TIPS AND TECHNIQUES FOR DEPLOYING A SCALABLE PORTAL

Jason Pepper, Oracle Corporation

EXECUTIVE OVERVIEW

For a portal to be successful, it must be able to deliver information to its intended audience in a timely manner. The major advantage of a portal is its ability to bring together content and applications from disparate sources. However, if the portal, as a focal point for information access, adds too much of its own performance overhead, users will quickly learn to bypass it.

Good performance results from an architecture that can take full advantage of available hardware resources and scale with those resources to handle increasing user loads. Oracle Application Server Portal (OracleAS Portal) is the only portal product that meets all the requirements for scaling performance to handle any size audience.

At the heart of OracleAS Portal is its inherently scalable and distributed platform, consisting of the Oracle Application Server (OracleAS) and the Oracle database. That architecture includes a unique approach to caching both static and dynamic Web content that effectively balances the tradeoffs between good performance and timely, up-to-the-minute, personalized access to information.

INTRODUCTION

Web portals have emerged as the preferred way to bring together all your information resources. By using a Web portal, your organization's employees, customers, suppliers, business partners, and other interested parties can have a customized, integrated, personalized, and secure view of the documents, content, and applications with which they need to interact.

But with this improved access to information comes a challenge: in order to meet the increased demand for customized information through a central point, the portal infrastructure must be highly scalable. Scalability means that the portal will be able to handle increasing user loads without creating unacceptable response times. A scalable portal will satisfy the following requirements:

- The underlying architecture must be cross-platform, allowing the portal to take advantage of the full range of available hardware resources. Portal administrators must be able to choose the best platform for each portal component, allowing optimization of both performance and total cost of ownership.
- The architecture must support load distribution and parallel execution of portal components across multiple servers.
- The portal must implement intelligent caching of dynamically generated portal pages. Intelligent caching ensures that information remains fresh and timely while minimizing the cost of regenerating content from databases and back-end services. Intelligent caching supports user-level customizations and is also modular, so that entire pages or just page components can be stored in a cache and refreshed as necessary. Finally, intelligent caching must also protect security by ensuring that cached content is only accessible by authorized users.

OracleAS Portal is the only portal product that meets all three of these essential scalability requirements. As described in this document, OracleAS Portal combines a scalable, cross-platform, and distributed architecture with an integrated caching capability that is unrivalled by any other portal product. OracleAS Portal can thus serve any amount of content to any size of audience, without fear of running into scalability limits.

This paper describes the process used by OracleAS Portal to generate, assemble, and cache Portal content and will show real world examples based on Oracle's own employee portal – My Oracle

WHAT IS MY.ORACLE.COM ?

My.Oracle.Com (My Oracle) (*http://my.oracle.com*) is an Internet portal that provides business users, including both Oracle customers and employees, with a personalized, single entry point to essential applications and information. The site creates a destination to access all of Oracle's online services and leading business information sources, including more than thirty other global content providers.

