ORACLE

SSL, Load Balancers, Rewrite, Redirect and More Advanced Configuration

Matt Topper – Oracle National Security Group
Dan Norris – Piocon Technologies, Inc.

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

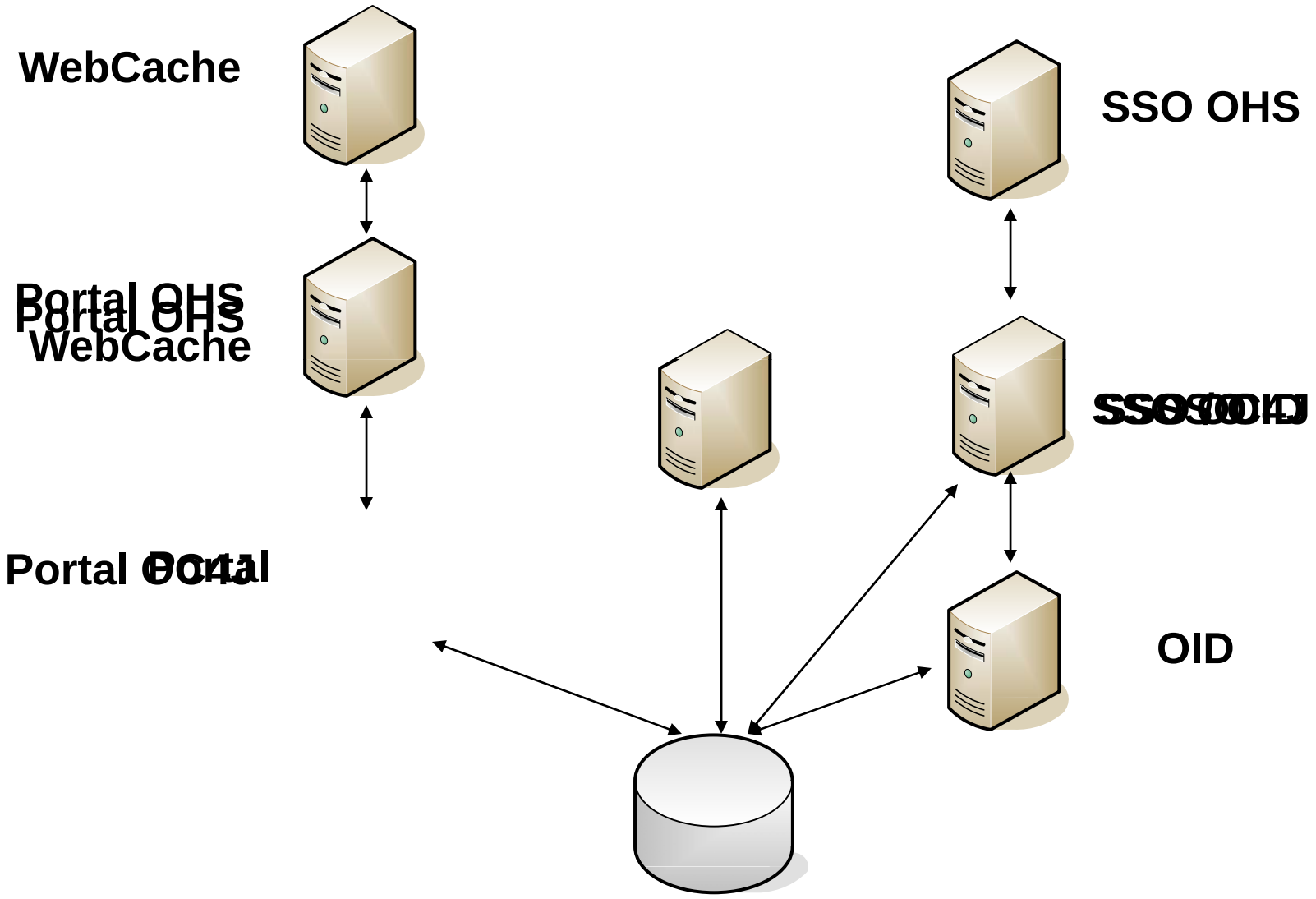
Agenda

- SSL, Load Balancers, Rewrites, Redirects and High Availability Configurations:
  - Enterprise Deployments
    - Deployment Types
    - High Availability
    - WebCache
  - Load Balancing
  - SSL
  - Rewriting URLs
  - The complete Oracle Fusion Middleware MAA: Disaster Recovery and Protection
- Q&A

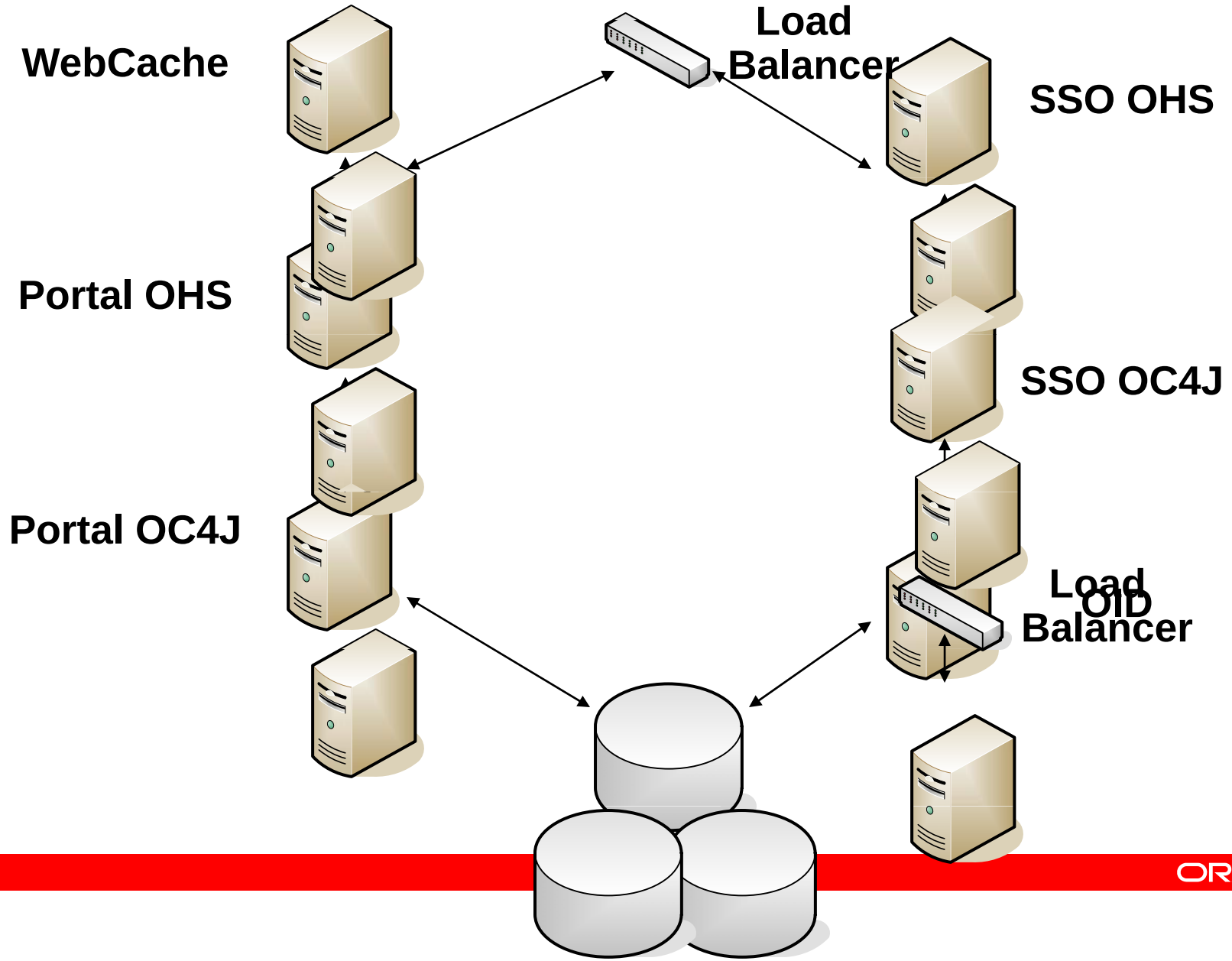
Who's running what?

- OID?
- Oracle SSO?
- Portal?
- Discoverer, Forms and Reports?
- Applications/eBusiness?
- SOA?
- OIM/OAM/OVD?
- BIEE?
- Web Center?
- jRuby?
- Anything else?

Scalability



High Availability

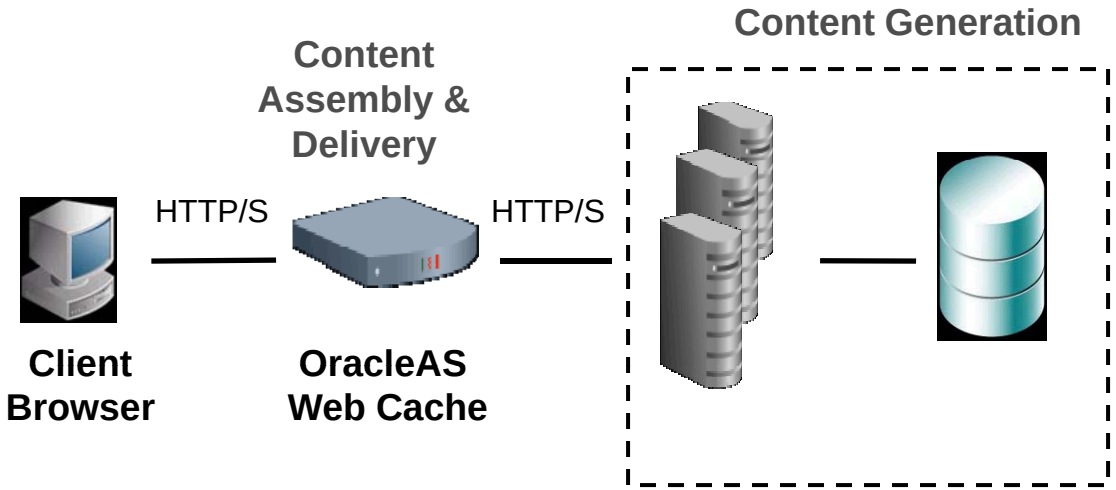


Oracle Web Cache

OracleAS Web Cache

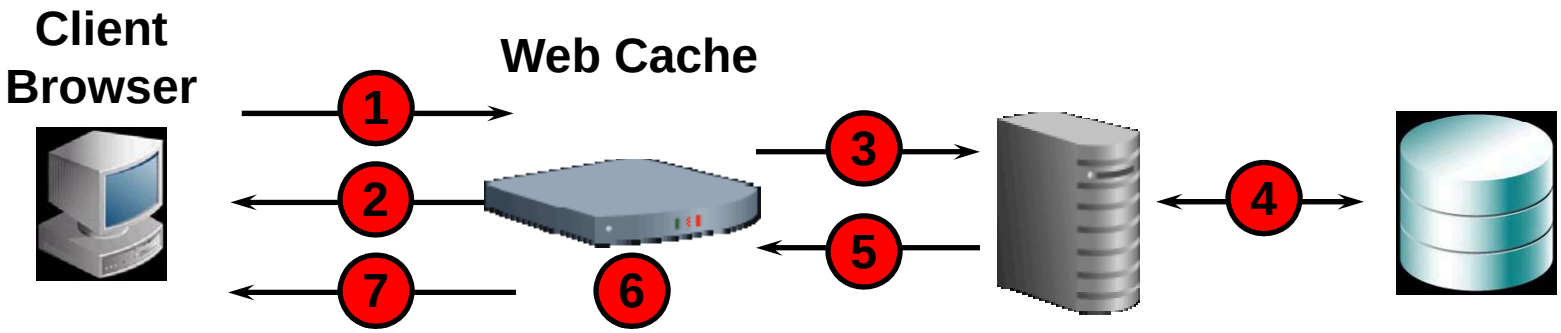
What is Web Cache?

- Reverse proxy cache and compression engine
- Deployed between browser and HTTP server



- Make more effective use of low-cost, existing hardware
  - Enables high hit-rate caching (which improves performance and scalability)
- Maintain quality of service with workload management
  - Improves system reliability
- Manage the end-user experience
  - Supports end-user performance monitoring and flexible deployment options

How Web Cache Works



1. Client sends HTTP request
2. Web Cache responds immediately if cached object is available
3. If object is not in cache, Web Cache requests object from Application Server
4. Application Server generates response (may include Database queries)
5. Application Server responds to Web Cache
6. If response is cacheable, Web Cache retains a copy for subsequent requests
7. Web Cache compresses page and responds to Client

Efficient Use of Low-Cost Hardware

IT Problem

- Dynamic, Web-based applications are compute-intensive, yet IT planners are asked to cut costs and do more with less

Web Cache Solution

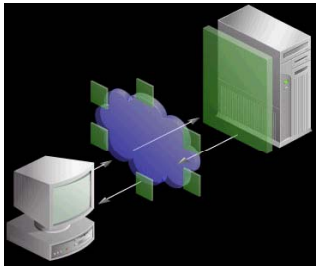
- Efficient use of low-cost, existing hardware
  - Automatic compression
  - Static content caching
  - Dynamic content caching
  - Expiration
  - Invalidation
  - Partial-page caching

Automatic Compression

- Problem: poor response times for end-users in remote locations or with low-bandwidth clients

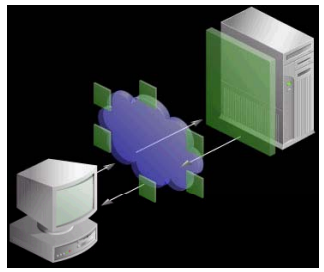
- Solution: automatic GZIP compression
  - All standard Web browsers support GZIP expansion
  - HTML typically compresses down to 1/10 original size
  - Web Cache compresses both cacheable and non-cacheable content, and stores content in compressed form
  - Streamed delivery
  - *No application changes required*
- Benefits:
  - Reduced network latency for dial-up, WAN and Wireless clients
  - Bandwidth savings

Static Content Caching



- Problem: repetitive requests for voluminous static content
- Solution: static content caching
- Benefits: get immediate boost by caching static files
  - GIF, JPEG, PDF, HTML, CSS, JavaScript, Applets, SWF, WAV, Shockwave, etc.
- Static content caching is easy
  - *No application changes required*
  - Most file types cached “out of the box” using default policies

