Oracle Application Server 10g
Release 3 High Availability

An Oracle White Paper
February 2006
1. - EXECUTIVE OVERVIEW ......................................................... 3
2. - HIGH AVAILABILITY INTRODUCTION ...................................... 4
3. - UNPLANNED DOWNTIME PROTECTION ................................. 5
   3.1. - Oracle Application Server 10g Release 3 Clusters ............... 5
      Oracle Application Server Clusters (OC4J) and session replication. 6
      Oracle Application Server Clusters and JNDI Replication .......... 8
      Oracle Application Server Clusters and JMS High Availability .... 8
   3.2. - Death-Detection and auto-restarts .................................... 10
   3.3 - Load Balancing, intelligent routing and Dynamic Resource
      Management ........................................................................... 11
   3.4. - Backup and restore: Oracle Application Server Recovery
      Manager .................................................................................... 13
   3.5 - Disaster Recovery Solution: Oracle Application Server Guard.. 14
4. - PLANNED DOWNTIME PROTECTION .................................... 15
   4.1. - OracleAS Hot deployment ................................................. 15
   4.2. - OracleAS Rolling Upgrades .............................................. 16
   4.3. - OracleAS Dynamic Clusters (topology extensions and
      modifications) ....................................................................... 18
5. – BEST HIGH AVAILABILITY WITH ORACLE DATABASE ... 19
SUMMARY.- ..................................................................................... 20
1. - EXECUTIVE OVERVIEW

Service-oriented architecture has quickly become the key design principle upon which organizations are building and integrating modern applications. In this type of architecture, multiple systems cooperate and interact to provide a higher-level functionality. This intrinsic “pluggable” and “reusable” nature makes it more critical than ever to confer the required protection mechanisms to the systems providing the aggregated services. The outage of a single component, can affect multiple running operations.

Oracle Application Server 10g Release 3 (10.1.3), the new release of Oracle Application Server 10g and a component of Oracle Fusion Middleware, is designed to provide a standards-based, mission critical platform for organizations planning their futures around such reliable architectures. Oracle Application Server 10g Release 3 has extended the High Availability features of previous releases to provide the most flexible, resilient and fault-tolerant application server platform for the Grid, making it possible to deploy SOA systems with maximum availability and guarantee of service.

Additionally, and as a key value for all those customers using the leading database server product in the market, Oracle Application Server 10g Release 3 is highly integrated with the latest high availability features included in Oracle Database 10g and provides the most advanced mechanisms for load balancing and failover between the middle tier and the database tier of an application.

This document, describes the high availability features of Oracle Application Server Release 3 10g (10.1.3). 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.
2. - HIGH AVAILABILITY INTRODUCTION

An application server platform has to provide the features and components required to guaranty that all the types of **planned and unplanned** outages have a minimum impact on the availability of a system.

Most application server platforms in the market only provide some protection mechanisms for system failures, leaving out to third party solutions how to protect systems from data failures and disasters. The challenge for an Application Server is, in reality, providing a highly available Grid infrastructure that protects all components and applications from all the possible causes of downtime.

**Unplanned outages** are caused by Process failures, Data failures, Hardware failures, Human errors or Disasters (such as floods, earthquakes or regional network outages) that can affect an entire datacenter. Oracle Application Server Release 3 10g (10.1.3) provides different features and components to protect against all of those.

**Planned outages** are caused by changes that are applied to the running system and that have an operational impact in the application (deployment of new versions of an application, upgrades and modifications of the infrastructure being used by the application etc). Oracle Application Server Release 3 10g (10.1.3) offers multiple new and exciting features designed to minimize the impact of configuration changes, re-deployments and transformations in a system.
The following sections in this document describe in more detail each one of the features above

3. - UNPLANNED DOWNTIME PROTECTION

3.1.- Oracle Application Server 10g Release 3 Clusters

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. From the system’s entry point (load balancers or content cache) to the back end layer (data sources) all the tiers that are in the request-reply path can be configured in a redundant manner with Oracle Application Server. The main layers that are crossed are: load balancers and/or content cache, HTTP servers, J2EE containers and data sources. All these components usually invoke system services such as LDAP servers and single sign on servers. Oracle Application Server Release 3 allows choosing between active-active or active-passive redundant models in all its sub-tiers.

