## Middle Tier High Availability

The most common technique for making Oracle Application Server Middle Tiers highly available is to have multiple Middle Tiers with an external load balancer in front to distribute incoming requests across the available Middle Tiers (see figure 5). In addition, certain Oracle Application Server Middle Tier components support software-clustering techniques. This section covers ways to make Middle Tier components highly available.

### Web Cache

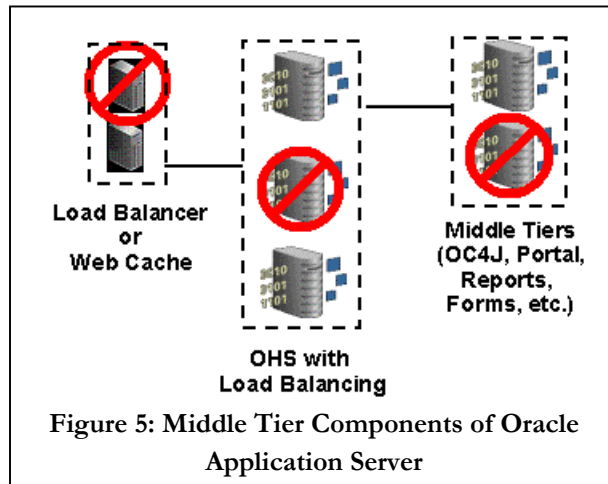


Figure 5: Middle Tier Components of Oracle Application Server

Web Cache is typically deployed on a set of machines separate from the rest of the Middle Tier. Web Cache instances are *glued* together by either: (a) a Load Balancer that balances across the multiple independent Web Cache instances (b) Web Cache clusters, which make these instances aware of each other, enabling them to have a *shared distributed cache*, thus improving system scalability as a whole. This advantage does not exist in option (a), where each Web Cache instance is standalone.

Furthermore, leveraging Web Cache can, in certain cases, hide failure in the backend, if the requested content is available in the cache. By reducing the load on the backend systems, it contributes to reducing stress on those components, again yielding better availability.

Web Cache will periodically ping backend servers. With this feature, Web Cache avoids routing the request to a backend server that is down or cannot provide the service. In the case of “sticky” session routing, this failure cannot be masked, so it is recommended that stateless routing be used at the Web Cache level, since OHS can better handle routing around failures with sticky sessions.

Changes in configuration are propagated across a Web Cache cluster in a rolling fashion – since not all changes take effect right away; you can selectively start the Web Cache instances with the new configuration.

### **Oracle HTTP Server (OHS)**

In configurations without Web Cache, the Oracle HTTP Server (OHS) is the entry point within Oracle Application Server for incoming requests. OHS is highly available because each OHS child process is independent and thus its failure has no major system wide impact. The in-flight requests do fail and some connection pooling, if used, has to be setup again.

Any application deployment or configuration change does require OHS to be restarted. However, it supports graceful restart – it lets each child process finish the in-flight requests and then starts only that child process (or thread) with the new configuration. With this approach, the HTTP Service itself is never down.

### **Mod\_oc4j**

Mod-oc4j is responsible for routing to OC4J instances. It is always aware of which OC4J processes are alive and thus avoids routing to those that are down.

This module leverages HTTP cookies to keep track of a session's association with *islands*. Within an island, OC4J processes replicate their session state automatically. If an OC4J process goes down between requests, this module is able to hide that failure from the end user by transparently routing the next request to another, available OC4J within the same island. This behavior is not possible with either an external load balancer or Web Cache.

In case of stateless requests or requests that are not being served by an OC4J island, there is not as significant an advantage from leveraging mod-oc4j routing over Web Cache or an external load balancing mechanism.

### **Mod\_plsql**

Mod\_plsql availability is tightly tied to: (a) the OHS process availability, which we have discussed above, and (b) the database connection availability, since mod\_plsql caches the connection for optimal performance.

If a connection is dropped [say, because the database was on a Cold Failover Cluster, and it failed over], mod\_plsql will need to flush its connection cache and re-try the connections. This action may cause a prolonged window where failures are intermittently visible to end-users<sup>1</sup>. A graceful restart of OHS can reduce this window.