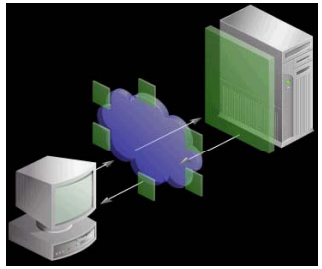
Dynamic Content Caching



- Problem: dynamically generated content strains infrastructure
  - JSP, Servlet, ASP, PERL, PHP, etc. put added strain on mid-tier and database
- Solution: policy-based caching with Web Cache
  - Caching policies specified by administrator (rules) or by application (response headers)
- Benefits: higher cache hit rates
  - Offload content generation infrastructure

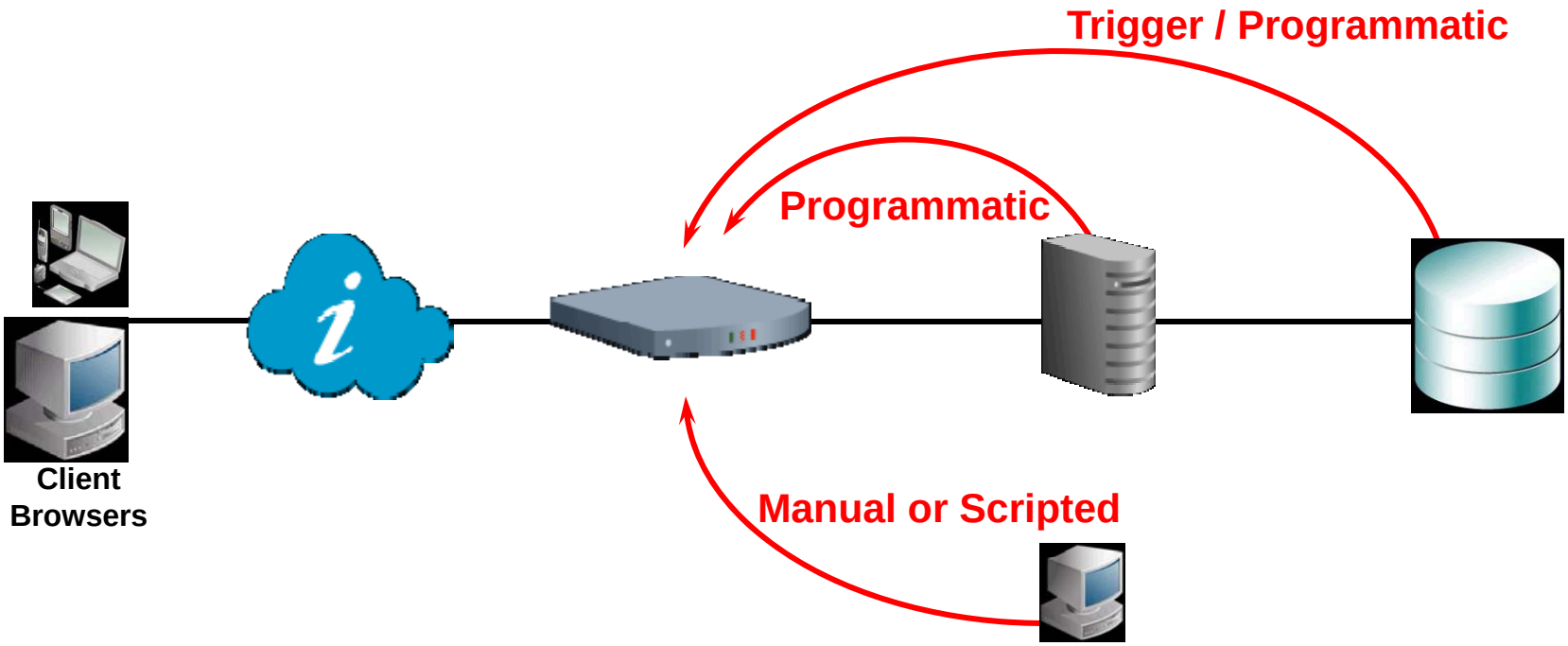
Priority	Selectors			Cache/Don't Cache	Compression	Detailed Settings
	URL Expression	HTTP Method(s)	POST Body Expression			
1	\.html?$	GET, GET with query string		Cache	On for all browsers	details
2	\.(gif|jpe?g)$	GET, GET with query string		Cache	Off	details
3	/OA_HTML/ibeCCtpBuyRoute\.jsp.*	GET, GET with query string, POST	.*	Don't Cache	On for all browsers	details
4	/OA_HTML/ibeCCt.*\.jsp.*	GET, GET with query string, POST	.*	Cache	On for all browsers	details
5	/OA_HTML/ibeSQtdItemSrch\.jsp.*	GET, GET with query string, POST	.*	Cache	On for all browsers	details
6	/OA_HTML/ibeSQtpSrch\.jsp.*	GET, GET with query string, POST	.*	Cache	On for all browsers	details
7	/OA_HTML/ibeSQtpSrpSrchRslts\.jsp.*	GET, GET with query string,	.*	Cache	On for all browsers	details

Multi-version URL Caching



- Personalized pages with the same URL
- Same URL, different cookies or headers
  - For example, `Accept-Language=en-us`
  - For example, `region=Brazil`
- Same path prefix, with different session ID at the end of URL
  - For example, `http://admissions.anycollege.edu/course_info?course=123&student=9e7shew`
  - OracleAS Web Cache can ignore `student=9e7shew`

Staying Fresh with Invalidation

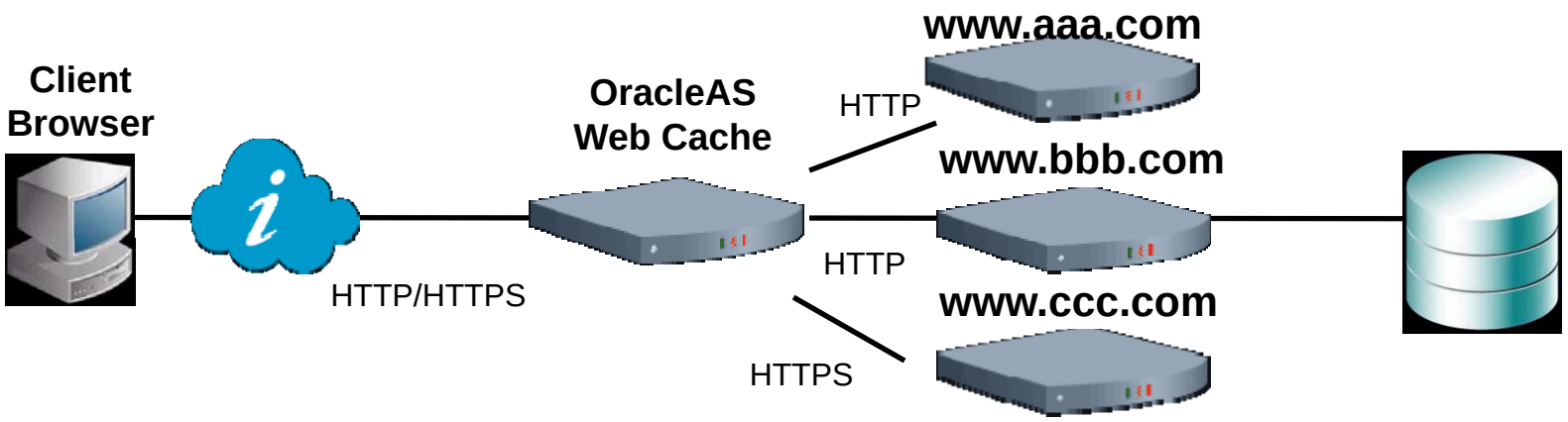


Consistency Management Mechanisms:

- Expiration Policies (for predictable changes)
- Invalidation Messages (for unpredictable changes)
  - Scope: fine-grained or broad; support for application-specified Search Keys
  - Effect: remove immediately or specify a refresh grace period
  - Format: XML over HTTP (Java and PL/SQL APIs provided)

Virtual Hosting

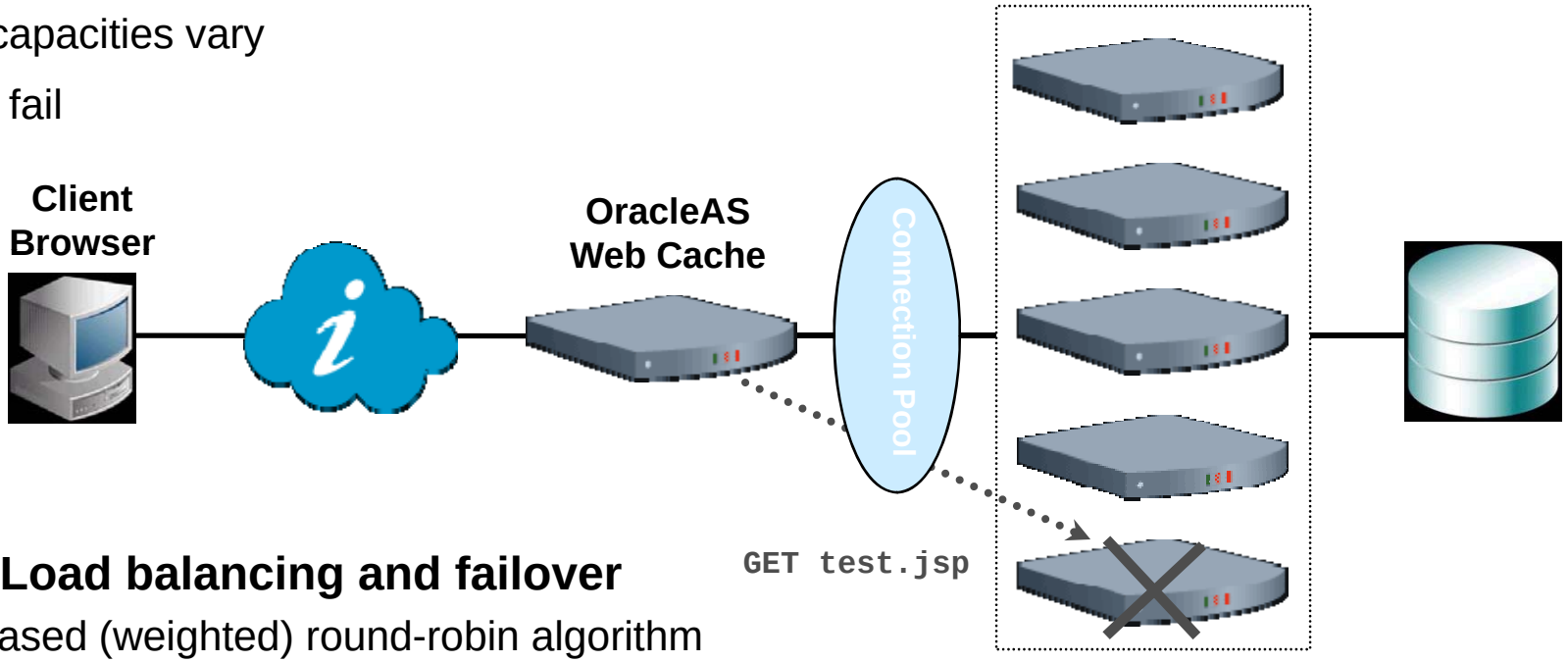
- Problem: Use multiple vendors of application servers running different applications (e.g. hosting services)
- Solution: Use Web Cache to front-end all application servers
- Benefits: Single point of entry for easy management



Load Balancing and Failover

Problem:

- Server capacities vary
- Servers fail



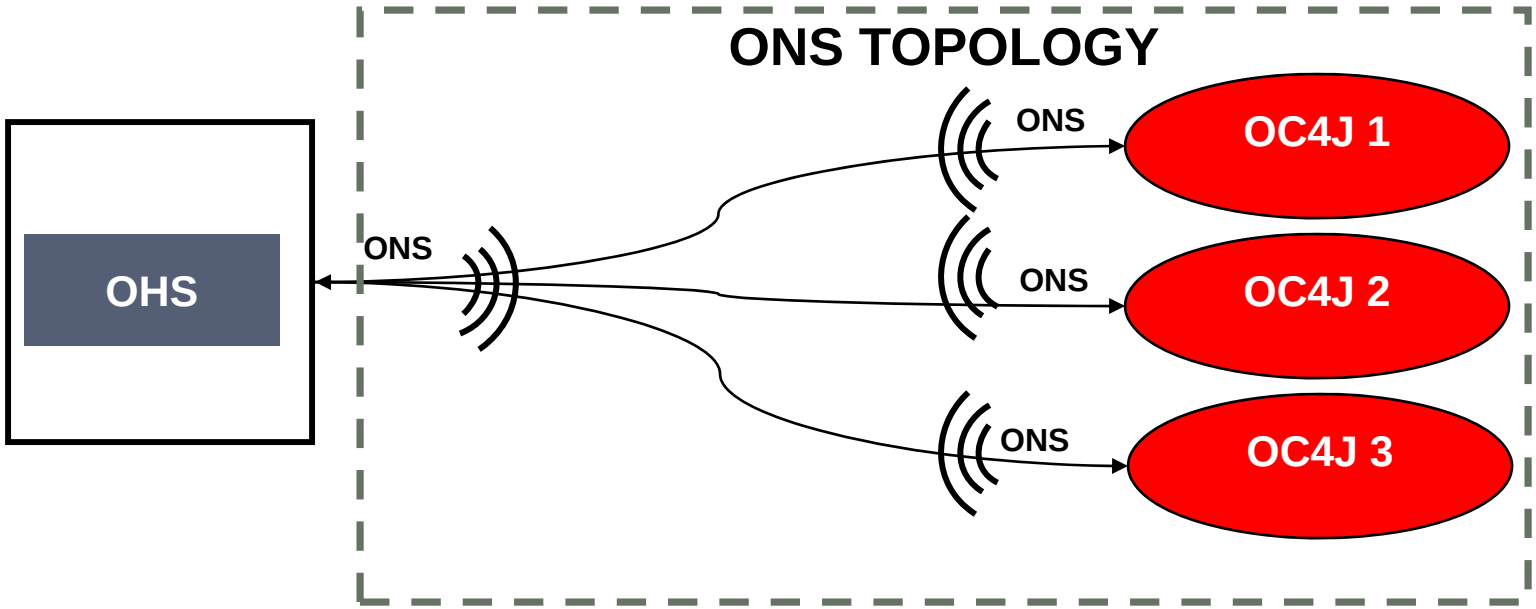
Answer: Load balancing and failover

- Policy-based (weighted) round-robin algorithm
- Cookies and session-IDs used to maintain server affinity for stateful applications
- Layer 7 status checking for failure/recovery detection
- Connection pooling for TCP connection reuse
- *Benefits: enable policy-based resource usage and mask server failures*