In an active-passive configuration the passive components are only used when the active component fails. Active-passive solutions deploy an active instance that handles requests and a passive instance that is on standby. In addition, a heartbeat mechanism is usually set up between these two instances together with a hardware cluster (such as Sun Cluster, Veritas, RedHat Cluster Manager, Oracle CRS…) agent so that when the active instance fails, the agent shuts down the active instance completely, brings up the passive instance, and resumes application services.

In an active-active model all equivalent members are active and none are on standby. All instances handle requests concurrently.

1 Oracle Application Server 10g R3 can use Oracle Identity management 10g R2 (10.1.2) and Oracle Identity Management 10g R2 (10.1.4). The high availability properties of these components are covered separately in the specific collateral for those releases.
Obviously, an active-active system generally provides higher transparency to consumers and has a greater scalability than an active-passive system. On the other hand, the operational and licensing costs of an active-passive model are lower than that of an active-active deployment.

The installation flexibility provided by Oracle Application Server 10g Release 3 allows both types of redundancy and adapts to different business requirements. For those systems requiring an active-active model, Oracle Application Server 10g Release 3 provides **Oracle Application Server Clusters**, a set of application server instances configured in an active-active model to serve the same set of applications and/or services. When an active-passive model is needed, Oracle Application Server 10g Release 3 provides **Oracle Application Server Cold Failover Cluster**, which is a set of application server instances (2 in most cases, since only one remains active and no greater benefits are achieved by including more nodes) configured in an active-passive model to serve the same set of applications and/or services. Oracle Application Server installer provide the choice of deploying either model “out of the box.”

<table>
<thead>
<tr>
<th>Feature</th>
<th>Oracle Application Server Clusters</th>
<th>Oracle Application Server Cold Failover Clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>☺</td>
<td>☺</td>
</tr>
<tr>
<td>Operational Cost</td>
<td>☺</td>
<td>☺</td>
</tr>
<tr>
<td>Licensing costs</td>
<td>☺</td>
<td>☺</td>
</tr>
<tr>
<td>High Availability</td>
<td>☺</td>
<td>☺</td>
</tr>
<tr>
<td>Installation</td>
<td>☺</td>
<td>☺</td>
</tr>
</tbody>
</table>

**Oracle Application Server Clusters (OC4J) and session replication**

Oracle Application Server 10g Release 3 allows the grouping of J2EE containers that serve the same application through Oracle Application Server Cluster (Oracle Containers for J2EE). In OracleAS 10g Release 3 (10.1.3), a cluster topology is defined as two or more loosely connected Oracle Application Server nodes.

The connectivity provided within a cluster is a function of Oracle Notification Server (ONS), which manages communications between Oracle Application Server components, including Oracle Containers for J2EE (OC4J) and Oracle HTTP Server. Creating a cluster in OracleAS 10g Release 3 (10.1.3) is as simple as grouping OracleAS instances inside the same ONS topology.

For stateless applications, the deployment of the same application in different J2EE containers that receive requests from Oracle HTTP Server suffices to provide High Availability. For stateful J2EE applications it is necessary that the client application state be adequately recreated whenever the J2EE processes are started or the
application’s control is transferred to a redundant process. In general, this operation requires that individual processes (and applications) persist their state and read and recreate that state on startup.

Two different types of state are typically maintained in a J2EE application: HttpSession state (updated by servlets and jsp) and Stateful Session Beans state (updated by Stateful Session Beans instances). Oracle Application Server 10g Release 3 uses the same cluster framework for both. Also, in Oracle Application Server 10g Release 3 the Oracle Containers for J2EE’s clustering framework has been redesigned to provide greater flexibility and extensibility. This new model uses reliable unicast and multicast message transmission to retransmit those state messages that are lost, so that delivery of state to participants in the same replication group cluster is guaranteed. The new Oracle Containers for J2EE clustering framework allows a fast recover from crashes since each node keeps track of the location of its replicas.

In Oracle Application Server 10g Release 3 replication of state for J2EE applications can be enabled both at application and container level thus providing a more granular control of the replication scope. The new replication model allows choosing the number of nodes to replicate to and uses fragmentation for large messages, thus providing a more scalable and performant framework.