---

<sup>1</sup> A connection failure is recognized as part of a request processing failure. Once the failure is recognized, mod\_plsql drops that connection and re-establishes the connection cache. On UNIX platforms, due to the process-oriented architecture, this connection pool is specific to a process and thus each child process has to first receive a failure. To avoid this, a graceful restart of OHS is recommended. The thread-oriented architecture on Windows avoids this problem.

### **OC4J – Deployment**

Deploying new applications on a “live” OC4J instance should in general have no availability impact on other applications on the same instance. This feature is also referred to as “hot deployment”<sup>2</sup>. However, there may be some performance impact during the deployment, since additional one-time activities are performed during the deployment process.

Re-deploying an existing application is different. The existing sessions may not be carried forward post-deployment – depending on the changes to the session object.

Deployment failures can sometimes leave OC4J in a state such that it cannot start; however, quick recovery mechanisms to a stable state via DCM archival/restore make it easy to ensure downtime window stays small.

### **OC4J – HTTP Session State**

In the case of stateless requests, better availability is a simple case of having more OC4J processes on separate nodes. In the case where a session is established and state information is required with each return request, the three choices available are: (a) save the state in the database, (b) replicate the state by defining *islands*, and (c) leverage Java Object Cache to distribute the session object across OC4J processes.

Each option has implications: State replication does not have guarantee semantics, but Java Object Cache does. And neither provides the transactional semantics that database persistence can. Neither database saved state, nor the Java Object Cache allow the simpler declarative model that session replication does. The choice really depends on the cost and tradeoffs that a system designer is willing to undertake.

### **OC4J – EJBs**

Mod\_oc4j routes requests to Servlets and JSPs. EJB Proxies perform the same function for EJBs. EJB Proxy classes can discover EJB servers in one of two ways: (a) dynamically, by asking a known EJB server for the list of all servers that can provide it the specific service, and (b) by using a list of existing servers configured on the EJB client side.

The good thing about the dynamic discovery is that new server members could be added to the EJB Cluster. The downside is that the single entry point – the server used to discover other servers – might impact the discovery process if it is down.

The members of EJB cluster behave similar to OC4J islands in that they replicate EJB state. The replication is of value only in the case of stateful session beans. In this case, the point at which replication happens is: (a) JVM termination – i.e. right before the JVM crashes or terminates, (b) end of call – in that case the state is

---

<sup>2</sup> Strictly speaking, “hot deployment” only has one requirement: the system should not be restarted during deployment of an application. This can be achieved by carefully planning your deployment. For more details see Oracle Application Server Documentation.

replicated at the end of each EJB remote method.

#### **OC4J – TAF**

Transparent Application Failover (TAF) is an important High Availability feature of the JDBC Thick driver, also referred to as the Oracle Call Interface (OCI) based driver. When a J2EE application connects to a backend customer RAC database using the Thick driver, the driver can be configured to support TAF.

TAF supports two modes: Basic and Pre-connect. In the Basic mode, when a connection to an instance of the RAC database is lost, the JDBC driver attempts to establish a connection to another, available instance of the database to continue processing. In the Pre-connect mode, every JDBC connection is shadowed by another connection to a different instance of the RAC database. Upon failure of the primary connection, the driver shifts processing to the backup connection.

For read-only requests to the database, TAF is truly transparent. For inserts, updates or deletes, failure of the JDBC connection in the middle of a request causes the database to rollback the partially completed request. In order to correctly recover and proceed, the application developer needs to provide a “cleanup” method using the `oracle.jdbc.OracleOCIFailover` interface. This method allows the application to make the failover transparent to the end user.

#### **Portal, Reports, Forms and other Middle Tiers**

Portal uses (a) Web Cache (b) `mod_plsql` (c) EAR files deployed on OC4J (Parallel Page Engine) – a stateless application, (d) stored procedures in the Portal metadata repository, (e) Portal metadata store and (f) IM Service.

