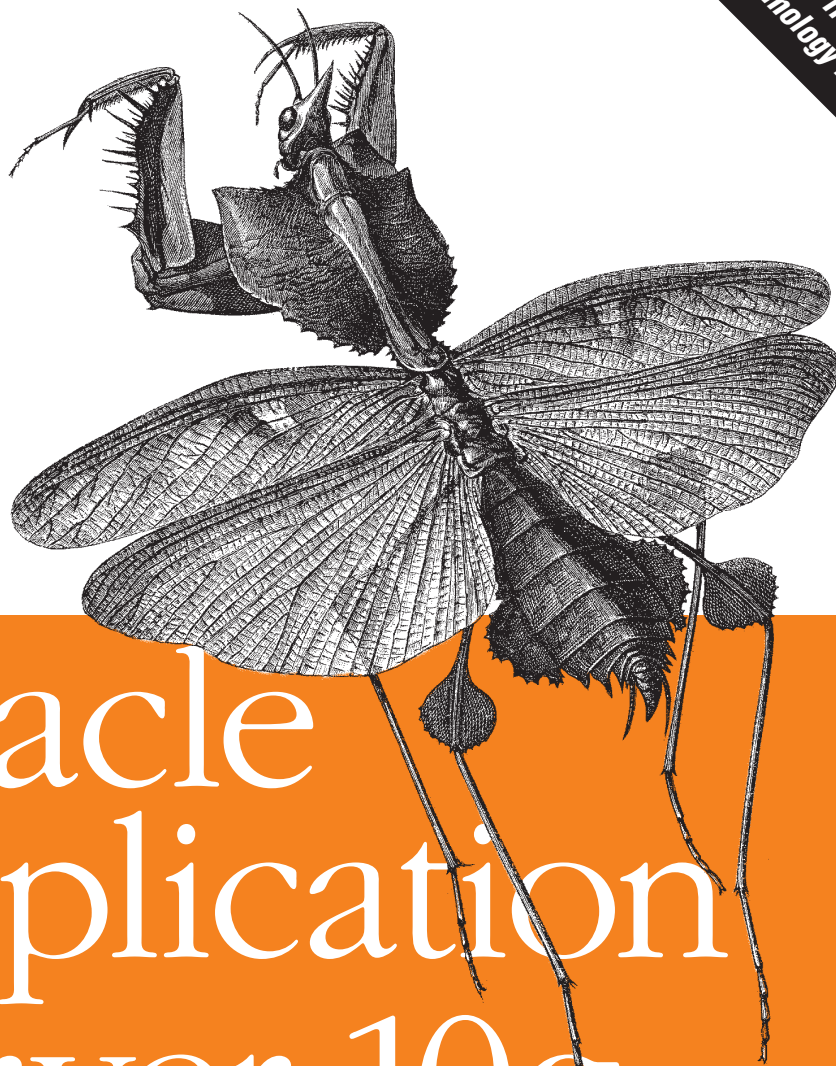


Architecture & Components

Download
product for free from
Oracle Technology Network



Oracle Application Server 10g

Essentials

O'REILLY®

Rick Greenwald, Robert Stackowiak & Donald Bales

Architecture

If you understand the architecture of Oracle Application Server, you will have an excellent framework for understanding how the product works. Learning how the various components of Oracle Application Server interact can help you avoid many potential problems as you get the product up and running and provide a foundation that lets you quickly identify the root cause of any problems you do encounter.

Your particular role may determine what you want to learn from this chapter:

Programmer

As a programmer, you may want to understand how Oracle Application Server handles each incoming request so that you can properly troubleshoot problems.

Designer

As a designer, you will want to know what features are available so that you can provide the best user interface.

Systems analyst or architect

As a systems analyst or architect, you may want to know what overall functions can be accomplished using the product.

Manager

As a manager, you may need a high-level understanding of Oracle Application Server so that you can compare it to similar middleware product offerings.

To best serve all these audiences, this chapter covers the big picture. We identify the major components of Oracle Application Server and explain how they interact. We also examine the product's infrastructure and the management and high-availability options available with it. Our goal here is to give you enough information on each individual component (most of which were introduced very briefly in Chapter 1) so that you can understand how Oracle Application Server works as a whole. You will see that some of Oracle Application Server's functionality is actually implemented by J2EE, by PL/SQL, or by hybrid applications that are implemented using Oracle Application Server's core components. Subsequent chapters provide more detailed discussions on each component introduced here.

Oracle Application Server Core Components

At its core, Oracle Application Server consists of three components:

- Oracle HTTP Server
- Oracle Application Server Containers for J2EE
- OracleAS Web Cache

Each component, described briefly in the following subsections, plays a major role in Oracle Application Server operations; they also provide some supporting functionality to their counterparts in various configurations.

Oracle HTTP Server

The Oracle HTTP Server is the web server for Oracle Application Server. It takes an incoming request in the form of a URI. It then either sends the requested static content—for example, an HTML file—directly to the requestor, or reroutes a request for dynamic content, effectively handing off the request, to an appropriate executable resource.

The Oracle HTTP Server can process a dynamic request using any of the following:

Common Gateway Interface (CGI) environment

For example, programs written in C, C++, Java, Perl, and other languages

FastCGI

An optimized CGI environment

`mod_perl`

A highly efficient Perl execution environment

`mod_OC4J`

A scalable J2EE environment

`mod_plsql`

A module that executes PL/SQL stored procedures in an Oracle database

Chapter 5 describes the Oracle HTTP Server in detail.

Oracle Application Server Containers for J2EE

OC4J is a set of J2EE containers and a JavaServer Pages translator that provides a J2EE-certified Java environment that is scalable both horizontally and vertically. You can cluster OC4J instances across hosts; doing so eliminates both hardware and software failure and provides almost unlimited capacity. You can run multiple JVMs in an OC4J instance to leverage the capacity of hardware with multiple CPUs.