Different mechanisms can be used to trigger the replication of sessions across an OracleAS Cluster. This is done using different “replication policies”. A replication policy defines when replication of HttpSession or Stateful Session Bean state occurs, and whether all attributes/variable values or changed values only are replicated. Replication can be an expensive process, and replicating data too frequently can affect server performance. On the other hand, replicating data too infrequently can result in lost data in the event of server failure. This provides a great flexibility to adapt a cluster configuration to different protection needs.

Additionally, the session replication framework of Oracle Application Server Clusters is based on a pluggable framework that guarantees services such as reliability, ordering, fragmentation and group membership but allowing also customized extensions.

Finally and for those cases where the state of an application is critical enough as to justify its preservation from the downtime of a complete application server environment (all nodes in a cluster), Oracle Application Server Cluster (OC4J) allows the automatically controlled persistence of a the session (either HttpSession or Stateful Session Bean state) in a database. This feature enables that session information is available across instance restarts even with only one application server instance. Session persistence to the database can be done synchronously or asynchronously. In asynchronous persistence mode OC4J continues its operation without waiting for the session to be persisted, thus offering a good trade off in performance and reliability. The synchronous option makes sending the response dependent on the completion of the persistence operation.
This comes at the cost of reduced perceived performance but with maximum protection for the session information.

**Oracle Application Server Clusters and JNDI Replication**

OracleAS 10g (10.1.3) provides protection for JNDI contexts in an OracleAS Cluster (OC4J) environment with JNDI state replication. JNDI state replication causes changes made to the context on one OC4J instance of an OC4J cluster to be replicated to the name space of every other OC4J instance in the cluster. When JNDI state replication is enabled, the objects that are bound into an application context (using a remote client, EJB, or servlet) on one server can be read on another server. JNDI state replication is enabled on a per-application basis. JNDI replication is automatically configured when HttpSession or Stateful Session Bean replication is enabled and can be specified at application or container level. The protocols and policies available for JNDI replication are the same as the ones available for HttpSession or Stateful Session Beans, with this OracleAS 10g Release 3 (10.1.3) provides a unique cluster framework for multiple services simplifying its configuration and management.

**Oracle Application Server Clusters and JMS High Availability**

Oracle Application Server 10g Release 3 provides different options for the high availability of asynchronous systems based on Java Message Service. Oracle Application Server 10g Release 3 provides its JMS implementation through Oracle Enterprise Messaging Service (OEMS) JMS. OEMS JMS offers a single JMS interface with three message persistence options: In-Memory, File-Based, or Oracle Database. OEMS JMS In-memory maintains independent topics and queues in the process memory of each OC4J. OEMS JMS File-Based persists topics and queues to a specific file for each OC4J instance. Finally, OEMS JMS Oracle Database persists topics and queues to an Oracle Database (shareable by multiple OC4J processes). Each one of the persistence options has different High Availability characteristics.

1.- High Availability options for OEMS JMS In-memory and File-Based. These are the different ways that OEMS JMS In-memory and File-Based can be deployed to provide high availability:

- **Dedicated JMS Server**: In this configuration, a single OC4J instance is dedicated as the JMS server. All other OC4J instances that are hosting JMS clients forward their JMS requests to the dedicated JMS server. This OC4J instance handles all messages, thus message ordering is always maintained. There is no possible JMS failover to other instances and high availability is achieved only at process restart level (this is automatically provided by OPMN, see section 3.2). If an OEMS JMS In-memory process dies unexpectedly OPMN will try to restart it (see section 3.2), service will be resumed but in-memory messages will be lost. If an OEMS JMS File-based process dies unexpectedly OPMN will try...
to restart it (see section 3.2) service will be resumed and committed messages will be available again. JMS clients need to reconnect after a failure with a delay that will vary depending on the time it take to restart the OEMS JMS (OC4J start).

- **Distributed Destinations:** In this configuration, all factories and destinations are defined on all OC4J instances. Each OC4J instance has a separate copy of each destination. Since OEMS JMS does not provide queues and topics replication across OC4J instances, failover to a different OEMS JMS running in a separate instance is not allowed. In other words, a message enqueued to the JMS Server in one OC4J process can be dequeued only from that OC4J process. The same rules apply regarding service restoration as for Dedicated JMS Server. This option can provide better performance than Dedicated JMS Server, but does not provide any improvements regarding availability.