Making Portal highly available requires each one of the components to be made highly available. The previous sections already covered Web Cache, `mod_plsql`, and OC4J. Each one of these components can be configured on multiple machines or multiple processes on same machine for higher availability.

The Portal metadata can be made highly available by using the two approaches defined in the Infrastructure section. In addition, this metadata can be separated from the infrastructure and reside in a customer RAC database.

The IM Service can be made available with the options discussed earlier, independent of what approaches are used to make other Portal components highly available.

The Portal system availability does not necessarily imply the availability of Portlets. The individual Portlets may be dependent on other OC4J instances or on external systems.

Reports Services in Oracle Application Server can be deployed in multiple ways – as an *in-process server*, or, as an *out-of-process server*. Both leverage a separate *reports engine* process to actually run the report. The reports engine is stateless and multiple instances are spawned on demand.

Since the Reports Server processes maintain the end-user session, their availability is key. The Reports system also requires the IM service, so IM service availability impacts the Reports system as well.

If an in-flight request fails, the failure is visible to the browser. A browser refresh re-issues the request and can be routed to a Reports server or a Reports engine that is up and running or, one is started specifically for that request. Thus, the failure is easily recoverable.

The initial request from a browser for a Forms application always goes to the *Forms Servlet*, which starts a Java applet on the browser. This applet establishes communication (and a session) with the *Forms Listener Servlet*. The listener servlet, in turn, starts a *Forms Runtime* process – one for each distinct session. It is this Forms Runtime process that actually runs the Form: executing PL/SQL and triggers, running the logic, connecting to the database, etc.

The Listener Servlet maintains communication state between the client and the Forms Runtime process. The actual user interface state is in the Forms Runtime process.

Due to this architecture a Listener servlet going down causes wider failure, but a Forms Runtime failure is localized to a single user. The Forms Listener Servlet can start and connect only to a Forms Runtime on the same machine. These two processes can be duplicated on multiple nodes so a node failure does not impact all the users – only a portion of the users.

#### **Enterprise Manager and Distributed Configuration Manager**

Oracle Enterprise Manager (EM), also called Grid Control, manages the Oracle Application Server grid environment. Grid Control is the primary interface for performing all management tasks in a grid. A reliable and scalable multi-administrator repository offers the option of cooperative management. Distributed Configuration Manager (DCM) provides a generic mechanism for managing heterogeneous configuration data. EM and DCM together provide:

- Unified architecture for managing the complete Oracle environment
- Centralized console for single point of management
- Real-time monitoring
- Oracle Application Server Grid Management

EM and DCM make Oracle Application Server system management easier, helping reduce configuration errors – a significant cause of system downtime.

Syntax and semantic checks at several places help avoid erroneous configurations.

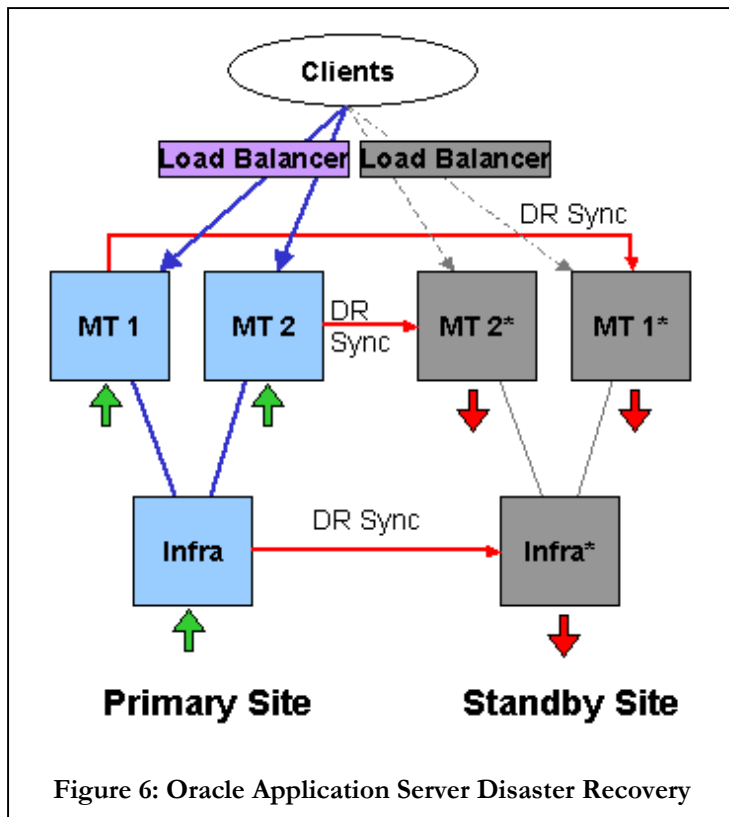
Clusters and the associated configuration replication further reduce the management workload. In addition, the new archival and retrieval capability allows for simpler mechanism for configuration cloning.