Oracle Clustering

Fusion Middleware Maximum Availability Architecture

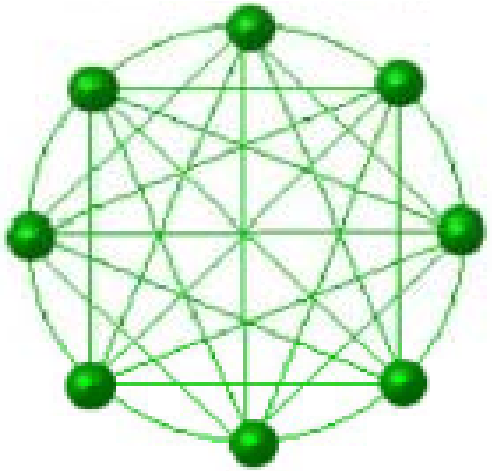
- OracleAS Clusters: instances share a workload based on the Oracle Notification Server Topology that they participate at



Topology Discovery

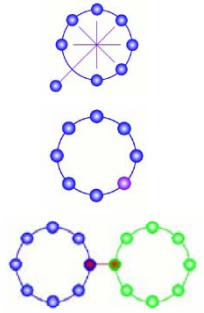
- 10.1.3 New Feature
- Previous versions of ONS required each instance server be manually configured (typically by DCM)
- Fully connected instance map
- Induces system brittleness and extra management overhead

Fully Connected (Static)



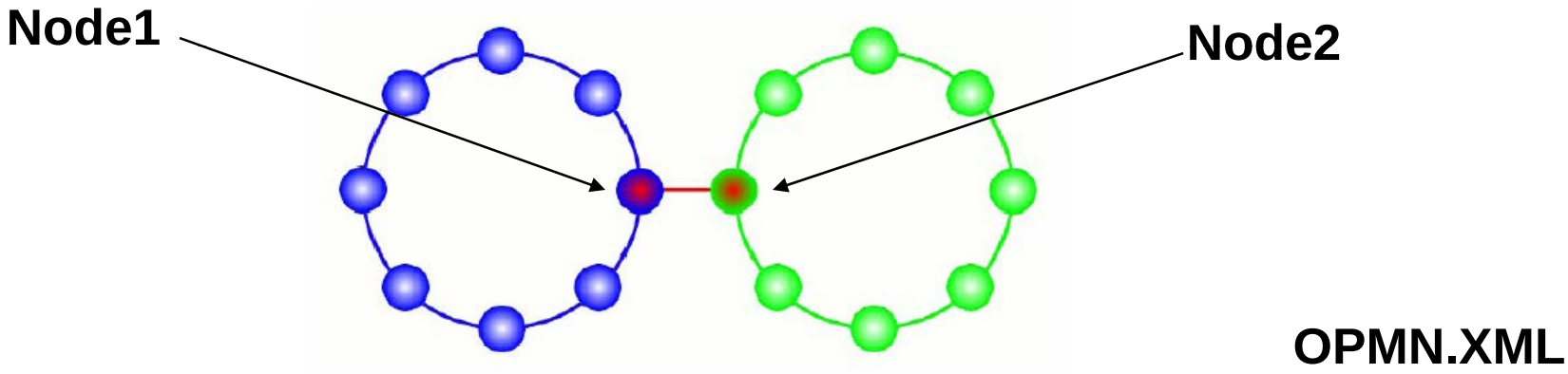
Topology Discovery

- Flexible topology options to handle different requirements
  - Multicast discovery
  - Server discovery
  - Gateways



Fusion Middleware Maximum Availability Architecture

- Topology recomendations example, default gateway config:

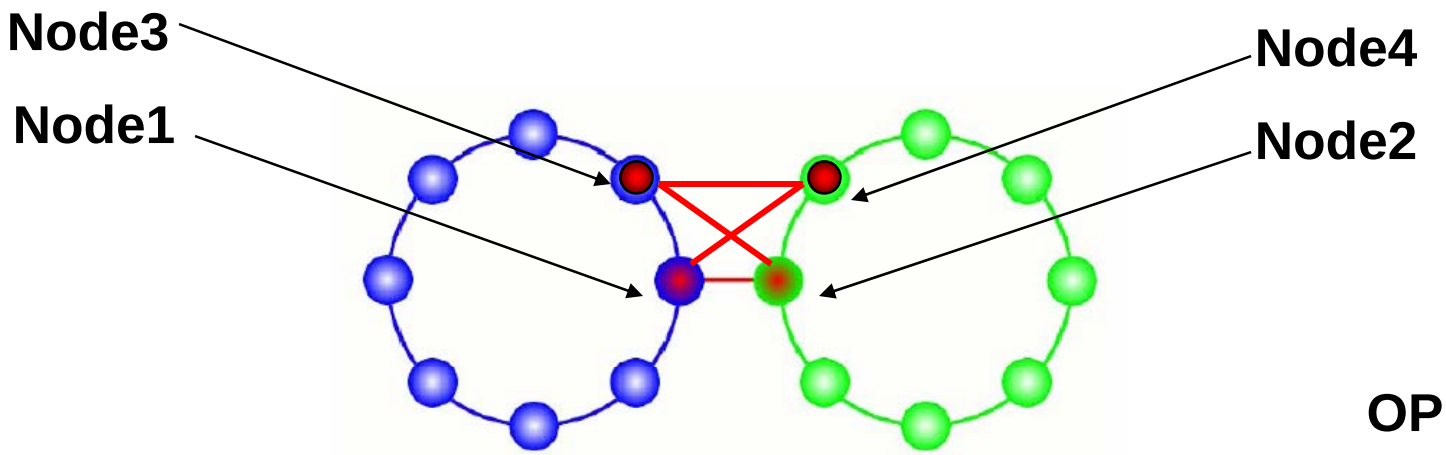


OPMN.XML

```
...
<gateway list="node1.com:6200&amp;node2.com:6200/"/>
<discover list="*224.0.0.0:8200"/>
...
```

Fusion Middleware Maximum Availability Architecture

- Topology recomendations example, redundant gateway config:



OPMN.XML

```
...
<gateway list="node1.com:6200,node3.com:6200,node2.com:6200,node4.com:6200/"/>
<discover list="*224.0.0.0:8200"/>
...
```

Fusion Middleware Maximum Availability Architecture

- Toplogy recomendations:
  - Multicast is the most flexible topology config...but requires multicast (overhead is mimimum though)
  - Use Gateway option for topologies that span multiple networks (firewalls between different tiers that need to be connected)
  - When using either Server Discovery or Gateways, specify multiple contacts to mantain connectivity in case of a failure in one of the gateway nodes or discovery server

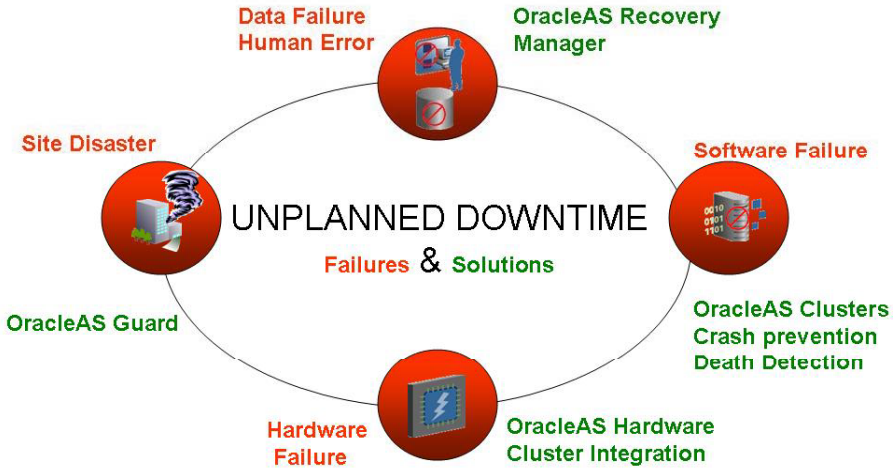
Maximum Availability

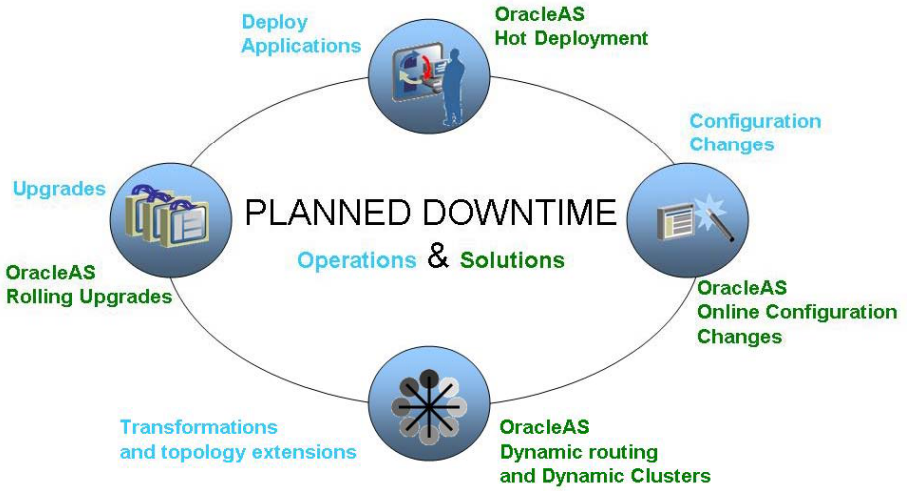
Oracle MAA and Oracle FM

- Oracle Maximum Availability Architecture (MAA) is Oracle's best practices blueprint based on proven Oracle high availability technologies and recommendations
- Oracle FM is a comprehensive and well-integrated family of products that offers complete support for development, deployment, and management of Service Oriented Architecture
- As the most critical piece in any multi-tier Oracle System, Oracle FM has joined the MAA effort to provide best practices in the Application Server infrastructure area

Fusion Middleware Maximum Availability Architecture

- An Oracle Fusion Middleware MAA is a system that has been configured to minimize planned and unplanned downtimes



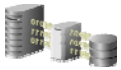


Fusion Middleware Maximum Availability Architecture

- The Key pieces in a Fusion Middleware Maximum Availability Architecture:

  - Process Management and death detection:
    - If a process dies, detect it quickly and IF POSSIBLE restart it

  - Redundancy:
    - Provide redundant components that can take over in case of a planned or unplanned downtime

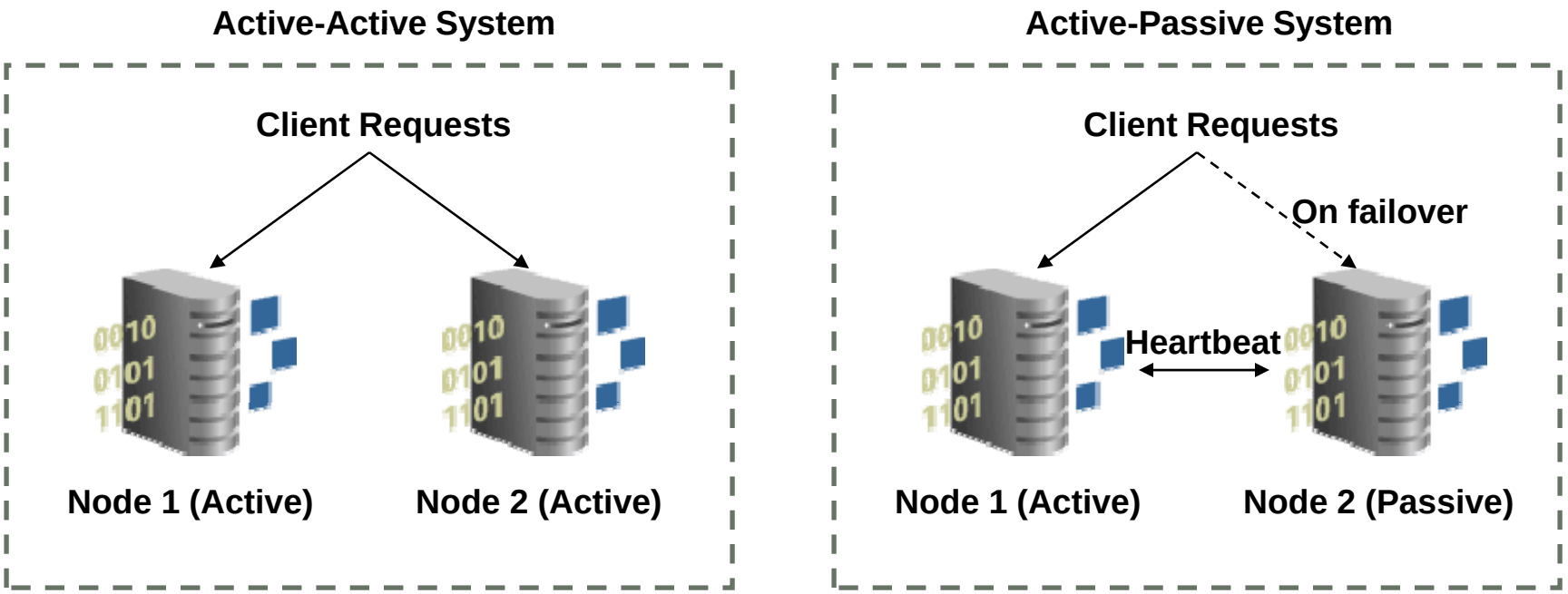


  - Connection management:
    - Make sure that tiers load balance their outbound connections and respond accordingly to failures in other tiers

Redundancy for Oracle FM Architectures: TOPOLOGIES

Fusion Middleware Maximum Availability Architecture

- Redundancy: Oracle FM systems can be configured in redundancy using two different models