- **Cold Failover Cluster (only with OEMS JMS File-based):** This is an active-passive configuration (see introduction to section 3.1) based on a two-node hardware cluster. Only one OEMS JMS node is active at any time. The second node is made active if the first node fails. In case of failover, the shared storage holding the file where queues and topics are persisted is mounted (automatically by clusterware in most cases) on the second node so that service is resumed with the same queue and topic information that was available before the failure. Since this is an active-passive solution with an implicit downtime in the failover phase, JMS clients need to reconnect after a delay that will vary depending on the hardware cluster failover period.

2.- In none of the above configuration for In-memory and File-based topologies, messages, queues/topics are replicated across multiple OEMS JMS instances. It needs to be understood that, for OEMS JMS In-memory and File-based only one instance is “the owner” and process manager of the messages queued and dequeued in it. When maximum protection and failover functionality are needed, the best option is to use **OEMS JMS Oracle Database with RAC and Transparent Application Failover (TAF).** In this case if a failure happens in an OEMS JMS instance, all committed messages will be available in other OC4Js (load balancing and failover of JMS requests is allowed). If a failure happens in any of the database instances providing the AQ service, automatic failover happens and the OEMS JMS instance continues its operations without noticeable failures. The load balancing and failover of requests can be provided either through load-balanced connection factory lookups (in the client code that makes OEMS JMS invocations) or through an external load balancer (hardware or software).
3.2. - Death-Detection and auto-restarts

Processes may die unexpectedly due to configuration or software problems. To maximize the availability of a system and reduce the impact of this type of system failures, all Oracle Application Server components are monitored by Oracle Process Manager and Notification Server (OPMN) – and are automatically 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 for routing requests between HTTP servers and J2EE containers) 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.

Oracle Application Server 10g Release 3 introduces new exciting features for process control and notifications across instances. Oracle Notification Service (ONS) has been enhanced to support the specification of either a discover multicast address or, alternatively, a set of gateway nodes that bridge the communications between different application server instances. This enables the automatic discovery of new processes that participate in a notification group and allows the dynamic addition of redundant nodes to a topology.

Additionally, new service failover functionality has been added to OPMN allowing the creation of singleton services that are monitored by OPMN. Any process managed by OPMN that is configured with the service-failover attribute, will contain a single process-set, which will represent a single process in an entire Oracle Application Server Topology on any one instance with the same configuration. If a service-failover process is configured with the restart-on-death attribute set to true, then the instance running the service process will attempt to restart the process once if it terminates unexpectedly (crashes), and if this attempt fails, the instance will lower its service-weight to 1 and the process will fail-over to another participating instance (if there are no other available instances, then the current instance will continue to retry starting the process). If the service-failover process is configured with the restart-on-death attribute set to false, then the process will immediately fail over to another participating instance, however if there are no other available instances, the current instance will attempt to continue to try starting the process.

These capabilities extend the death-detection and auto-restart properties present in previous releases of Oracle Application Server and make Oracle Process Manager and Notification Server (OPMN) the most advanced monitoring and self healing mechanism in the application server market.
3.3 - Load Balancing, intelligent routing and Dynamic Resource Management

When multiple instances of identical server components are available, client requests to these components need to be load balanced to ensure that the instances have roughly the same workload. If any of the instances fail, requests to the failed instance can be sent to the surviving instances. Load Balancing mechanisms are used to communicate every tier in an Oracle Application Server system. Load balancing takes place:

- From Web Cache to Oracle HTTP servers
- From Oracle HTTP servers to OC4J components for Java Applications
- From Oracle HTTP servers to different database instances components for mod_plsql applications
- Intra OC4J components (from the presentation layer components to the business components layer)
- From OC4J components to databases
- From OC4J components and single sign on services to Oracle Internet Directory (LDAP server)