Chapter 6 describes OC4J in detail.

OracleAS Web Cache

OracleAS Web Cache is an optional component that can play several important roles:

- It can offload request processing from the Oracle HTTP Server and OC4J by efficiently caching both static and dynamic content. OracleAS Web Cache is highly configurable and programmable, so what it caches can be customized to fit any application's needs.
- It can be used as a front end load balancer to the application server when Oracle Application Server is clustered.



Because using OracleAS Web Cache so often makes sense in the Oracle Application Server environment, the remainder of this chapter assumes its use.

Chapter 7 describes OracleAS Web Cache, as well as the other caches used by Oracle Application Server, in detail.

Core Component Interaction

Figure 2-1 shows the relationships among the three main components of Oracle Application Server.

As shown in the figure, the interactions among components work as follows:

1. OracleAS Web Cache receives an end-user request and determines whether the requested content is available in its cache. If the content is available, it returns that content. Otherwise, it forwards the request to the Oracle HTTP Server.
2. The Oracle HTTP Server immediately sends static content to the end user through OracleAS Web Cache. If the request is for dynamic content, it forwards the request to the appropriate dynamic content provider.
3. If Java-generated content is requested, OC4J processes the request using a JSP or servlet, and then sends the dynamically generated content to the Oracle HTTP Server. The Oracle HTTP Server then passes it on to OracleAS Web Cache.
4. Finally, OracleAS Web Cache passes the content on to the end user.

As the figure illustrates, there are additional dynamic processing engines, such as FastCGI, `mod_perl`, and PL/SQL. The architecture also includes a number of shadow processes, shown in gray, that support the use of Oracle Application Server core components. These shadow processes include:

Oracle Process Manager and Notification Server (OPMN)

Monitors the core components to ensure that they continue running

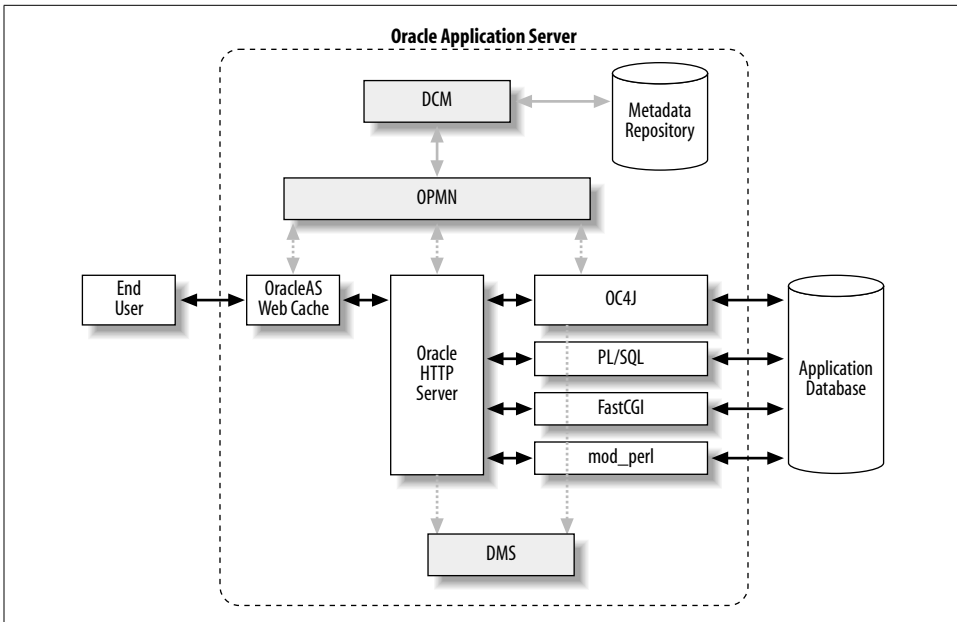


Figure 2-1. Interactions among Oracle Application Server core components

Distributed Configuration Management (DCM)

Marshals Oracle Application Server’s configuration data between OPMN and the core components

Dynamic Monitoring Service (DMS)

Gathers performance statistics

See the later “Shadow Processes” section for more detailed information on these processes.

The configuration data itself is kept in a repository, which is part of the Oracle Application Server infrastructure, described in the next section.

Oracle Application Server Infrastructure

Shadow processes such as OPMN, DCM, and DMS, and configuration management tools such as Oracle Enterprise Manager, depend on a repository of configuration data. This repository is part of what is known as the *Oracle Application Server Infrastructure*. The OracleAS Infrastructure may consist of a simple file-based repository, or may be as complex as a database repository with an LDAP server.

OracleAS Infrastructure Components

The OracleAS Infrastructure is divided into two parts:

OracleAS Metadata Repository

Holds Oracle Application Server configuration information and other component information in a collection of schemas

Oracle Identity Management

Contains several subcomponents that work together to provide a seamless Single Sign-on facility

In addition, the Application Server Control tool and a group of shadow processes can be considered a part of the overall OracleAS Infrastructure.

OracleAS Metadata Repository

The OracleAS Metadata Repository is either operating system file-based or an Oracle database. The Java Edition of Oracle Application Server uses a file-based repository for DCM. The Portal/Wireless Edition and the BI/Forms Edition always use an OracleAS Metadata Repository in an Oracle database.

If Oracle Application Server is clustered (a topic described later in this chapter) and a file-based repository is being used, one of the clustered Oracle Application Server instances is designated as a *central repository*. This designation isn't necessary if a database repository is being used.