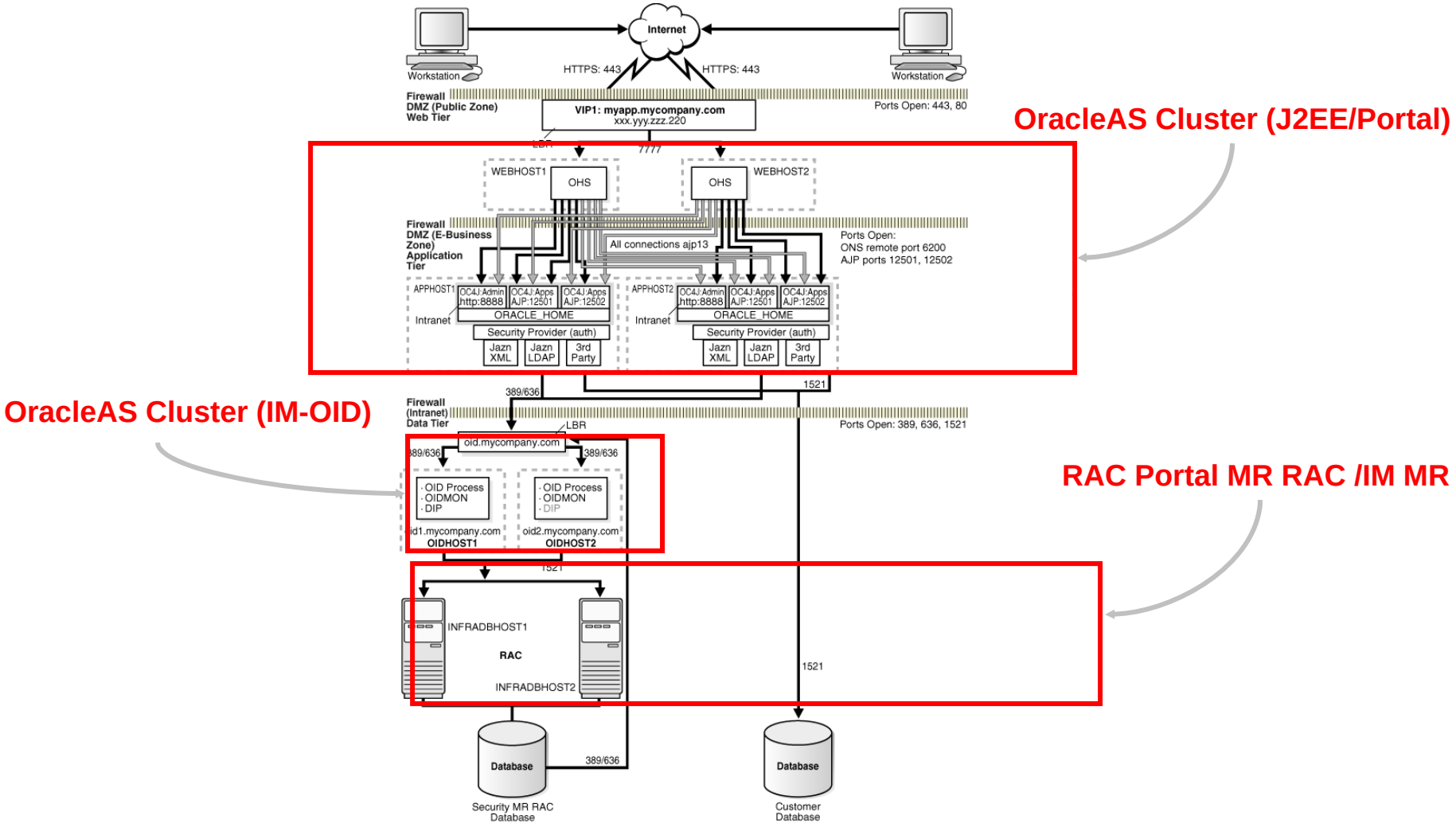


Fusion Middleware Maximum Availability Architecture

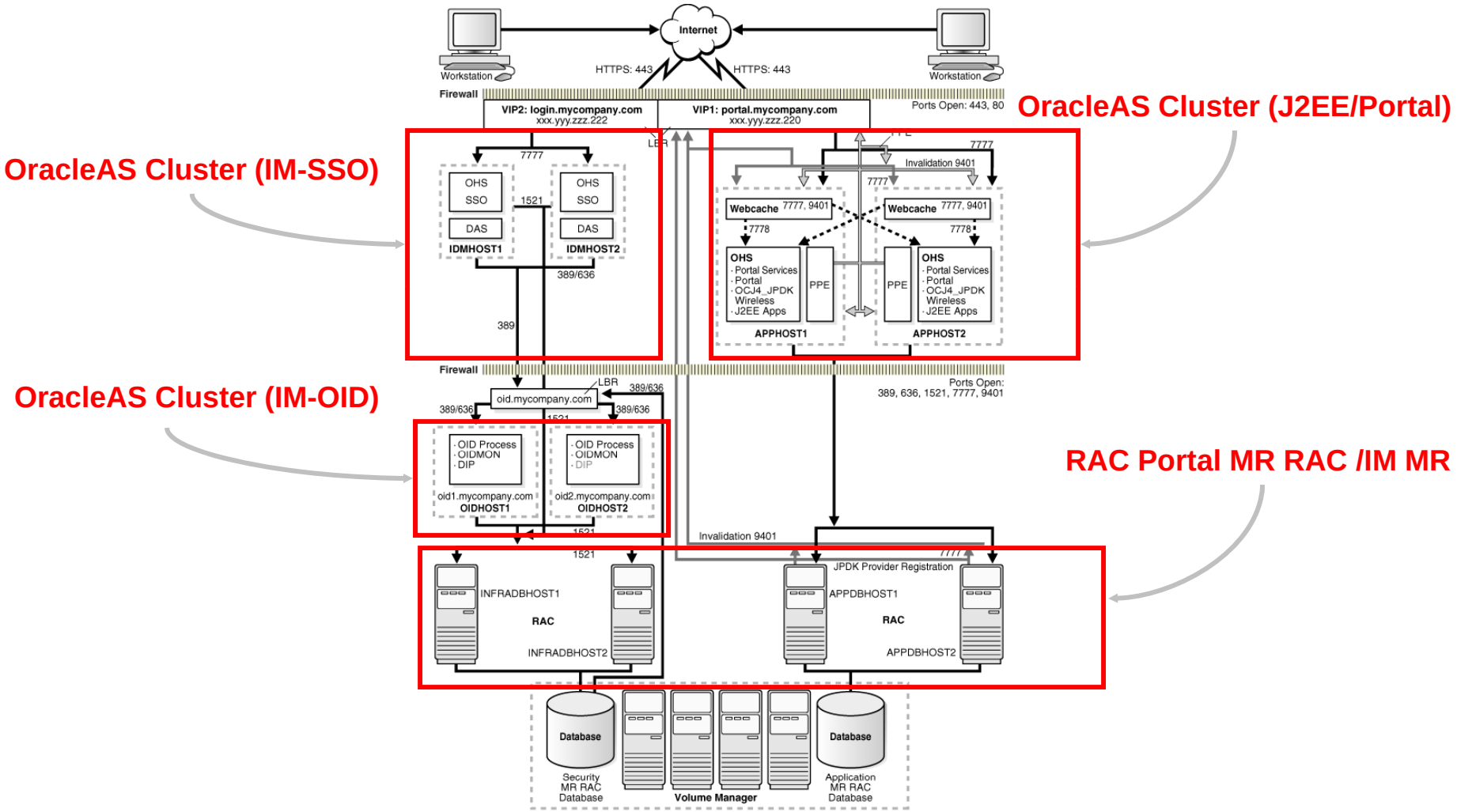
- Redundancy: For Maximum Availbility and scalability, use Oracle Application Server Clusters (AA)

	Oracle Application Server Clusters	Oracle Application Server Cold Failover Clusters
Scalability	☺	☹
Operational Cost	😐	☺
Licensing costs	😐	☺
High Availability	☺	😐
Installation	😐	☺

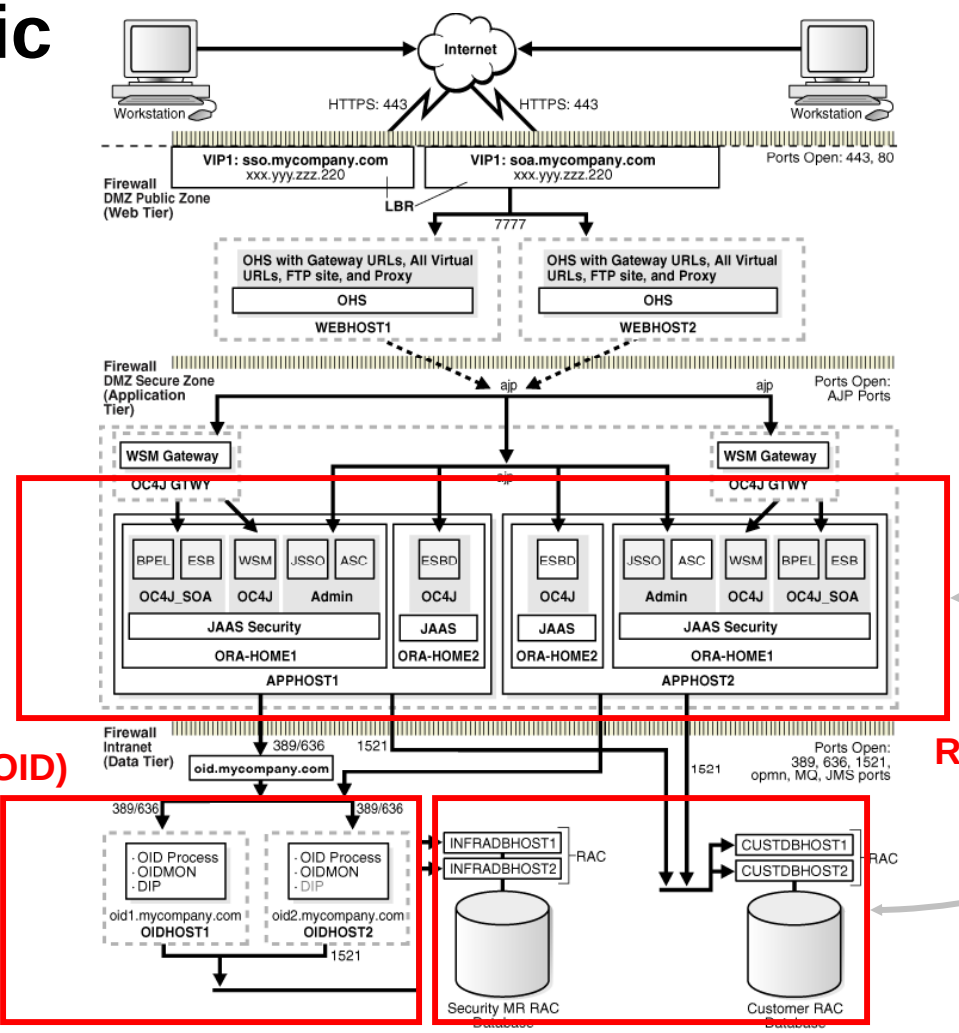
Fusion Middleware Clusters in a Maximum Availability Architecture



Fusion Middleware Clusters in a Maximum Availability Architecture with SSO



Fusion Middleware Clusters in a Maximum Availability Architecture SOA Components specific



OracleAS Cluster (SOA)

OracleAS Cluster (IM-OID)

RAC Portal MR RAC /IM MR

Redundancy for Oracle FM Architectures: SESSION REPLICATION

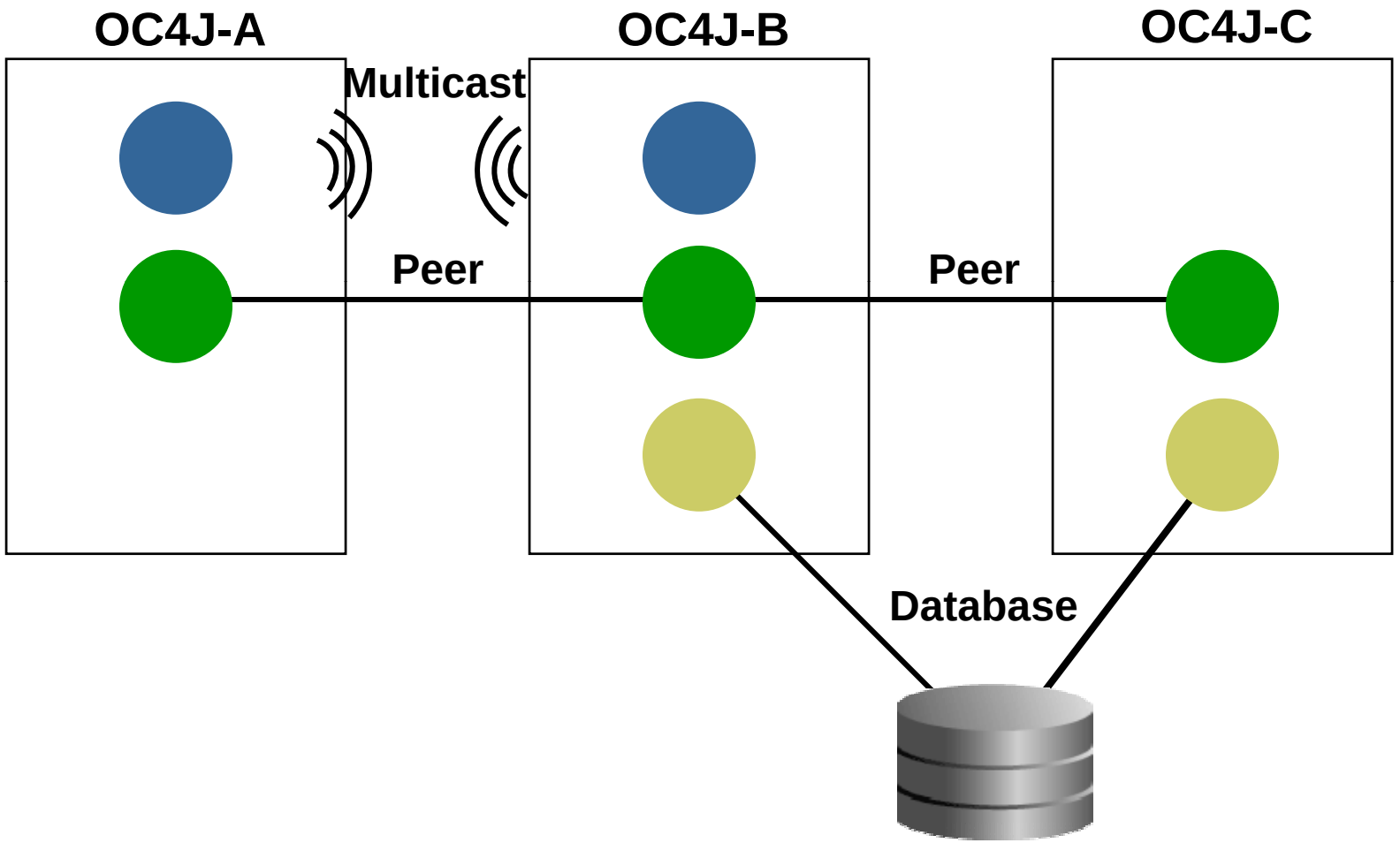
Fusion Middleware Maximum Availability Architecture

- Oracle Application Server Session Replication framework
  - Application level clustering or container level clustering
  - Ability to choose the number of nodes to replicate to
  - Common framework for HTTPSessions and Stateful Session Beans
  - Support for different replication protocols
  - Multiple replication triggers