The best way to maintain a system available is to adjust it and tune it so that the load is equally balanced between the available resources, thus the importance of an efficient and smart routing mechanism. In many cases, overloading of the resources being used by a running application causes system failures. When the load is not distributed based on the resources available in each target node, a failure in one of the nodes may result in a cascade fall of the entire system. Overloading is usually application’s code responsibility. This is why it is especially relevant that requests are correctly routed to the highest resource-consumers sub-tiers in an application: the business logic layer and the database. Oracle Application Server provides smart routing between the HTTP Servers and Oracle Containers for J2EE and between the J2EE containers (and in general any database client) and the database tier. This is why Oracle Application Server 10g Release 3 is enabled to provide the best crash prevention mechanisms in the market and the most resilient and reliable J2EE deployments. This type of dynamic workload management is a key enabler of Grid Computing.

The following table presents the different type of load balancing mechanisms that are present in an Oracle Application Server 10g Release 3 deployment.
As a key feature in Oracle Application Server 10g Release 3 (10.1.3), the routing relationship between Oracle HTTP Server and OC4J has been enhanced to increase the availability of Oracle Application Server. In Oracle Application Server 10g Release 3 all HTTP servers listen for notifications from OC4Js. These OC4J processes notify that they have a routing relationship with the HTTP Servers. Therefore, each of these OHSs discovers the OC4Js it needs to route to. The OC4J components announce their mount-point(s) (the URL mapping to be routed to each J2EE container) in the notifications they send out and OHS dynamically adjusts its routing table using this information without requiring any restarts. This makes it possible to define routing relationships between HTTP servers and J2EE containers totally dynamically (Dynamic Routing) and provides a very flexible and reliable model for cluster topologies.

Additionally and to improve even further the crash prevention mechanisms in previous releases, Oracle Application Server 10g Release 3 provides a Dynamic Resource Manager that allows to control process based on system conditions and according to a set of user-configured directives. The DRM (Dynamic Resource Manager) is one of the key high-level services added to Oracle Application Server to provide a full Enterprise Grid Computing solution. With DRM, new OC4J’s can be stopped, started and controlled based on the status of the system. The system conditions that can be taken into account include any DMS (Dynamic Monitoring Service) metrics that are available within the application server. This includes preset metrics that are always present as well as metrics that are defined and implemented by the user in their own applications. DRM enables to take actions based on the system’s load, thus providing dynamic control over the balancing of requests and enabling sophisticated crash prevention mechanism to be applied to different applications. DRM provides a proactive management of a cluster and can be used to guaranty the Service Levels required in a system.
3.4. – Backup and restore: Oracle Application Server Recovery Manager

Backup and restore refers to the various strategies and procedures involved in guarding against hardware failures and data loss, and reconstructing data should loss occur. Oracle Application Server Recovery Manager makes it easy to create this checkpoint and then restore it if necessary. Oppositely to other application server platforms, Oracle Application Server 10g Release 3 (10.1.3), takes care, with one single tool, of the complete environment where the application runs. Oracle Application Server Recovery Manager (formerly known as Oracle Application Server Backup and Restore), takes care of:

Oracle software files: These are static files such as binaries and libraries. They reside in the middle-tier and Infrastructure Oracle homes. They are created at installation time.

Oracle Application Server Configuration files: These files contain configuration information and deployed applications. They reside in the middle-tier and Infrastructure Oracle homes. They are created at installation time and are updated during the normal operation of your application server. Configuration files managed by DCM are not included in this group.

Oracle system files: These files may be in other directory than the Oracle Home. They usually reside outside of the Oracle Application Server installations.

Any custom files: any external files can also be added to the configuration to be backed up and recovered by Oracle Application Server Recovery Manager.

The key feature of Oracle Application Server 10g Release 3 Recovery Manager is that the entire environment around an application is considered when backing up and restoring. In very few occasions, an application is deployed on a J2EE container on its own and without using typical pieces such as an LDAP server, an HTTP server or other services such as a single sign on. Oracle Application Server Recovery Manager protects such complex environments by:

1.- Guarantying that the recovery will be consistent and easily integrated.

2.- Taking care of possible mismatches between the different configuration sources.
3.- Preventing administration changes from being run while a backup is taking place.