DCM Archival and Recovery is a new feature that allows the Application Server Administrator to revert back to a previous configuration should there be an error in modifying the configuration. This method is far quicker than restoring configuration from a backup and can be performed online.

### Disaster Recovery

As mentioned earlier, any system, no matter how well designed, will eventually experience failure – possibly from external factors. In order to cope with such failure, several high availability configurations have been discussed in the preceding sections. However, when disaster, such as total power failure or earthquake strikes and disables the entire data center, most HA solutions discussed so far will be unable to prevent downtime.

In order to cope with such situations, Oracle Application Server provides the Oracle Application Server Disaster Recovery solution: the ability to quickly bring up services at a geographically distant “standby” location in case of total data center loss at the primary site.



In 9.0.4 Disaster Recovery uses OracleAS Backup & Recovery for restoring configuration files of the Infrastructure and Middle Tiers and Oracle Data Guard technology for restoring the Infrastructure database at the standby site.

Each node on the primary site has a corresponding standby node on the standby site. The Disaster Recovery scripts are run by the Administrator regularly to ensure that the primary and the standby sites remain synchronized (see Figure 6). The standby site is inactive during normal operations. When the primary site goes down, the administrator activates the standby site and then modifies the DNS to ensure that traffic is routed to the standby site. Once the switch is completed, the former standby site can process all incoming requests. The entire operation is designed to minimize downtime.

## 4. SUMMARY

Oracle Application Server 10g (9.0.4) provides significant new high availability and Disaster Recovery capabilities, which can handle scenarios ranging from instance level failure recovery to recovering from complete site failures. The Infrastructure tier supports Cold Failover Clusters, Active Failover Clusters, RAC install of metadata repository, as well as use of IM replication to make the IM service more available. The middle tier supports the standard multi-machine deployment with load balancer in front and provides additional session replication, clustering, and smart routing capabilities.

Since most of these features are implemented without a need to re-program or re-architect an application, Oracle Application Server delivers improved availability for all deployed applications. For further details about high availability configurations, the reader is encouraged to refer to the High Availability Guide of the Oracle Application Server Documentation [5].

## 5. REFERENCES

1. *Oracle Application Server 10g (9.0.4) Concepts Guide*. Standard Oracle Application Server Documentation
2. *High Availability Application Design: Key Strategies*, Randy Heffner, February 18, 2003 Giga Information Group Inc.
3. *Blueprints for High Availability: Designing Resilient Distributed Systems*, Evan Marcus and Hal Stern, Wiley, 1999.
4. *Oracle Application Server Upgrading to 10g (9.0.4)*. Standard Oracle Application Server Documentation
5. *Oracle Application Server High Availability Guide*. Standard Oracle Application Server Documentation



Oracle Application Server 10g High Availability

January 2004

Authors: Ashesh Parekh, Mukul Paithane

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

[www.oracle.com](http://www.oracle.com)

Oracle is a registered trademark of Oracle Corporation. Various product and service names referenced herein may be trademarks of Oracle Corporation. All other product and service names mentioned may be trademarks of their respective owners.

Copyright © 2003 Oracle Corporation

All rights reserved.