Application Centric Model

- Applications become the focal point in a cluster
  - Application is fully self-contained, resources defined
  - Application can be configured to participate in state replication activities
  - The state of an applications is replicated to other instances of the application running in the cluster
  - Different replication types can be used by different applications
- OC4J instances can host clustered and non clustered applications
  - Become pooled container resources
  - No need to be homogenous

Session replication Protocols

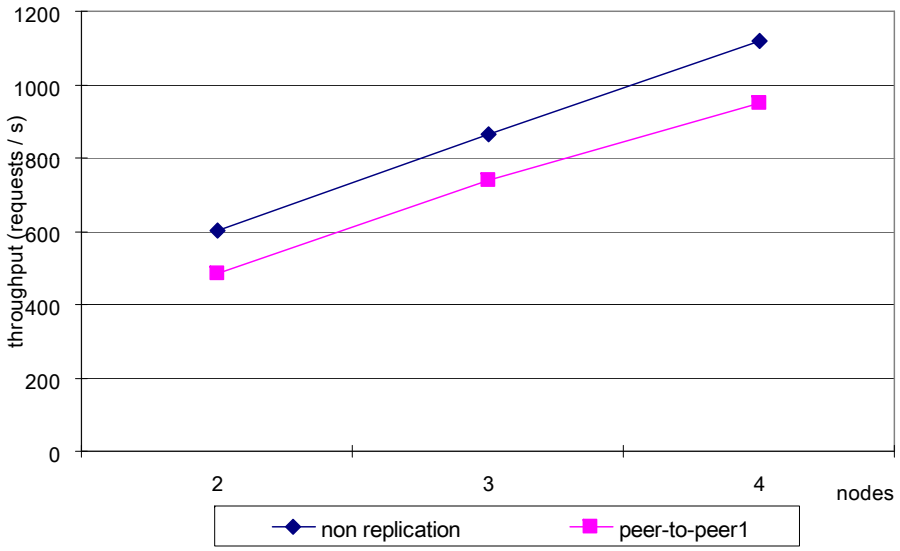


Fusion Middleware Maximum Availability Architecture

- The In-Memory based protocols are the less costly to replicate state
  - You can tell OC4J to obtain the list of peers to replicate to from the opmn topology (no need to hard code the list of peers, so very easy to configure and maintain)
  - Peer-to-peer is very reliable, and more performant by default (multicast may require tuning)
- For guaranteed session store and recovery the database protocol is the best choice
  - Has a higher cost than in-memory protocols
  - For frequent session updates will reduce performance
  - Requires additional resources for data store

Fusion Middleware Maximum Availability Architecture

- Peer-to-peer is very reliable, and more performant with default settings. ~10% performance impact over non replication group
- Multicast can offer even better performance than peer to peer but… with tuning



Session Replication-Policies

- onSetAttribute: Replicates each change made to an HTTP session attribute at the time the method “setAttribute” is invoked (different from the time when the object value is modified).
- onRequestEnd: Queue all changes made to HTTP session attributes, then replicate all changes just before the HTTP response is sent
- onShutdown: Replicates the current state of the HTTP session and EJB bean instances whenever the JVM is shutdowns

Fusion Middleware Maximum Availability Architecture

```
class TestServlet extends HttpServlet() {
        doGet() {
                PrintWriter out = response.getWriter();
                /* doing business logic … */
                HttpSession.setAttribute();
                doSomethingElse();
                nowDoMOreThings(),
                HttpSession.setAttribute();
                out.flush();
                out.close();
        }
}
```

- In onSetAttibute replication happens twice

- In onRequestEnd replication happens only here

Fusion Middleware Maximum Availability Architecture

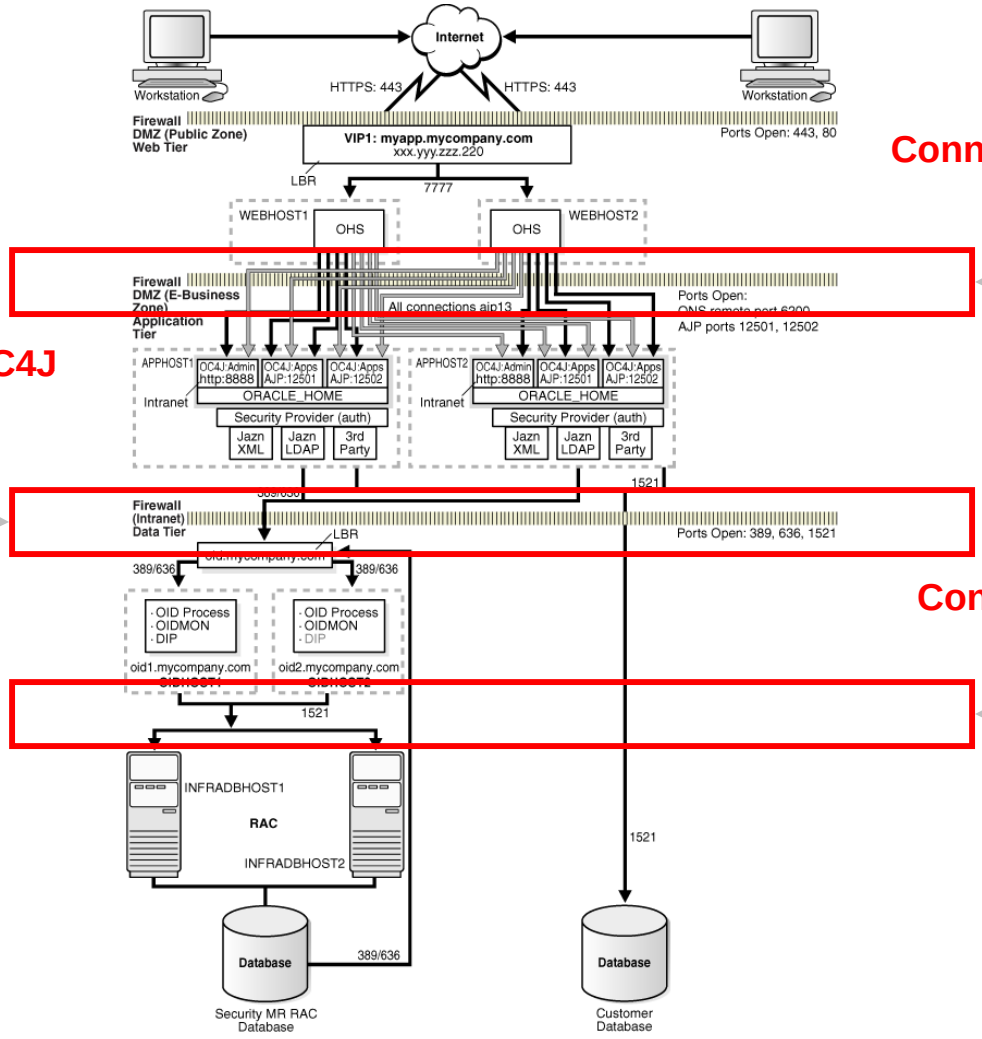
- onShutdown is the most performant, but very risky
- Since OHS retries requests automatically if an error is experienced in the middle of an invocation, onSetAttribute is less useful (can be powerful in standlone OC4J environments)
- onRequestEnd is more performant and hence provides a faster replication (hence, is more reliable!)

Redundancy for Oracle FM Architectures: Configuring connections

Fusion Middleware Maximum Availability Architecture

- Oracle Fusion Middleware architecture spans multiple tiers
- Components communicate through the network and load balance requests to other components/services
- Proper adjustment of these connections is critical for maximum availability

Fusion Middleware Maximum Availability Architecture: most relevant connections



Connections from OHS to OC4J

Connections from OC4J to OID/LDAP

Connections from OID and OC4J To the DB

Fusion Middleware Maximum Availability Architecture: most relevant connections

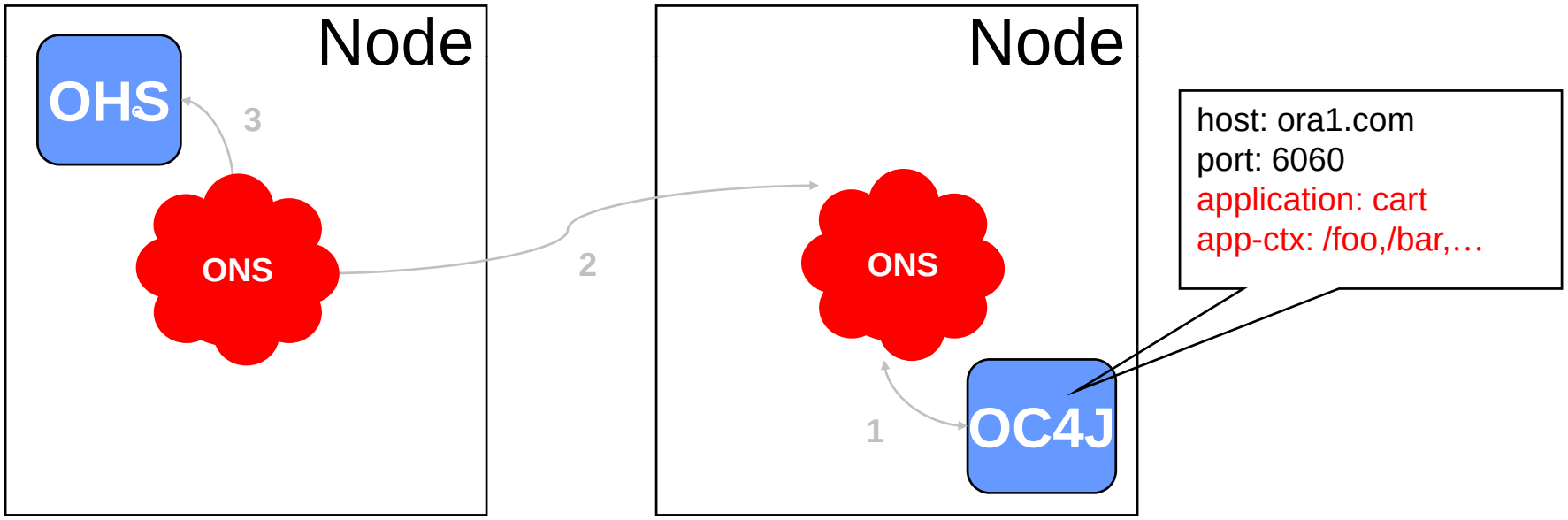
Tiers Involved	Critical? (0-5)	Oracle FM LB mechanisms
Webcache to Http servers	3	Round-robin, weighed round robin
Http servers to j2ee containers (mod_oc4j)	5	Random, Round Robin, Random with Local Affinity, Round Robin with Local Affinity, Random using Routing Weight, Round Robin using Routing Weight, Metrics Based, Metric Based with Local Affinity
J2ee containers to LDAP	3	Load Balancer dependent
J2ee containers to database	5	Runtime load balancing mechanisms

Dynamic Routing between OHS and OC4J

- mod_oc4j routes requests from Web Server (OHS) to J2EE Containers (OC4J) instances where application is running
  - OC4J now announces mappings
  - Ajp13 on steroids
  - mod_oc4j dynamically adds mappings to its route table
- Removes system down time during routing updates
- Reduces maintenance

Dynamic Routing between OHS and OC4J

cart:
/foo/* ora1.com:6060
/bar/* ora1.com:6060

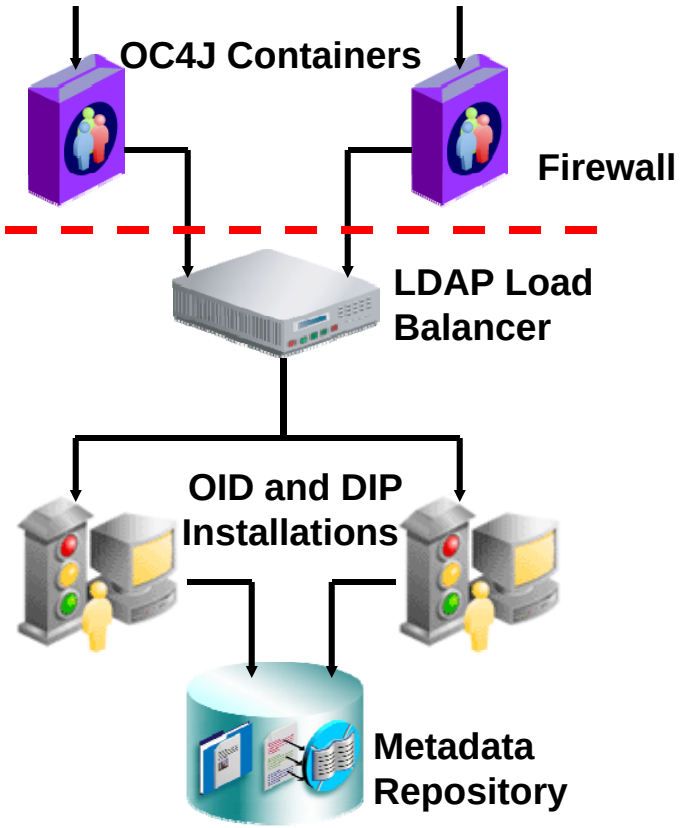


Fusion Middleware Maximum Availability Architecture

- Retry of connections:
  - Oracle HTTP Server RETRIES automatically any requests that fail before retuning data (i.e. if the container is reported “death” before sending any data back to mod_oc4j)
  - Make sure the application is “retry ready” and use proper transaction control to ellude any inconsistent retries

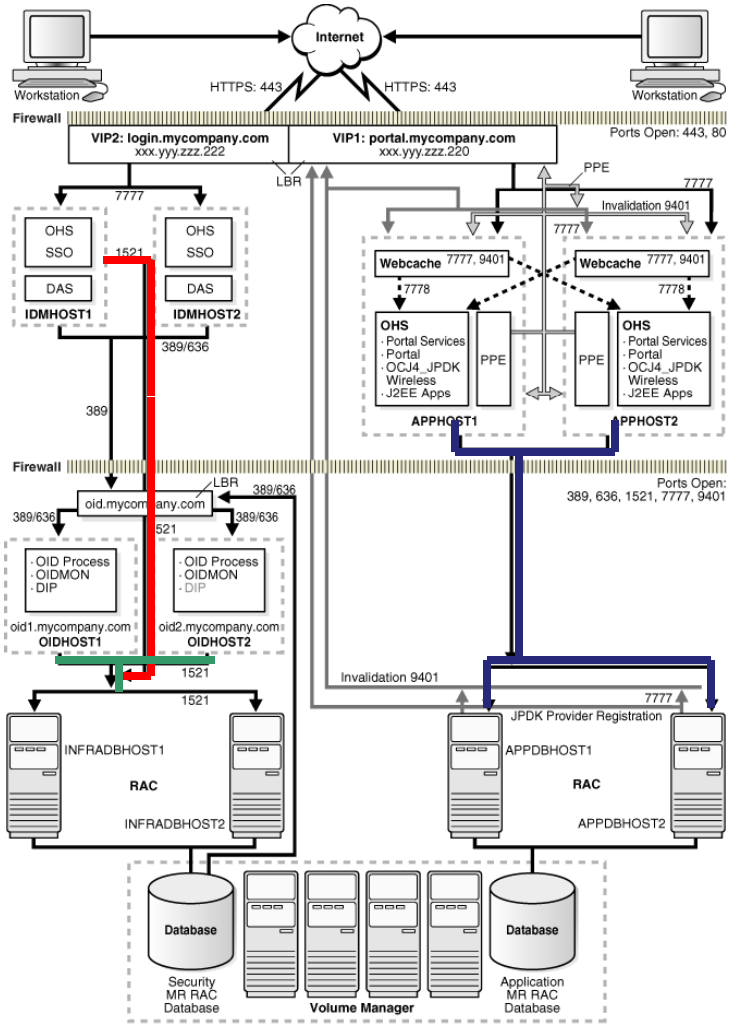
Fusion Middleware Maximum Availability Architecture

- Connections to OID/LDAP:
  - In tiered architectures the load balancer or firewall may terminate connections to Oracle Internet Directory.