The OracleAS Infrastructure installation normally clones a new Oracle database to use as a database repository. During the installation process, it is possible to use an existing database instance for the OracleAS Metadata Repository, instead of creating a new one. However, specifying an existing instance has an impact on the use of the OracleAS Backup and Recovery Tool, described in Chapter 3.

Oracle Identity Management

Oracle Identity Management is a term used with Oracle Application Server 10g that encompasses all the components used to support enterprise-style security in Oracle Application Server. All these components depend on the OracleAS Metadata Repository:

Oracle Internet Directory

Provides an LDAP directory for security principals (login information)

Oracle Directory Provisioning Integration Service

Allows the LDAP directory to import and replicate data between homogeneous or heterogeneous types of LDAP servers

Oracle Delegated Administration Services

Allows regional administrators and users to update directory information in the Oracle Internet Directory

OracleAS Single Sign-On

Supports the Single Sign-on capability for all Oracle Application Server applications

OracleAS Certificate Authority

Provides an infrastructure for creating and maintaining SSL certificates

Chapter 4 describes Oracle Application Server security in greater detail.

Application Server Control

The web-based Oracle Enterprise Manager 10g tool used to monitor and control Oracle Application Server is known as *Application Server Control*. Application Server Control is actually an Oracle Application Server application that is installed with the OracleAS Infrastructure and with each individual application server home. The tool has a graphical user interface and is used to manage individual instances of Oracle Application Server. Other portions of the overall Oracle Enterprise Manager 10g product handle management of the database and a grid of computing resources.

Oracle Enterprise Manager stores component configuration data in the Metadata Repository. Doing so simplifies the management for multiple Oracle Application Server instances when Oracle Application Server clusters are being used.

Chapter 3 describes Oracle Enterprise Manager 10g, Application Server Control, and Grid Control, as well as other Oracle Application Server management tools, in more detail.

Shadow Processes

Oracle Application Server has many different components, and some deployment architectures may have more than one instance of any one of these components. To keep the burden of monitoring, managing, and coordinating the interaction between these components from being too overwhelming, Oracle Application Server includes three shadow processes, shown earlier in Figure 2-1:

- Oracle Process Manager and Notification Server
- Distributed Configuration Management
- Dynamic Monitoring Service

These shadow processes also use the OracleAS Metadata Repository. DCM updates component configuration files so that they will match the values in the repository. OPMN uses the repository to identify what components should be started and monitored. DMS stores runtime statistics in the repository.

Oracle Process Manager and Notification Server

The Oracle Process Management and Notification Server monitors the health of the individual components in an Oracle Application Server architecture. OPMN monitors and manages all Oracle Application Server components, with the exception of the following:

- The repository database, if used
- The listener for the database
- The Application Server Console

If a particular component becomes unavailable, OPMN automatically restarts it. OPMN also informs any dependent component that a component isn't available. For example, suppose that a request for a Java application comes into an Oracle HTTP Server. It is passed to the `mod_oc4j` module, which passes the request to an OC4J process. If the OC4J process goes down, OPMN lets the `mod_oc4j` process know that it should prevent any additional requests from being routed to the failed process. In this way, OPMN simplifies management while making the entire set of Oracle Application Server components highly available.

Distributed Configuration Management

For high availability and scalability, you can create *clusters* of Oracle Application Server components (Oracle HTTP Server and OC4J) and OracleAS Web Cache. The Distributed Configuration Management service coordinates configuration information across all members of a cluster.

DCM automatically replicates base configuration information to a new member of a cluster. It also propagates changes to any of this information to all members of a cluster. In addition to this base information, each cluster member also has some individual configuration parameters of its own, such as port numbers for the instance.

You can also use DCM to make a backup of your configuration files that can be used to restore these crucial files in the event of corruption.

DCM is invoked through the Application Server Control tool or via the `dcmtcl` command-line tool.

Dynamic Monitoring Service

Oracle Application Server uses the Dynamic Monitoring Service to collect information about the performance of some of its components. DMS uses sensors to measure various types of durations or simply the occurrence of different types of events. DMS periodically collects the information from these sensors and compiles it into metrics. You can access these metrics using a variety of different tools. The most commonly used is `dmstool`, a command-line tool for presenting DMS metrics. The metrics may also be accessed in different portions of the Application Server Control tool.

DMS sensors are built into Oracle Application Server Containers for J2EE, so metrics are automatically provided for all applications that run in an OC4J container. You can add sensors to any application you deploy on Oracle Application Server through a DMS API, so the use of DMS is completely extensible. You can also turn off DMS to save the small amount of overhead incurred.

Installation Types

The complexity of the Oracle Application Server infrastructure depends on the type of product installation. There are three Oracle Application Server installation types:

- J2EE and Web Cache
- Portal and Wireless
- Business Intelligence and Forms

Each installation type builds on its predecessor's functionality; consequently, each requires a more complex infrastructure.

J2EE and Web Cache

The J2EE and Web Cache installation provides a high-performing and very scalable J2EE application server. In the Enterprise Edition version of the Oracle Application Server product, it consists of the following:

- Oracle HTTP Server
- Oracle Application Server Containers for J2EE
- OracleAS Web Cache

Note, however, that OracleAS Web Cache isn't included in the Java and Standard Editions.

This installation requires a DCM repository in which Oracle Application Server configuration data can be stored. The repository for this version can be either file-based or stored in an Oracle database.

Portal and Wireless

The Portal and Wireless installation contains all the components of the J2EE and OracleAS Web Cache installation. It also contains:

- OracleAS Portal, a complete, out-of-the-box, user-configurable portal solution
- OracleAS Wireless, which allows you to use web-based applications on mobile devices

Although this edition doesn't contain the following components, it uses them at runtime, so the infrastructure has to exist before you install this edition:

Identity Management

This infrastructure must exist in the form of Oracle Application Server Single Sign-On product support; this product allows your users to use one set of login credentials for all applications running on the application server.