Instead of leaving the work to different pieces of software and different procedures, Oracle Application Server Recovery Manager allows a complete Application Server Platform Suite backup with a single click from Oracle Application Server Control Management console.

With OracleAS Recovery Manager incremental backups, online backups and configuration-only backups can be performed in a single step also guaranteeing full recovery with optimized RPO and RTO\(^2\) of an application server instance.

### 3.5 - Disaster Recovery Solution: Oracle Application Server Guard

Disaster recovery solutions typically set up two mirror sites, one active and one passive. Each site is a self-contained system. The active site is generally called the production site, and the passive site is called the standby site. During normal operation, the production site services requests. When the primary site goes down, the administrator activates the standby site and then modifies the DNS settings 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. To maintain the standby site for failover, not only must the standby site contain homogeneous installations and applications, data and configurations must also be synchronized constantly from the production site to the standby site. Third-party remote mirroring solutions can be used for this synchronization. These types of systems rely on third-party remote mirroring products to perform the instantiation, synchronization, failover and switchover tasks between sites. However, physical images (bit-by-bit copies) created with these products may often be inconsistent (for example: if they are taken while some configuration changes are being applied to the application environment, since mirroring products are not aware of the logic operations performed in an Application Platform Suite) and require an expensive infrastructure (network and disk requirements mainly) to work properly.

Oracle Application Server Guard uses OracleAS Recovery Manager for restoring configuration files of the Infrastructure and Middle Tiers and Oracle Data Guard technology for restoring the Infrastructure database at the standby site, thus guaranteeing a totally consistent and reliable copy of the production system. Oracle Application Server R2 (10.1.3) has automated most of the tasks required to create and maintain a standby site thus tremendously decreasing the total cost of ownership of a disaster recovery solution.

The tool performs the following operations:

- **Clones/creates the standby site**: copies the OracleAS binaries and configuration files, thus eliminating the task of having to install these Oracle instances on the standby topology.

---

\(^2\) RPO (Recovery Point Objective) = length of service downtime during recovery

\(^2\) RTO (Recovery Time Objective) = amount of data loss after recovery
- **Instantiates standby site**: instantiates an Application Server standby topology that mirrors a primary topology
- **Verifies configuration**: Verifies that a topology meets the requirements to be used as a standby topology for a given primary topology
- **Synchronizes**: Synchronizes the production and the standby
- **Allows to perform switchover and failover operations making the current standby site the new production system in one single command**
- **Allows the creation of asymmetric topologies**, where the standby site does not need to have the same number of Application Middle Tier installs. OracleAS Guard can synchronize just a subset of the production site in the standby site, this allowing the use of “scaled down” standby sites for cost reduction purposes

As the only disaster recovery solution integrated in an Application Platform Suite, Oracle Application Server Guard 10g R2 (10.1.3) presents a highly available, low cost, and administrator-friendly alternative to multi-vendor application server disaster recovery solutions.

4. - PLANNED DOWNTIME PROTECTION

4.1. - OracleAS Hot deployment

“Hot deployment” of an application refers to the deployment of a new application without affecting an application server’s availability (including other applications running in the same application server instance). OracleAS 10g Release 3 (10.1.3) implements significant changes in the way that applications are made available for
incoming requests. OracleAS 10g (10.1.3) introduces new features that nullify (there is only a performance impact) the impact of the deployment of an application in the system’s availability. In OracleAS 10g (10.1.3) the J2EE containers (OC4J) announce their readiness to received requests in reverse notifications to Oracle HTTP Server (this is known as “OracleAS Dynamic Routing”). This eliminates the need for static lists of OC4J instances to route requests to (also known as “mount points”) and enables the HTTP Server to update its routing configuration without a restart. In Oracle Application Server, applications running in the same system do not experience any downtime during the deployment of other applications, since no component needs to be restarted during the deployment. This is possible because OracleAS 10g (10.1.3) increases the granularity in which processes and services are controled. Oracle Process Manager and Notification (OPMN) now monitors “applications” (as opposed to monitor “containers”) thus notifying Oracle HTTP Server of the availability of a service at a much finer level. This makes posible to start and stop an application service without any impact in the other applications running in the same container.