Fusion Middleware Maximum Availability Architecture

- Connections to the DB:
  - Everyone knows how to connect to a DB
  - But not in the same way:
    - C components (OID) use Net connections
    - Java Components (OC4J) use JDBC connections
    - “Intelligent” modules (mod_plsql) use specific pools

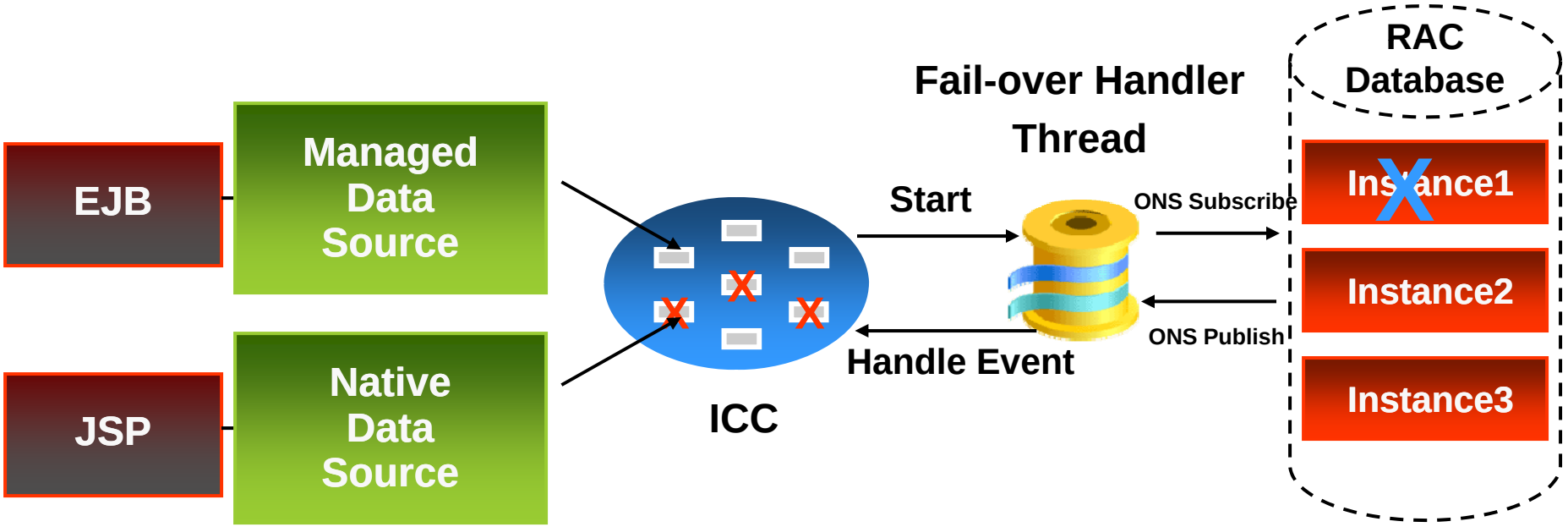


Fusion Middleware Maximum Availability Architecture

- Use TAF for OID
- Use mod_plsql conection validation feature for Portal and SSO connections:
  - **Automatic**: mod_plsql tests all pooled database connections which were created prior to the detection of a failure that could mean an instance failure.
  - **ThrowAwayOnFailure**: mod_plsql throws away all pooled database connections which were created prior to the detection of a failure that could mean an instance failure.
  - **AlwaysValidate**: mod_plsql always tests all pooled database connections which were created prior to issuing a request. Since this option has an associated performance overhead for each request, this should be used with caution.
  - **NeverValidate**: mod_plsql never pings any pooled database connection. This option always for older behavior in mod_plsql.

Fusion Middleware Maximum Availability Architecture

- Use Fast Connection Failover (feature of Oracle´s Implicit Connection Cache) for JDBC connections accessing RAC



Fusion Middleware Maximum Availability Architecture

- Fast Connection Failover configuration:
  - The implicit connection cache and FCF is enabled.
  - The application uses service names to connect to the database; it cannot use service IDs
  - The underlying database is a Release 10g (R1 or R2) Real Application Clusters (RAC) DB
  - Oracle Notification Service (ONS) is configured and available on the node where JDBC is running.

Fusion Middleware Maximum Availability Architecture

- Fast Connection Failover configuration:
  - It is critical to verify the proper connectivity through ONS between your middle tiers and the DB.
  - On RAC nodes:

    onsctl ping from OHOME/opmn/bin
  - On Middle Tier nodes

    opmnctl debug from OHOME/opmn/bin

Oracle FM MAA Architecture:

Disaster Protection for the entire Fusion Middleware Stack

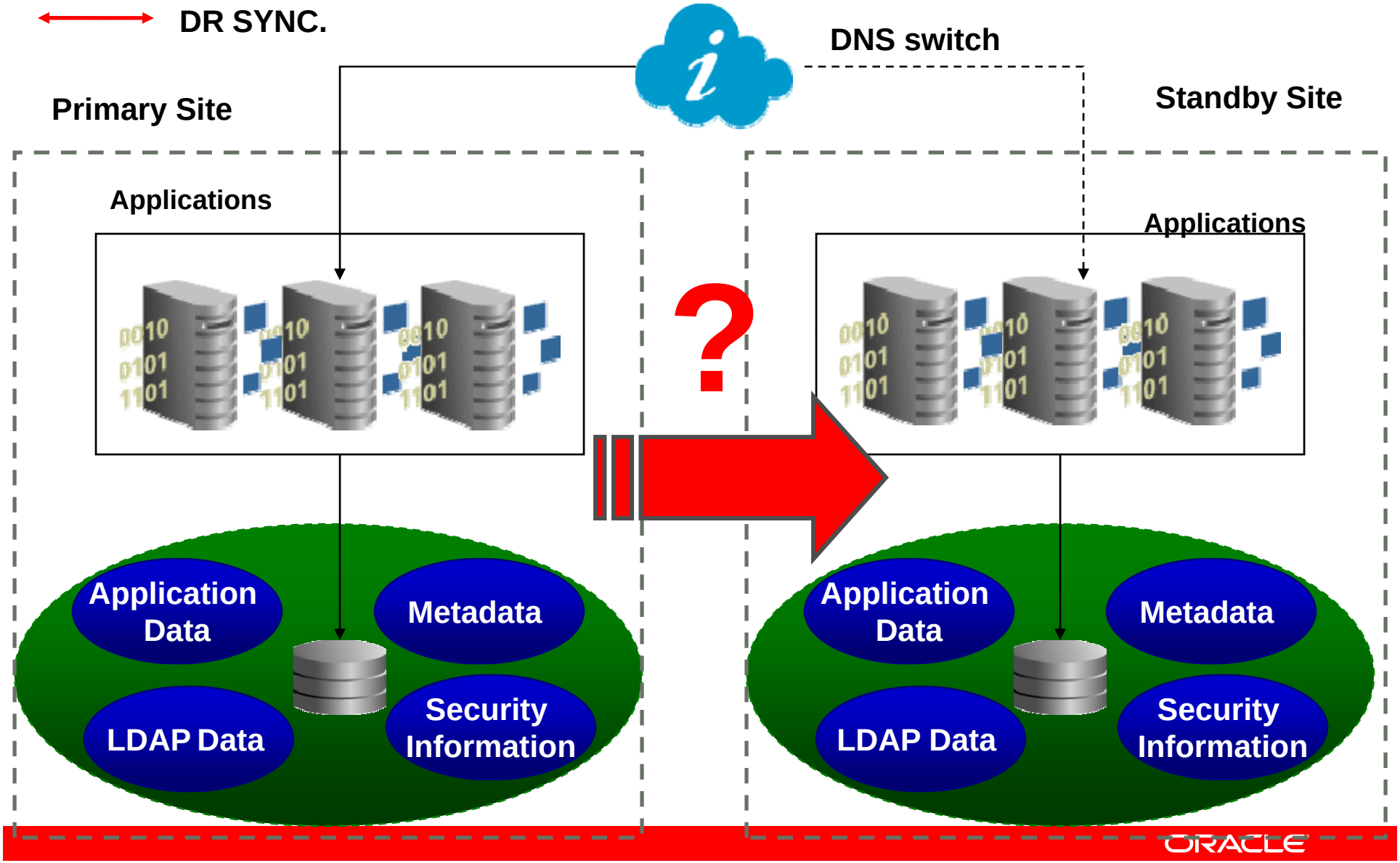
Fusion Middleware Maximum Availability Architecture

- So far we have focused on protection and recomendations that are scoped to a data center...but...what happens if I loose the entire datacenter???
- Local HA solutions protects individual systems
  - Hardware cluster, load balancer, disk array, etc.
- Disaster Recovery solutions protect the entire system
  - Redundant hardware
  - Redundant software
  - Redundant network
  - Redundant management

Fusion Middleware Maximum Availability Architecture

- A site-to-site solution enables the system to maintain availability regardless of a failure or extended system management operation
- A DR solution typically sets up a mirror site where the services and data of the production site are ready to take ove in case of a data center outage
- The major paradigm for a disaster protection solution is how to mantain an identical and consistent copy of that data and services

DR Paradigm


DR SYNC.
DNS switch
Primary Site
Standby Site
Applications
Applications
?
Application Data
Metadata
LDAP Data
Security Information
Application Data
Metadata
LDAP Data
Security Information
ORACLE

Fusion Middleware Maximum Availability Architecture

- Why some products that perform pure physical copy (remote mirroring or snapshots) are not the best approach for protecting Oracle FM stack? (I)
  - They require more powerful (and expensive) hardware
    - Special disk (SAN, NAS) and volume managers
    - Sometimes they even require the same type of storage in production and standby site
    - The overhead implied by remote mirroring solutions I considerable at i/o level but mainly at NETWORK level

Fusion Middleware Maximum Availability Architecture

- Why some physical copy (remote mirroring or snapshots) are not the best approach for protecting Oracle FM stack? (II)
  - They do not guarantee a logical consistency of the copy
  - They have not been tested by Oracle
  - They do not know about the dependencies between components in Oracle FM stack
  - What is worse…physical asynchronous replication solutions may not preserve write ordering across all mirrored volumes that the infra/oid database resides on, and can cause data corruption

  - ...so...WHAT DO WE DO?

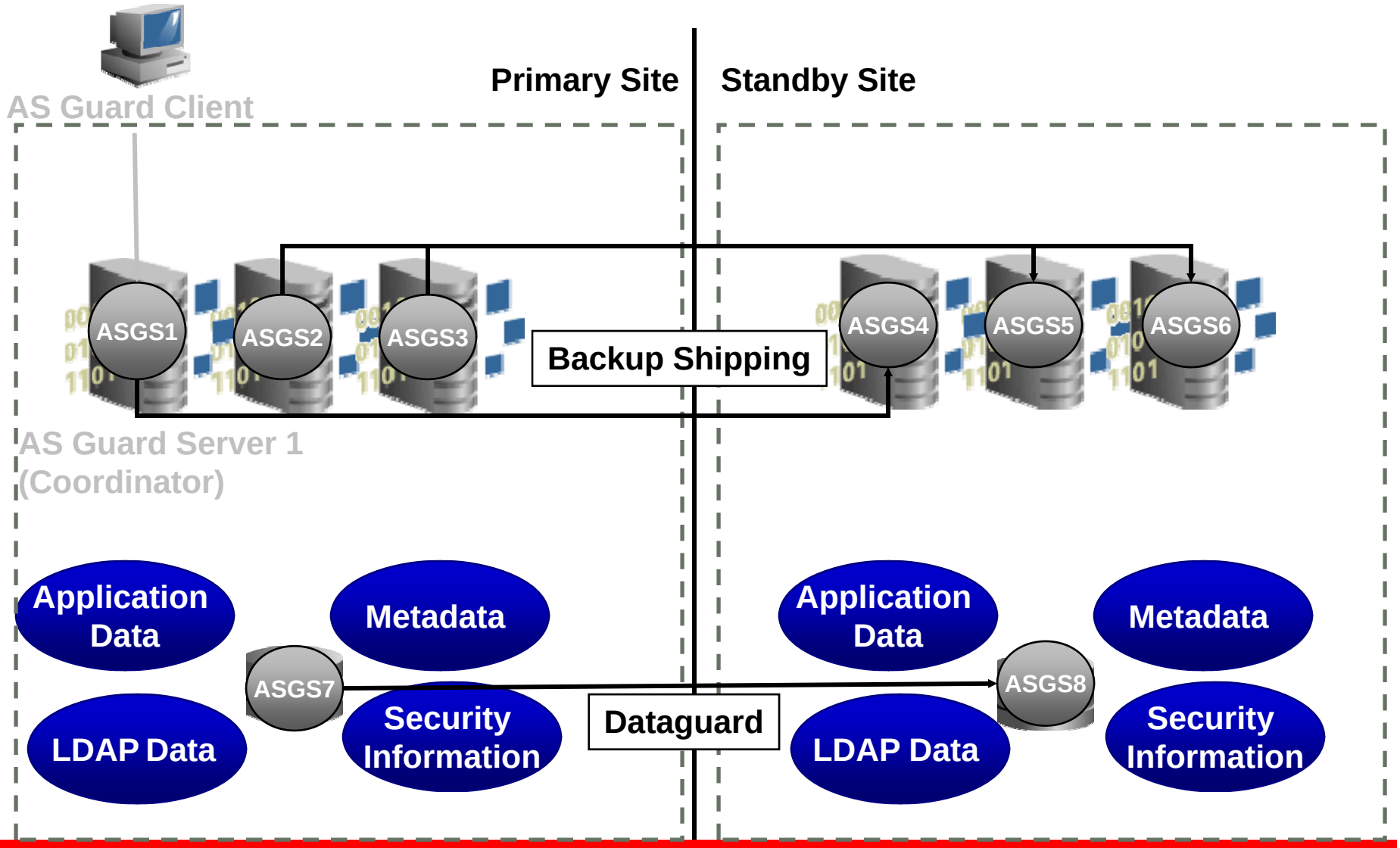
OracleAS Guard to the rescue!