Oracle Internet Directory

This infrastructure is added to the Identity Management infrastructure because Oracle Application Server Single Sign-On requires an LDAP server.

Note that OracleAS Portal and OracleAS Wireless store their own application data in the infrastructure's OracleAS Metadata Repository.

Business Intelligence and Forms

The Business Intelligence and Forms installation contains all the components available in the OracleAS Portal and OracleAS Wireless installation, as well as the following additional components:

- OracleAS Discoverer
- OracleAS Forms Services
- OracleAS Reports Services
- OracleAS Personalization

This installation doesn't add any more complexity to the infrastructure because no additional infrastructure components are required, but it does add more metadata. Like the Portal and Wireless edition, the Business Intelligence and Forms edition uses a preexisting infrastructure. However, there is also a standalone installation for the Forms Server and Reports Server that doesn't require an infrastructure.

Scalability Architectures

Oracle Application Server is highly scalable as a J2EE application server. As demand for a particular J2EE web application grows, so too can the application server's capacity grow, both vertically and horizontally. The following subsections describe the various types of scaling supported by the product.

Vertical Scaling with OC4J

Oracle Application Server can scale its capacity vertically by running more than one instance of OC4J on a given host, or by running more than one Java Virtual Machine in a given instance of OC4J.

Multiple OC4J instances per host

An Oracle Application Server instance can be configured to run one or more instances of OC4J. In such a configuration, each OC4J instance runs with its own

JVM. However, running multiple instances of OC4J on the same host is more of a configuration convenience than it is a means of scalability because multiple OC4J instances on a single machine share the same configuration files.

It isn't uncommon for several applications to share an application server. By running multiple OC4J instances, each heterogeneous application can run in its own OC4J instance. This reduces the risk that the deployment of an application may affect other applications (or appear that it has). It also makes it possible to do resource consumption accounting by application, which is a necessary evil in many business environments.

Multiple JVM processes per OC4J instance

Vertical scaling is better achieved by running more than one JVM process in an OC4J instance. Doing so leverages the processing power of a host that has multiple CPUs.

When more than one JVM process is running, HTTP session objects and EJB states are replicated among the JVMs. In addition to the server's offering increased capacity to process incoming requests in such a configuration, the server is also better equipped to recover from JVM process failures. In case of such a failure, it can redirect a request to one of the other JVMs running in the same OC4J instance.

Replicating HTTP session objects can become a resource-intensive task. To better manage this state replication, Oracle Application Server provides a mechanism that provides a finer level of control. Each JVM is assigned to an island. An *island* logically groups JVMs together for the purpose of HTTP session object replication. If two or more JVMs have the same island name (even in clustered Oracle Application Server instances on different hosts), HTTP session objects are replicated among them. This capability means that you can control which JVMs replicate state with each other.

For example, if you have an OC4J instance that has four JVM processes running, you can have two JVMs assigned to `default_island`, and the other two assigned to `island_two`. With this configuration, HTTP session objects are replicated between the two JVMs that are part of the same island. This reduces the overhead of replicating each HTTP session object because each JVM has only one replication site, rather than three.

In contrast, no refined replication control exists for EJB state. EJB state replication doesn't use the island replication facility. The state of any stateful EJB is replicated among all JVMs in an OC4J instance (or in clustered Oracle Application Server instances).

Horizontal Scaling with Oracle Application Server

Horizontal scaling is accomplished using clusters. A *cluster* groups two or more Oracle Application Server instances so that they appear as one application server.

Clustered Oracle Application Server instances utilize only the Oracle HTTP Server and OC4J components.

Figure 2-2 shows two Oracle Application Server instances in a cluster. In this figure, a load balancer receives a user request. The load balancer forwards the request to an Oracle Application Server instance, where it is processed. Looking at the figure, you can see that both instances of Oracle Application Server use the same metadata repository, as well as any other resource, such as an application database.

Oracle Application Server farm

To be clustered, Oracle Application Server instances must be part of the same farm. A *farm* is a group of one or more Oracle Application Server instances (clustered or not) that share the same OracleAS Infrastructure or that use the same Oracle Application Server instance for their file-based OracleAS Metadata Repository.

Oracle Application Server instances automatically become part of a farm during installation. The farm assignment is based on which OracleAS Infrastructure or Oracle Application Server instance (for a file-based repository) they use for their OracleAS Metadata Repository. This assignment can be changed at a later date using Oracle Enterprise Manager.

Oracle Application Server cluster definition

An Oracle Application Server cluster definition is created within a farm using Oracle Enterprise Manager. After a cluster is defined, the first Oracle Application Server instance added to the cluster defines the cluster-wide configuration. Thereafter, each server that is added to the cluster gets its configuration and any deployed applications from the cluster-wide configuration. The only configuration properties that aren't part of the cluster-wide configuration are the OC4J-specific properties: the number of JVM processes per OC4J instance and the ports an OC4J instance uses to communicate with the Oracle HTTP Server.

Cluster management

Clustered Oracle Application Server instances appear as one application server not only to the end user but also to the Oracle Application Server administrator. If changes are made to any Oracle Application Server instance in a cluster, the DCM process automatically replicates the changes to all the other instances in the cluster. This process significantly reduces the amount of work an administrator has to do to deploy applications in a clustered environment.