4.2. - OracleAS Rolling Upgrades
A “rolling upgrade” is an upgrade of a group of software components, performed without a noticeable downtime or other disruption of the common service that they provide. All systems face a need for component replacement and upgrades from time to time. The need to facilitate software upgrades is demanding because a system with continuous service uptime expectation cannot be just stopped for maintenance and upgrade. In order to provide service continuity, the software upgrades have to be performed on a running system in such a way that the availability requirements are met. There are several parts in an application environment that need to be considered for rolling upgrades:

- Application version upgrade (re-deployment of an application)
- Application Server version upgrade
- Database version upgrade
- OS version upgrade

The last two are out of the scope of this paper.

**Application version upgrade:** OracleAS Dynamic Routing (see section 4.1) warranties that stateless and stateful J2EE applications running in a OracleAS Cluster can be “upgraded” and re-deployed without any impact in the systems availability. For stateless applications, the fact that Oracle HTTP Server detects the unavailability of an application during the re-deployment process, makes it possible to continue the service transparently in another instance until the deployment process in the first one has finished. Once this happens, the deployment occurs in the “now servicing requests” instance and it is the old one which assumes the responsibility of continuing the service. For stateful applications, Oracle Application Server Cluster framework allows the coexistence of different
application versions that maintain the same HttpSession\(^3\) and Stateful Session Bean Session state, thus enabling that new versions of an application are deployed without losing the user’s session. The deployment utilities included with OracleAS 10g (10.1.3) allow performing a sequential deployment to multiple instances in a cluster in one single command. This enables state to be automatically preserved across the deployment process (This is known as the stateful rolling upgrade of an applications across an OracleAS 10g Release 3 (10.1.3) Cluster)

Additionally, Oracle Application Server Release 3 (10.1.3) introduces a new deployment model (accordingly to the latest J2EE platform specification (JSR-88)) that results in faster updates to applications. For instance, with the new incremental ejb deployment model, applications re-use prior deployments for unchanged beans when updating a single EJB module in a running application.

**Application Server Version Patching/Upgrade:** Application Server’s rolling upgrades involve updating the application server with patches or even with a complete new version of software without affecting a system’s availability. Two separate layers need to be considered in an upgrade procedure of the application server: the upgrade of the database used to store security, metadata and ldap information, and the upgrade of the runtime binaries and configuration files used by the application server.

For the metadata, security and single sign on data stored in Oracle Database, Oracle Application Server supports rolling upgrades of the database through Real Application Clusters. The Oracle Database supports the application of patches to the nodes of a Real Application Clusters (RAC) system in a rolling fashion. For the specific case of Oracle Application Server Identity Management environments (when the single sign on and ldap services are deployed separately from the rest of the Oracle Application Server components)

Oracle Application Server supports Multi Master Replication for rolling upgrades of the Metadata Repository when this is only used to store Identity management Information (e.g. when no other component metadata, such as OracleAS Portal’s, or OracleAS Wireless’ is stored in that database repository). With this mechanism, a replica database node is taken off from the replication group for upgrade/maintenance while other instance serves business application requests. After the upgrade/maintenance work is completed, the offline node is put back online to serve application requests. It also retrieves changes from the available node while it was offline. Similarly other nodes can be upgraded or patched by repeating above procedure. Multimaster Replication allows the installation of database software upgrades, and the application of patch sets to the different databases used by Identity Management without any downtime implied in the process.

---

\(^3\) The maintenance of session information across deployments may drive to some inconsistencies depending on the logic included in the latest version of the application: it is totally up to the application’s code that the session information is treated in a different/inconsistent way from the previous deployment.
For the rest of the software binaries and configuration files that are not part of the
database (e.g. HTTP Server, J2EE container, deployed applications…) the
minimum impact of an application server upgrade is guaranteed through Oracle
Application Server Clusters. With Oracle Application Server Clusters a no-request-
loss upgrade can be achieved by progressive install/patch of the new application
server instances deployed while the rest of the participants in the cluster still deliver
requests. OracleAS 10g (10.1.3) Containers for J2EE are compatible with
OracleAS 10g (10.1.2) HTTP Server, thus allowing a gradual change from a
previous release of OC4J to the new one

4.3. - OracleAS Dynamic Clusters (topology extensions and modifications)

One of the key new features of OracleAS 10g (10.1.3) is its flexibility. In OracleAS
10g Release 3 (10.1.3) the Process Notification framework has been enhanced to
adapt to the addition and elimination of nodes in a cluster without requiring any
restarts or configuration changes in the rest of the participant in the cluster. In
OracleAS 10g Release 3 (10.1.3) Oracle Notification Server (ONS) manages
network connections and handles notification, routing, ordering and delivery
dynamically. All events, including the addition and removal of a new node to a
topology, are dynamically notified to the other participants in that topology.