- OracleAS Guard is actually an automated HA Best Practice!
  - Integrated with Oracle Data Guard but frees users from Oracle Data Guard management (automation is key to High Availability)
  - Leverages Oracle Data Guard advantages: corruption protection, performance, expense, etc.
- No fancy hardware or network (besides just what is necessary to install OracleAS)
- Minimum degree of expertise and less training than other solutions to operate regular lifecycle operations such as synch, failover, switchover

OracleAS Guard Architecture

- OracleAS Guard client
  - Is installed on every OracleAS install type
  - The OracleAS Guard client provides an asgctl command-line interface to perform instantiation, synchronization, switchover, failover, and monitoring operations
- OracleAS Guard server
  - A distributed server that runs in all the systems that participate in an OracleAS Disaster Recovery configuration
  - One ASG server acts as the coordinator for the operations being performed by the servers in the other instances

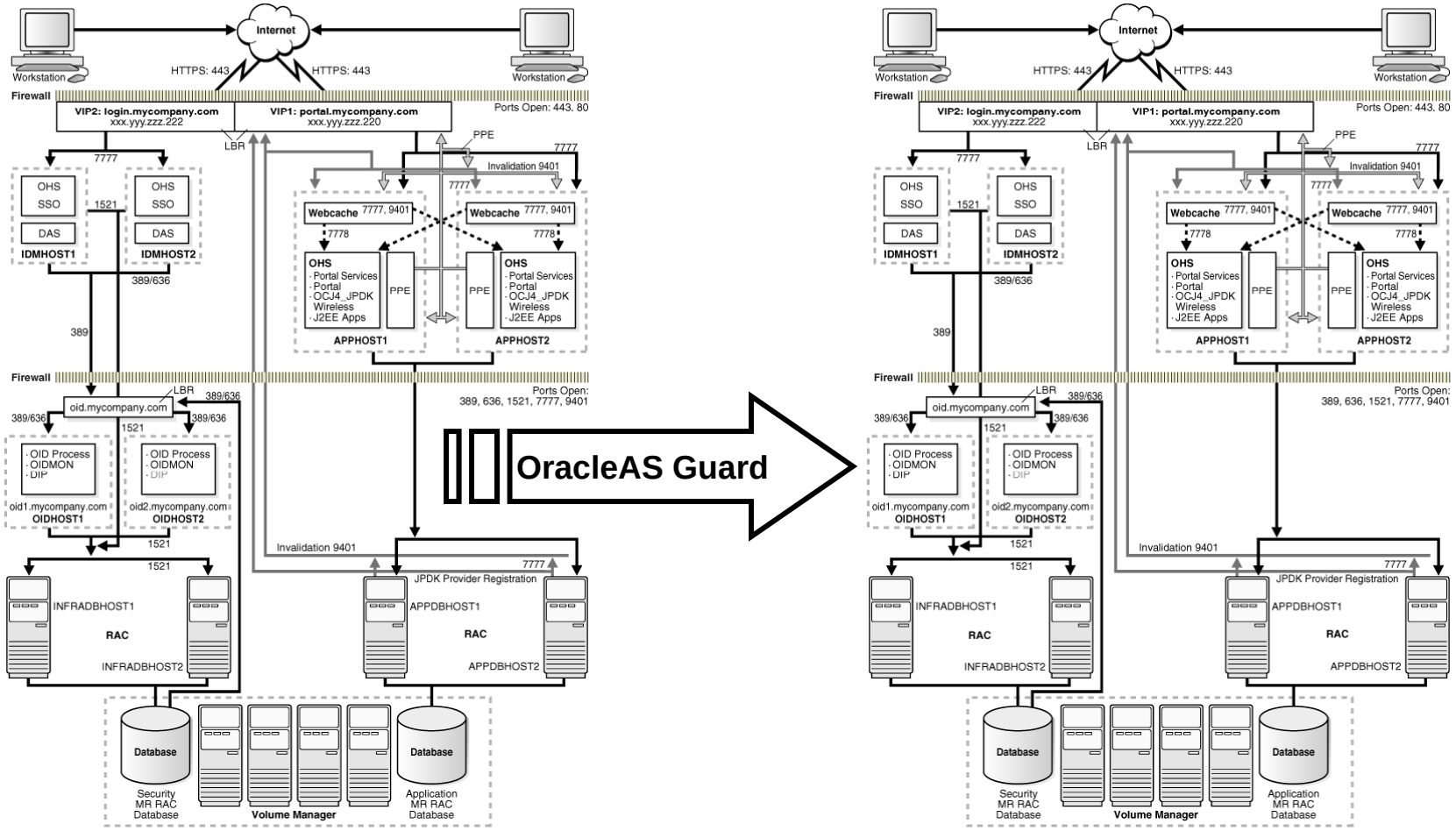
OracleAS Guard Architecture



Fusion Middleware Maximum Availability Architecture

- Configure ASGuard to synchronize farms on a regular basis (depending on your system´s lifecycle, it may be good to cron and synch every day at night)
- Relevant changes, such as deployment of new applications, should be synchronized as well (you can use deployment event to trigger an ASG synch)
- Bulk changes in Metadata (BPEL, Portal, etc) and Identity Management repositories should also trigger syncrhonization

So what does it look like when you put everything together?



Production Site **Standby Site**

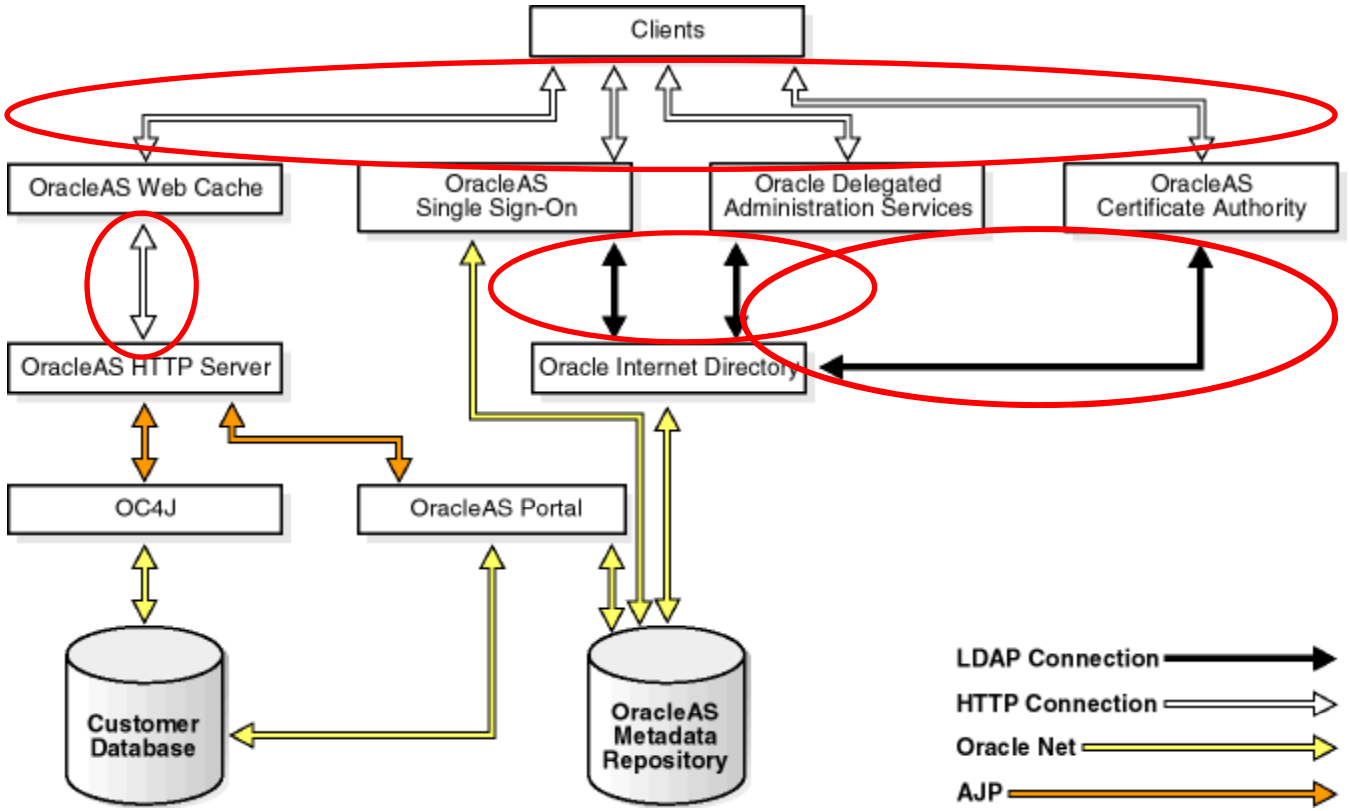
Fusion Middleware Maximum Availability Architecture Summary

- Key aspects for Maximum Availability in a FM deployment :
  - Process Management and death detection
  - Redundancy at process and session level
  - Proper Connection management
  - Keep an eye on things that you would’t expect to happen but that, unfortunately, occur (DISASTERS!)

Secure Sockets Layer (SSL)

Protocols in Fusion Middleware

= can use SSL



Oracle Wallets

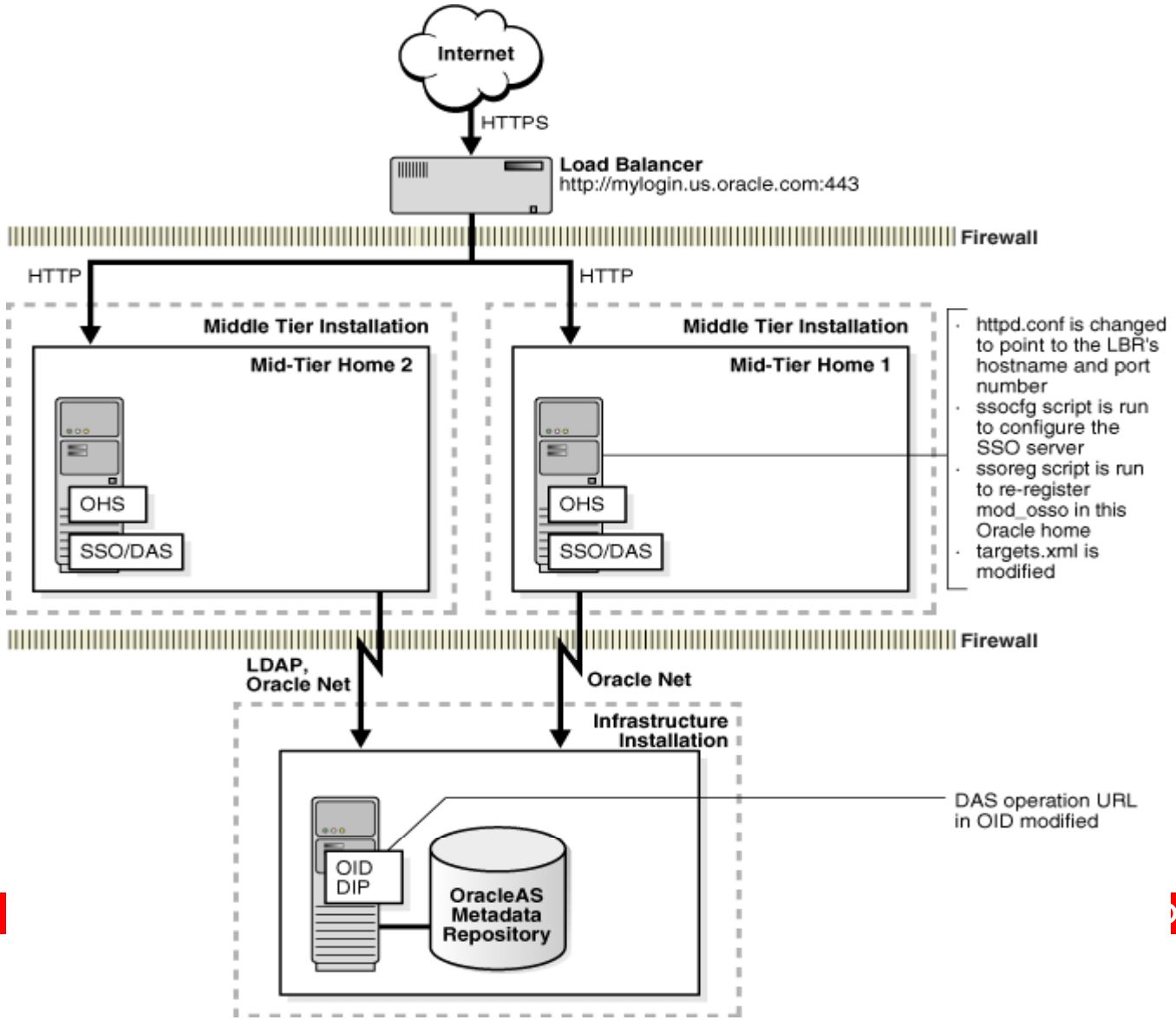
- For Oracle components (WebCache, HTTP Server, OID), Oracle Wallet is where your SSL certificate (and private key) is stored
- You must create and maintain the wallet(s)
- Wallet can be shared/copied between nodes. Certificates are based on hostname used by client, not the actual server hostname (significant in load balanced configurations)
- Oracle Wallet Manager (owm) is the tool used to manage wallets through OAS 10g.