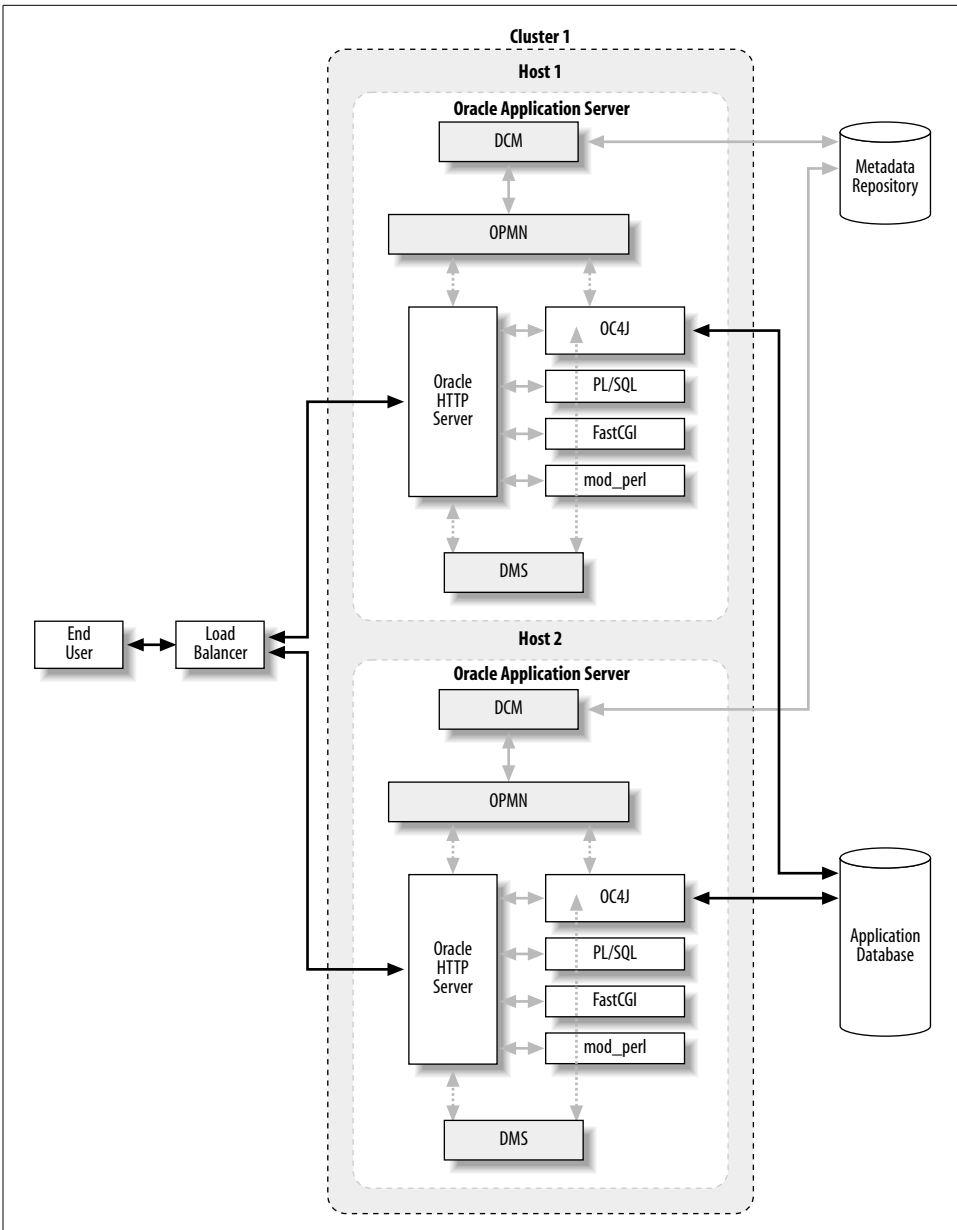


Figure 2-2. Clustered Oracle Application Server instances

Load Balancing

As mentioned earlier, a load balancer is required when Oracle Application Server instances are being clustered together. There are three traditional choices for load balancing:

- Hardware
- Operating system
- OracleAS Web Cache

Of the three, Oracle recommends using a hardware-based load balancer. The following subsections describe these three approaches, along with a brief discussion of load balancing with the Oracle HTTP Server.

Hardware load balancer

A hardware load balancer works by receiving a request from an end user and then forwarding the request to a clustered Oracle Application Server instance. Hardware load balancers support various forwarding algorithms, including the following:

Fastest Response

Forwards a request to the application server that responds the most quickly

Round Robin

Forwards a request to the next application server in a predetermined list of servers

Least Connections

Forwards a request to the application server with the fewest number of connections

Calculated Ratio

Forwards a request to the application server with the best calculated ratio based on predetermined statistics

Geographic Location

Forwards a request to the application server located closest to the requestor

Hardware load balancers are preferred because they are faster and have redundant components that eliminate downtime caused by mechanical or electrical failure. Figure 2-3 shows a typical hardware load balancer configuration. In this figure, an end user sends a request to a hardware load balancer. The hardware load balancer forwards the request to a clustered instance of OracleAS Web Cache. OracleAS Web Cache returns cached content, if available. If cached content isn't available, it forwards the request on to a clustered instance of Oracle Application Server.