Previous versions of ONS required each instance server to be manually configured.
In OracleAS 10g (10.1.3) ONS can be configured to automatically discover other
ONS servers and automatically manage the connection topology.

Flexible discovery options to handle different requirements are provided

- Multicast discovery
- Discovery Server
- Gateways

This is a key feature of OracleAS 10g (10.1.3) and enables the quick creation of
cluster topologies where nodes join and leave a cluster as needed. Together with
Dynamic Resource Manager (see section 3.3), which enables triggering actions based
on metrics automatically obtained at runtime by OracleAS, this makes possible a
pro-active management of a system and offers great benefits from the system’s
availability point of view (crash prevention)
5. – BEST HIGH AVAILABILITY WITH ORACLE DATABASE

Oracle Application Server's High Availability solutions can be used in concert with the Oracle Database's high availability features, thus offering an end-to-end High Availability that no other vendor can match.

Middleware applications typically maintain information both in backend databases and in configuration files. To consistently recover an application, a synchronized backup of database data and configuration files on a file system is necessary. As the best application server for Oracle Database, OracleAS provides complete backup and recovery system that backs up and recovers the application server and the database synchronously. With 10g Database, the OracleAS Recovery Manager also supports quick recovery from human errors via Flashback in Oracle Database.

Oracle Application Server Guard integrates with the industry-leading Oracle Database disaster recovery solution, Oracle Data Guard, to offer a fully automated disaster recovery solution that protects the database, the application server, and deployed applications as a whole.

Further than that, Oracle Application Server makes it easier to develop robust applications with the latest Oracle Database high availability features. For example, the Fast Connection Failover feature (available starting with Oracle 10gR1 JDBC drivers) is integrated into Oracle Application Server 10g Release 3 data sources. While developers using other application servers enable Fast Connection Failover programmatically, OracleAS developers can simply modify one parameter declaratively to enable this powerful new feature. Fast Connection failover is a clear demonstration of the high degree of integration between the middle tier and the database tier in an Oracle environment. The failure notification mechanisms of this two tiers have been integrated, allowing the database quickly notify the middle tier whenever a failure happens in a DB instance. Fast Connection Failover supports database events notifications and load balancing of connections across active RAC.
instances, adjusting the jdbc pool based on these events. This provides a more efficient resource allocation and hence greater scalability and performance.

Compared to other application servers that utilize Oracle DB, Oracle Application Server 10gR3 works with Oracle DB to deliver high availability solutions that are more integrated, less expensive, and easier to use.

SUMMARY.-
High Availability has shifted from a mission-critical requirement to a general requisite that affects all types of deployments, especially in Service Oriented Architectures, where the outage of a single component/application, can affect multiple running operations.

Application servers need to provide end-to-end protection against all type of failures for all the services and all the components that are used by an application. Oracle Application Server 10g Release 3 provides a rich set of features to protect a complete SOA environment against all type of failures. It leverages a common High Availability framework (system failure, data failure and disaster protection) for all the sub tiers and services in an application environment and has been enhanced to minimize the impact of configuration changes in the system’s availability at different levels (dynamic discovery, dynamic routing, rolling upgrades…) With Oracle Process Management and Notification, Oracle Application Server provides death detection and auto-restart for all the processes in all the sub-tiers. Oracle Application Server Clusters and Oracle Application Server Cold Failover Clusters, enable active-active and active-passive redundancy for all the layers and services used by an application. Oracle Application Server Backup and Recovery (for data failures) and Oracle Application Server Guard (for disaster protection) take care of recovering the complete environment required by the application to run. This integrated and comprehensive high availability framework makes possible to deploy SOA systems with maximum availability and guarantee of service.