SSL Configuration Options

- Browser -> (HTTPS) -> Load Balancer -> (HTTP) -> WebCache -> (HTTP) -> HTTP Server
- Browser -> (HTTPS) -> WebCache -> (HTTP) -> HTTP Server
- Browser -> (HTTPS) -> WebCache -> (HTTPS) -> HTTP Server
- Browser -> (HTTPS) -> HTTP Server

Common Configuration Scenarios

SSL Termination at LB for SSO/DAS

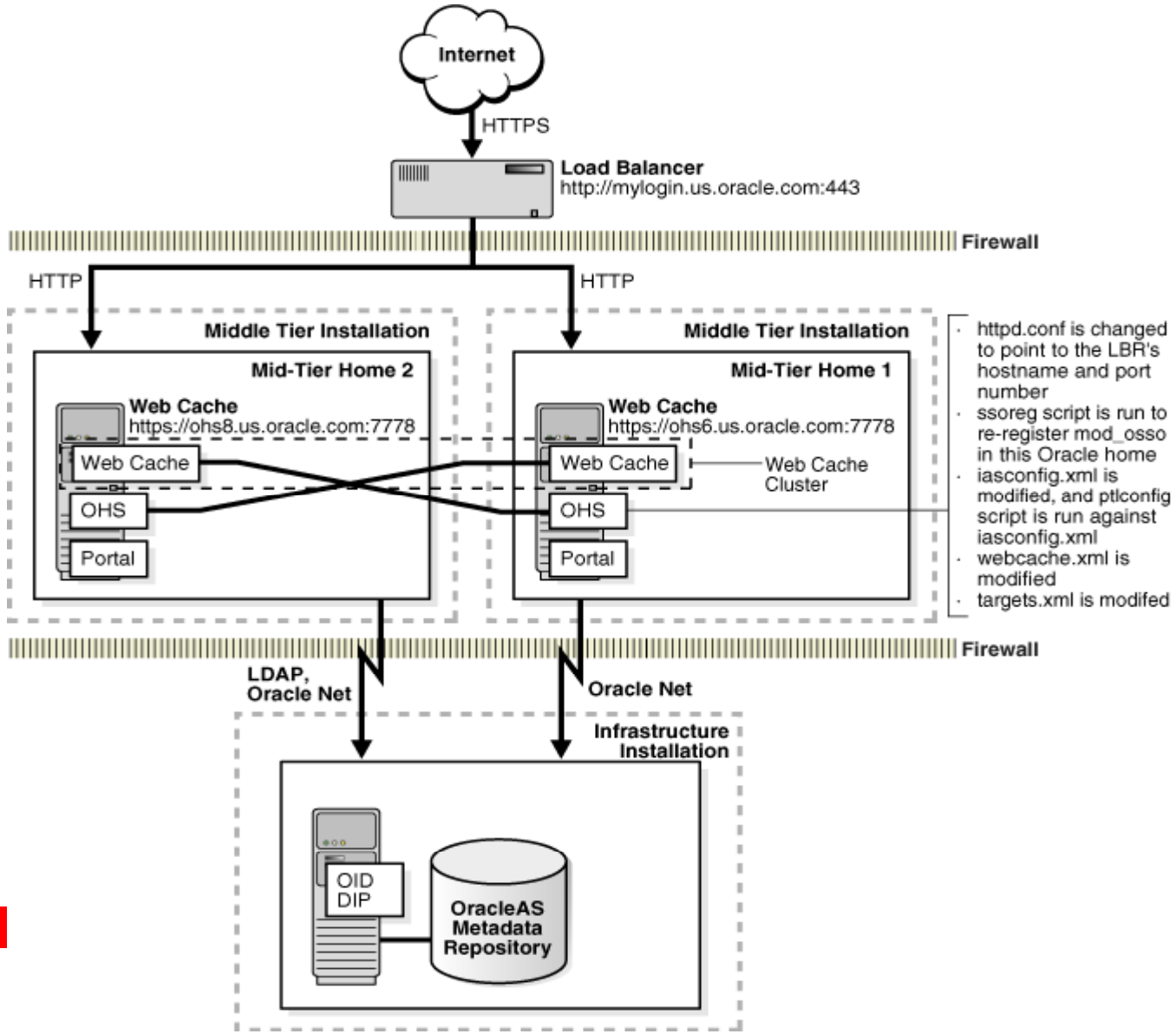


Common Configuration Scenarios
SSL Termination at LB for SSO/DAS

- No wallets needed
- Configure ServerName on HTTP Server in each infra httpd.conf
- Run ssocfg.sh to tell SSO its new protocol, name, port number
- Configure SimulateHttps on HTTP Servers
- Modify OID entry for orcldasurlbase for proper URL
- Re-register all partner applications (Portal, mod_osso's)

Common Configuration Scenarios

SSL Termination at LB for Portal

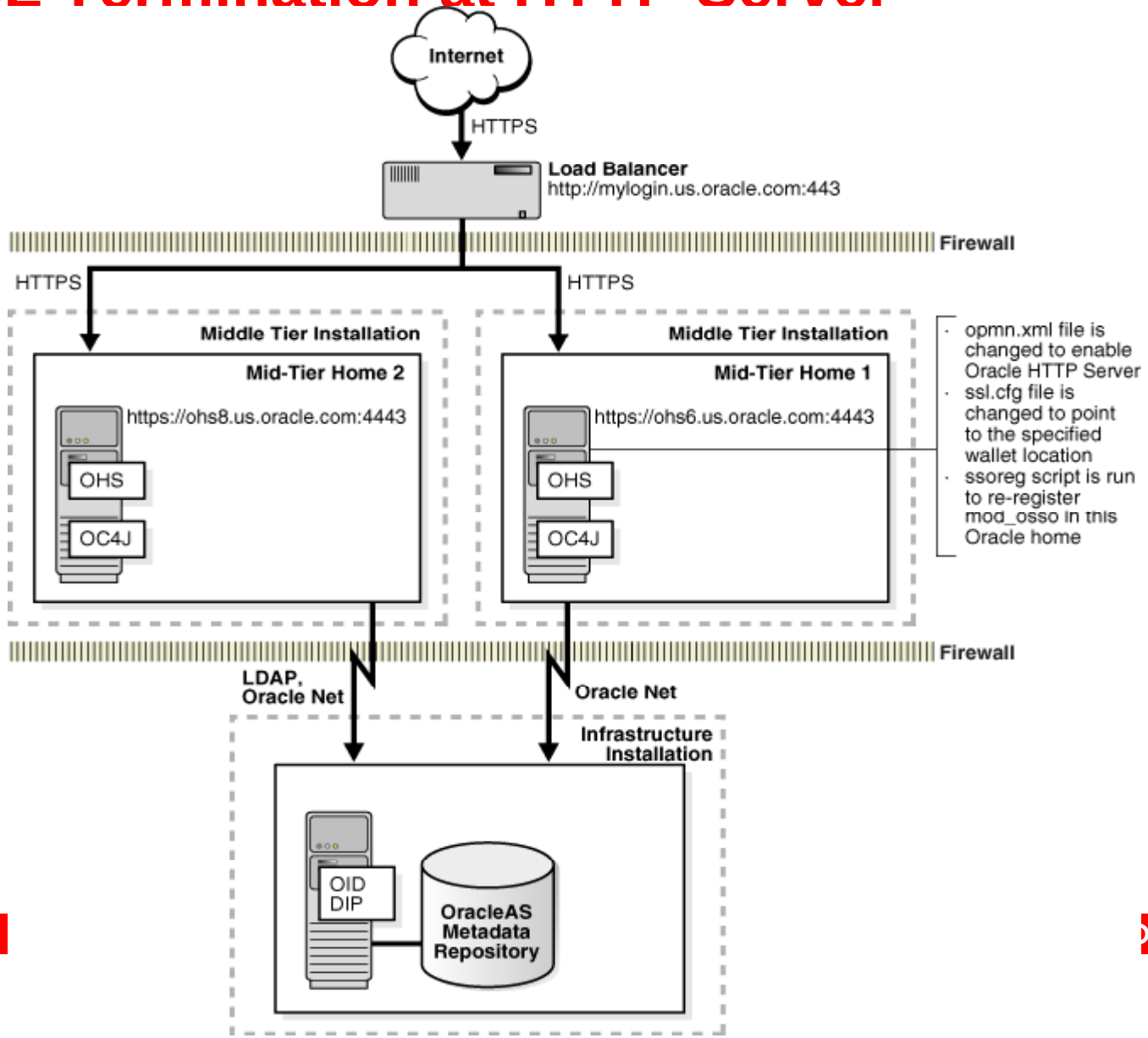


Common Configuration Scenarios
SSL Termination at LB for Portal

- No wallets needed
- Configure ServerName on HTTP Server in each midtier httpd.conf
- Modify iasconfig.xml and run ptlconfig to tell Portal its new protocol, name, port number
- Configure SimulateHttps on HTTP Servers
- Modify webcache configuration for new site definitions
- Re-register Portal and mod_osso partner applications for each midtier

Common Configuration Scenarios

SSL Termination at HTTP Server

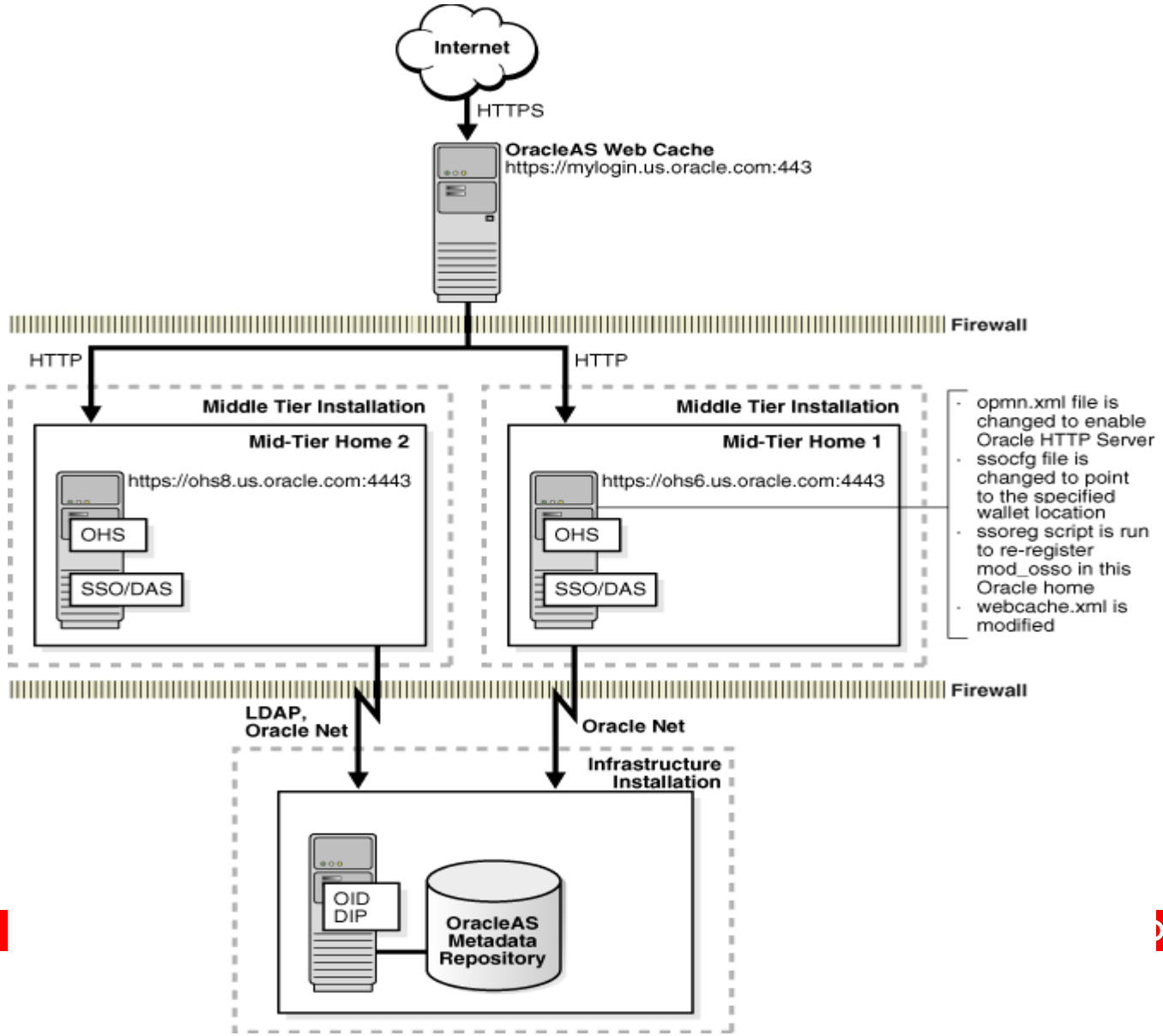


Common Configuration Scenarios
SSL Termination at HTTP Server

- HTTP Server wallet necessary
- Configure ServerName on HTTP Server in each midtier httpd.conf
- Modify opmn.xml to start HTTP Server in SSL mode
- Re-register mod_osso partner application for each midtier

Common Configuration Scenarios

SSL Termination at WebCache



Common Configuration Scenarios
SSL Termination at WebCache

- WebCache wallet necessary
- Configure ServerName on HTTP Server in each midtier httpd.conf
- Configure WebCache for additional site definitions
- Add listening port on WebCache
- Configure SimulateHttps on HTTP Servers
- Re-register mod_osso partner application for each midtier

Rewriting URLs

Rewriting URLs

- mod_rewrite is the module employed
- RewriteEngine, RewriteCond RewriteRule are the main directives used
- RewriteRule has flags
  - R – external Redirect (302 to browser)
  - L – Last rule, stop processing
  - PT – Pass Through: hand the results of this rule to other modules for further processing, response
- Rewrite* directives are not inherited by VirtualHost containers like some other directives (**`RewriteEngine on`** & **`RewriteOptions inherit`** will help)

Rewriting URLs
An Example

```
1) Alias /tpsredirect "/web/dannorris.com/tps"
2) RewriteRule ^/$ https://tps.dannorris.com/tpsredirect [R,L]
3) RewriteCond %{REQUEST_URI} !^/tpsredirect
4) RewriteCond %{REQUEST_URI} !^/tps/
5) RewriteRule ^/(.*) /tps/$1 [PT]
```

- Line 1 is referenced later
- Line 2 will redirect any requests for the root page of the site to the URL shown. This will be a 302 redirect [R] and if matched, this is the last rule processed [L].
- Line 3 is true if the request does not start with /tpsredirect
- Line 4 is true if the request does not start with /tps and is combined with the result of Line 3 (AND)
- Line 5 is always matched, but only used if 3 and 4 are true and prefixes the request with /tps

Concluding Tips & Tricks

Tips & Tricks

1. When debugging, eliminate all unnecessary components
2. Keep good notes and/or logs of every configuration change, you may want to go back and there are lots of moving parts (see #1)
3. Prepare test plans in advance, know what is necessary and what things are optional (browsers, OS platforms, load tests, etc)

Tips & Tricks

4. Use a whiteboard when too many components are involved. Sometimes drawing it the old fashioned way exposes the problem. If you can't draw the architecture, learn more until you can.
5. What's your best lesson learned or tip?

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.


QUESTIONS
ANSWERS


ORACLE