Examples of hardware load balancers include the following products:

F5 Networks' BIG-IP

See <http://www.f5.com/f5products/bigip/>

Nortel Network's Alteon

See <http://www.nortelnetworks.com/products/01/alteon/>

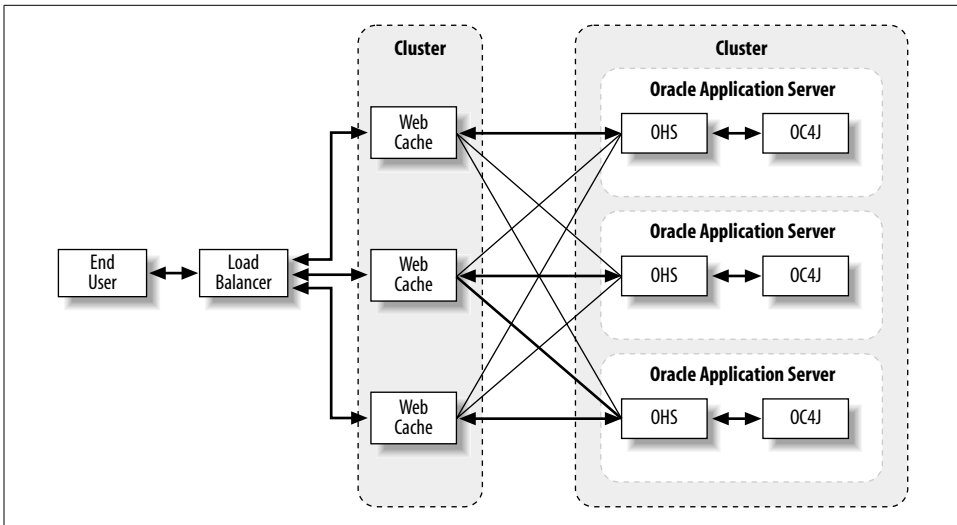


Figure 2-3. A hardware load balancer example

Operating system load balancer

Some operating systems provide a built-in load-balancing mechanism. In particular, Microsoft Windows Advanced Server allows you to forward requests to different machines that share the same IP or MAC level address. In addition, load-balancing software applications are available for almost all operating systems.

OracleAS Web Cache load balancer

OracleAS Web Cache can be used as a load balancer for clustered Oracle Application Server instances. When OracleAS Web Cache acts as a load balancer, it uses a Weighted Available Capacity algorithm to determine the particular Oracle Application Server instance to which it will forward the request. This algorithm uses a weight, assigned by an administrator, to distribute the load unequally among instances. If the weighted available capacity is the same, OracleAS Web Cache uses a Round Robin algorithm.

If an Oracle Application Server instance fails, OracleAS Web Cache redistributes requests to the remaining instances. Meanwhile, OracleAS Web Cache intermittently polls the failed server until it is once again available to process requests. At that time, the failed server is included in the load-balancing algorithm.

For more information about this capability of OracleAS Web Cache, see the detailed discussion in Chapter 7.

Oracle HTTP Server load balancer

While our discussion so far has focused on load-balancing requests to clustered Oracle Application Server instances, another type of load balancing may also be taking place. Oracle HTTP Server's `mod_oc4j` load-balances requests among multiple JVM processes in an OC4J instance or among OC4J instances in a cluster.

If an OC4J instance is configured to use two or more JVM processes or is part of a cluster, `mod_oc4j` routes a stateful request to the JVM that last processed the stateful request. If the JVM that last processed a stateful request has failed, the request is rerouted using a predetermined algorithm to another OC4J instance's JVM process that is part of the same island as the failed JVM.

Three algorithms are available for rerouting a request to a failed OC4J instance:

Round Robin

As usual with a Round Robin algorithm, `mod_oc4j` uses the next OC4J instance (and the next JVM process in the same island), remote and local to the Oracle Application Server instance, on a predetermined list.

Random

With the Random algorithm, an OC4J instance is picked randomly from the predetermined list.

Metric-Based

This algorithm uses OC4J performance metrics to determine where to route a request.

All these algorithms can be configured with local affinity, so a local OC4J process is used in favor of a remote process. Similarly, the Round Robin and Random algorithms can employ a weighting factor to determine which OC4J process is chosen.

Stateless requests are always processed using one of these predetermined algorithms. You can configure the desired algorithm, along with all the other settings, using Oracle Enterprise Manager.

High Availability

The availability of an application server can be measured by comparing the actual amount of time it is operational against the total time it could possibly be available. Highly available systems may need to operate 100% of the time. To accomplish that level of availability, you must address every possible single point of failure. In addition to hardware and software failures, you must have a way to continue operations while both the hardware and software are being upgraded or replaced.

Oracle Application Server solves these high-availability problems with clusters and with two different failover solutions for the OracleAS Infrastructure.

Oracle Application Server Clusters

We introduced the concept of clusters earlier in this chapter. Although both a farm and a cluster are made up of multiple instances of Oracle Application Server components, a cluster is a more integrated grouping than a farm.

In a cluster, the individual instances of a component are aware of each other, and they cooperate to provide service. The OPMN server keeps the members of a cluster informed as to the status of other members in that cluster, so if one member fails, the others pick up its load. The members of a cluster are designed to be interchangeable, providing a higher level of availability than an individual instance or a group of instances in a farm.

You can create clusters of OracleAS Web Cache instances or Oracle Application Servers. As mentioned earlier, a cluster requires some type of load balancer in front of the cluster to distribute the load. The redundant nature of an individual instance within a cluster makes clustering a way to provide a high-availability solution.

Oracle Application Server can also run on hardware clusters. These clusters provide an underlying redundancy and transparency for anything running on them, including Oracle Application Server.

Infrastructure Failover

Oracle Application Server clusters deliver high availability with their own software. However, these clusters don't provide high availability for the OracleAS Infrastructure. Because all Oracle Application Server components need to access the infrastructure and because the availability of an entire system is dependent on every part of the system, this deficiency can imperil the availability of your application server.

To address this risk, you can implement Oracle Application Server 10g with either of two types of failover solutions:

Oracle Application Server Cold Failover Clusters

This approach uses hardware clustering to provide some measure of availability protection.

Oracle Application Server Active Failover Clusters

This approach provides a higher level of availability and combines it with increased scalability for your infrastructure.

OracleAS Cold Failover Clusters

The OPMN server manages most Oracle Application Server components and services, and this server can automatically restart any failed processes. This capability protects against the failure of individual components of Oracle Application Server. However, larger problems, such as server failure, are beyond the scope of OPMN.

Thus, Oracle Application Server provides an availability solution, OracleAS Cold Failover Clusters, that guards against these larger-scale failures.

OracleAS Cold Failover Clusters are built on capabilities inherent in a hardware cluster. A hardware cluster facilitates the sharing of resources between nodes, provides a health-monitoring heartbeat, and supports failover from one node to another without impacting users. Hardware clusters provide a “virtual IP” address, which it can associate with either node in the cluster. Users don’t know which physical node is servicing requests for the virtual IP address.

If you are familiar with database availability concepts, you will recognize the term *cold failover*. With Oracle Application Server, the term has the same meaning it does for a database: one identical set of resources stands by, waiting for a failure in the primary system. If that system fails, the cluster starts to use the standby system. Figure 2-4 shows the architecture of a typical OracleAS Cold Failover Clusters implementation before a failure has occurred.

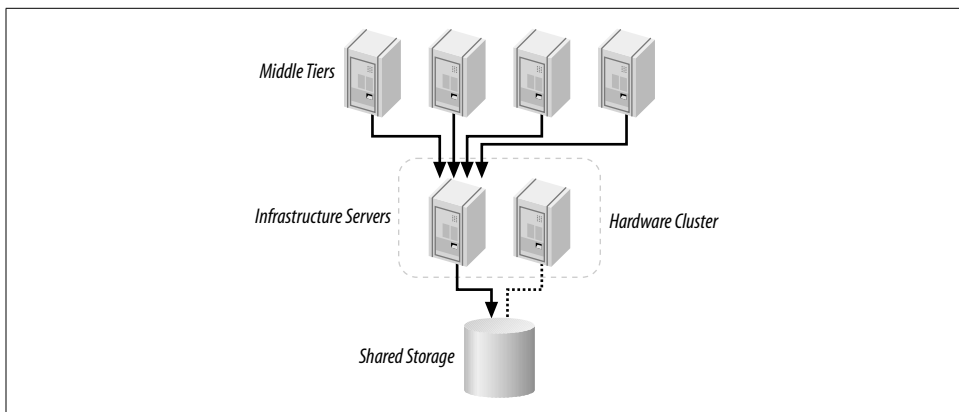


Figure 2-4. OracleAS Cold Failover Clusters prefailure

Because many Oracle Application Server components depend on the infrastructure for continuous operation, OracleAS Cold Failover Clusters are designed to protect the components of the OracleAS Infrastructure. If an active node in an OracleAS Cold Failover Cluster configuration fails, the IP address is switched to the failover node, and the infrastructure processes are started on that node. Once these processes are available, middle-tier components can access the new infrastructure, as shown in Figure 2-5. Because the storage for both infrastructure nodes is shared, there is no need to recreate or migrate the information in the database.

You can have instances of other Oracle Application Server and OracleAS Web Cache coexisting on either node in the OracleAS Cold Failover Cluster. However, no failover for these instances is supported. You can group the instances in their own cluster, but the operation of these clusters is separate from the operation of the OracleAS Cold Failover Cluster.

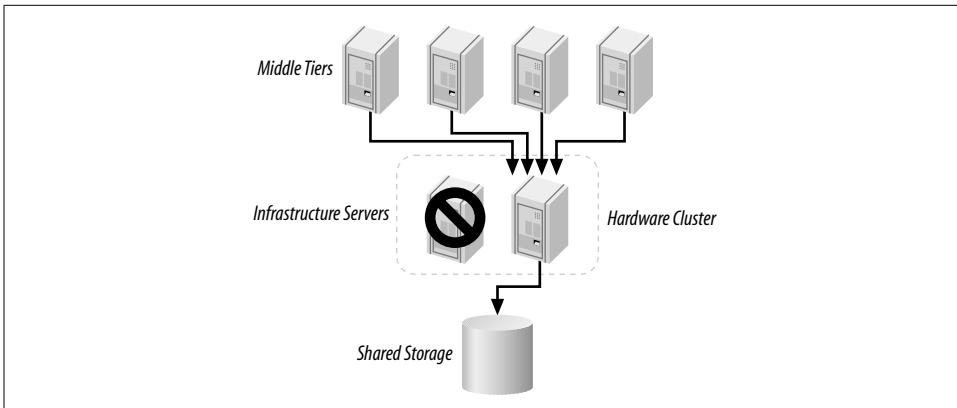


Figure 2-5. OracleAS Cold Failover Clusters post-failure

OracleAS Active Failover Clusters

Oracle Application Server 10g, in its initial release, provides limited (i.e., beta-style) support for another type of failover solution: Active Failover Clusters. OracleAS Active Failover Clusters uses the same architecture as OracleAS Cold Failover Clusters.



At the time of this writing, OracleAS Active Failover Clusters is still in limited release. By the time you read this book, OracleAS Active Failover Clusters is likely to be available for a number of platforms for Oracle Application Server. For up-to-date information on certified platforms, please see http://otn.oracle.com/products/ias/hi_av/10g_904_HA_Certification.html.

The big difference between OracleAS Active Failover Clusters and OracleAS Cold Failover Clusters is that with OracleAS Active Failover Clusters, you don't have to have one node in the hardware cluster sitting around waiting for a failure. Both infrastructure nodes can service infrastructure requests. Because both nodes in an OracleAS Active Failover Cluster are servicing requests, OracleAS Active Failover Clusters can provide some scalability, in addition to the availability offered by both types of failover clusters.

The architecture for an OracleAS Active Failover Cluster configuration is shown in Figure 2-6.

The major differences between this configuration and a OracleAS Cold Failover Cluster configuration are the following:

- An OracleAS Active Failover Cluster configuration uses a hardware load balancer in front of the active infrastructure nodes.
- The infrastructure uses Real Application Clusters (RAC) for its database.

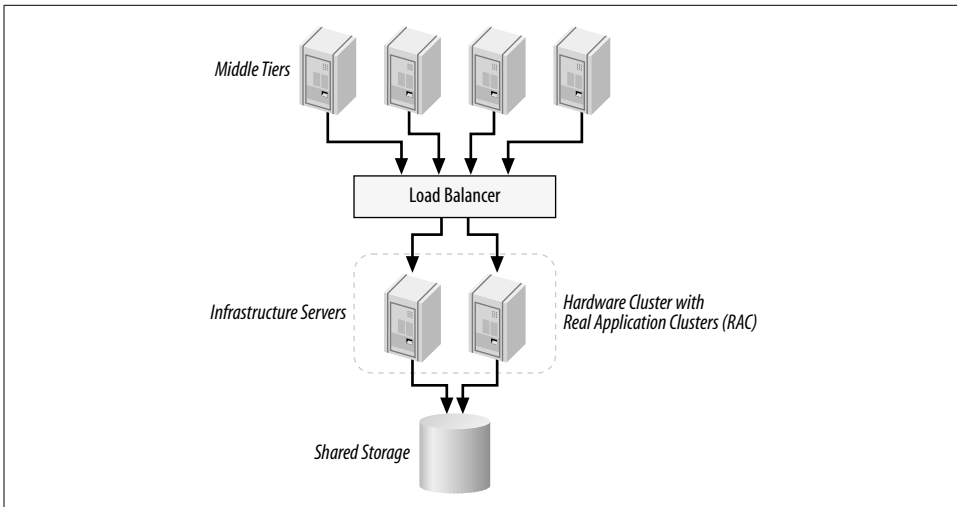


Figure 2-6. OracleAS Active Failover Clusters architecture

Both nodes have the same infrastructure components running at the same time, and the load is split between the infrastructure nodes. If one node fails, the failover is much faster and more transparent than it is for OracleAS Cold Failover Clusters.

OracleAS Active Failover Clusters use a RAC database to provide scalability and availability for the OracleAS Metadata Repository. RAC is Oracle's clustered database solution. It runs on a hardware cluster and allows any application to run against a clustered database without modification. For more information on RAC, please refer to the discussion in our companion database book *Oracle Essentials* (O'Reilly).

In addition to using RAC for the OracleAS Metadata Repository, OracleAS Active Failover Clusters have duplicate infrastructure processes, such as those used for identity management, running on each node in the cluster. All nodes in an OracleAS Active Failover Cluster must have virtually identical configurations, down to the pathnames for Oracle *HOME* directories.

Oracle Application Server Applications

The following sections provide brief descriptions of the applications that are part of the architecture of Oracle Application Server. Each application, or set of applications, is covered in its own chapter; the following sections focus on how they fit into the overall architecture of Oracle Application Server.

OracleAS Portal

OracleAS Portal performs its basic construction of portal pages in the Portal Page Engine, which is a J2EE application. OracleAS Portal also makes extensive use of the

PL/SQL language in the construction of its pages. OracleAS Portal is closely integrated with the OracleAS Single Sign-on capability provided by the Oracle Identity Management infrastructure, so an OracleAS Portal installation requires an infrastructure.

Chapter 13 describes OracleAS Portal in detail.

OracleAS Wireless

OracleAS Wireless is designed to enable wireless devices such as cell phones, messaging devices/pagers, and wireless Internet-connected laptops to connect to an application that provides content. The Oracle Application Server architecture includes a device/network gateway provider, an XML application framework, and an adapter that enables communication to and from the application leveraging a multichannel server.

Chapter 14 describes OracleAS Wireless and its various mobile components in detail.

Business Intelligence

The primary business intelligence (BI) components available in Oracle Application Server are OracleAS Discoverer and OracleAS Reports Services. Oracle Application Server also includes an OracleAS Personalization component that leverages data mining features to present web-site pages based on previously analyzed tendencies of visitors.

OracleAS Discoverer tools enabled for Internet access include Discoverer Plus and Discoverer Viewer:

Discoverer Plus

This tool is a full-featured ad hoc query and analysis tool. It is initiated using a downloaded Java applet. After the initial applet download, the applet resides in local cache.

Discoverer Viewer

This tool can view intelligence query results and charts generated by Discoverer Plus in a browser. It is servlet-based and leverages Discoverer Services.

Both tools take advantage of the fact that Oracle Application Server contains a data cache, which enables rapid data manipulation without the need to requery the back-end database. The database tier of the architecture contains Discoverer workbooks, the End User Layer, and, of course, the data.

The OracleAS Reports Services component lets you create high-quality printed and web-based reports. OracleAS Reports Services uses Java servlets (known as *Java servers*) to receive requests and then direct the requests to a runtime engine that provides the requested functionality. You can configure the number of runtime engines for OracleAS Reports Servers initially started, the maximum number of engines that

can be running at one time, and the idle time before an engine is shut down. Multiple OracleAS Reports Servers can provide load balancing, although not failover.

Chapter 9 describes OracleAS Reports Services in detail, and Chapter 12 describes OracleAS Discoverer and other business intelligence components.

OracleAS Forms Services

OracleAS Forms Services can build applications that provide an interactive graphical interface for data entry with support for data validation. As with OracleAS Reports Services, OracleAS Forms Services use Java servlets to receive requests and direct them to a runtime engine. Each user has a matching runtime process, which is also a Java servlet. As with OracleAS Reports Servers, you can have multiple OracleAS Forms Servers that can provide load balancing, although not failover.

Chapter 9 describes OracleAS Forms Services in detail.