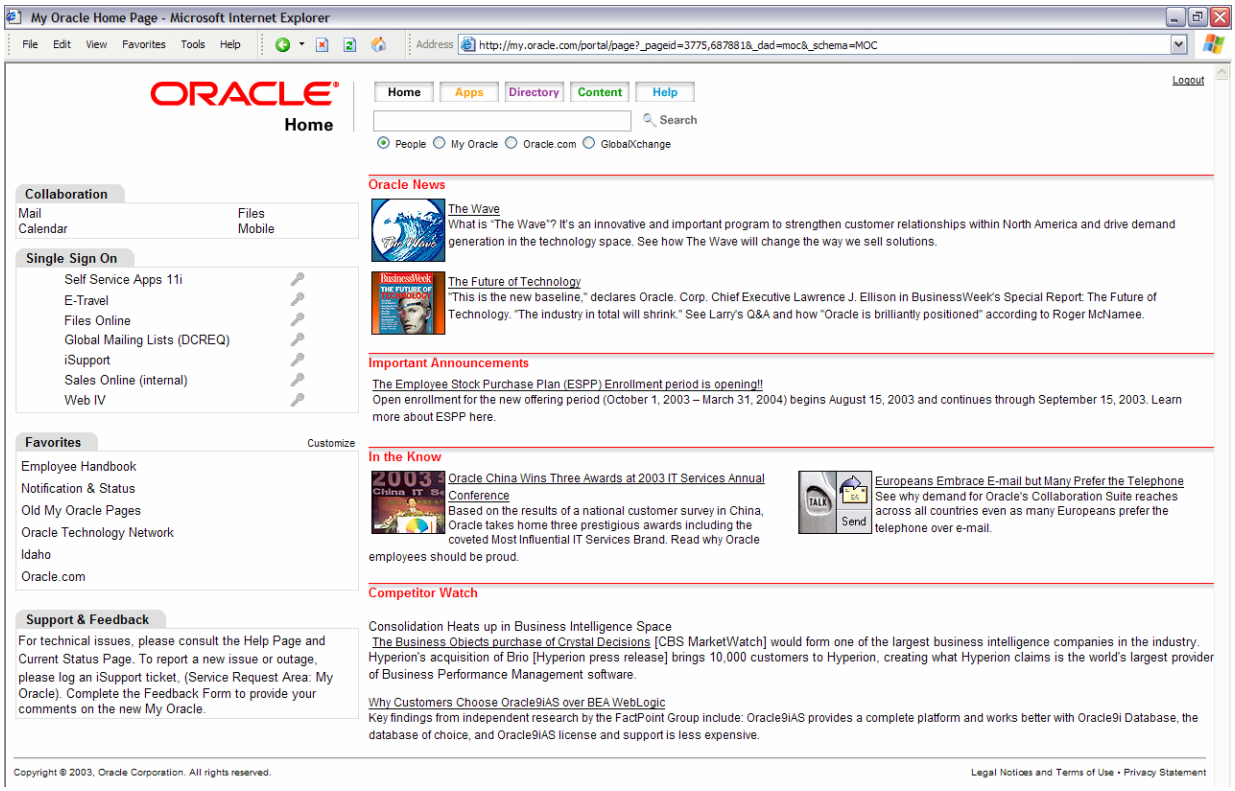

Figure 1: View of My Oracle's Default Home Page

A SCALABLE ARCHITECTURE FOR PAGE GENERATION AND ASSEMBLY

OracleAS Portal provides a scalable architecture for dynamically generating customized pages that can meet the performance requirements of any size of portal, including departmental, regional, enterprise-wide, and Internet deployments. The deployment model supports both single-box and multi-tier configurations on a broad set of hardware platforms and operating systems.

OracleAS Portal is developed with the facilities of the Oracle Application Server and the Oracle database, and therefore benefits from the inherent scalability of these underlying products. In particular, OracleAS Portal combines unique OracleAS scalability and performance features like content caching, load balancing, and resource pooling with the ability to distribute portal installations to meet the unique requirements of a scalable portal platform.

The following sections describe the pieces of the OracleAS Portal architecture that are used in page generation and assembly, as shown in the following diagram (Figure 2).

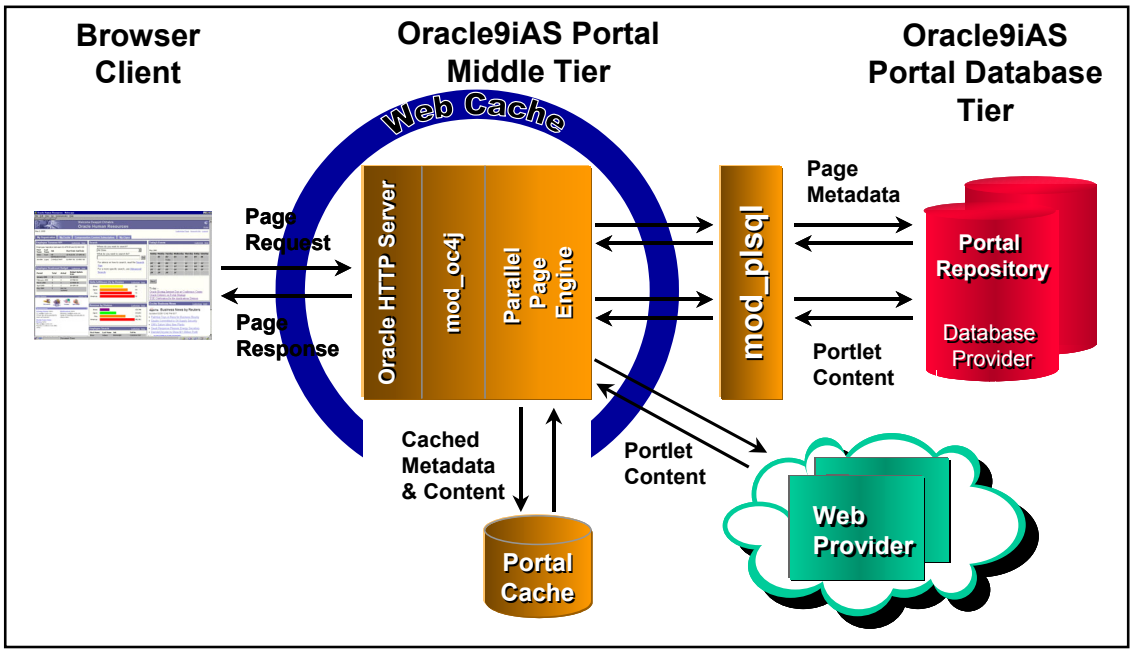

Figure 2: OracleAS Portal Logical Architecture

MY ORACLE TECHNICAL OVERVIEW

Throughout this paper we will be providing real world examples of configurations based on My Oracle. This section shows an overall view of the physical implementation.

There are multiple portals hosted within the same architectural layout, including portal.oracle.com, portalcenter.oracle.com and oraclepartnernetwork.oracle.com. The infrastructure described here has been designed to handle a very large amount of traffic. Each of these services have large user populations.

The current architecture as pictured in Figure 3 shows three distinct elements to the my.oracle.com setup: my.oracle.com, login.oracle.com, and provider.oracle.com

- my.oracle.com (My Oracle) is the physical portal infrastructure that handles the requests for pages and passes requests for logins and portlets to the other components when needed.
- login.oracle.com is the centralized SSO Server used by My Oracle and other services within the Oracle infrastructure. login.oracle.com provides a common login service for all Oracle Web sites.
- provider.oracle.com is a separate entity for dealing with requests for content from providers registered within the My Oracle repository.

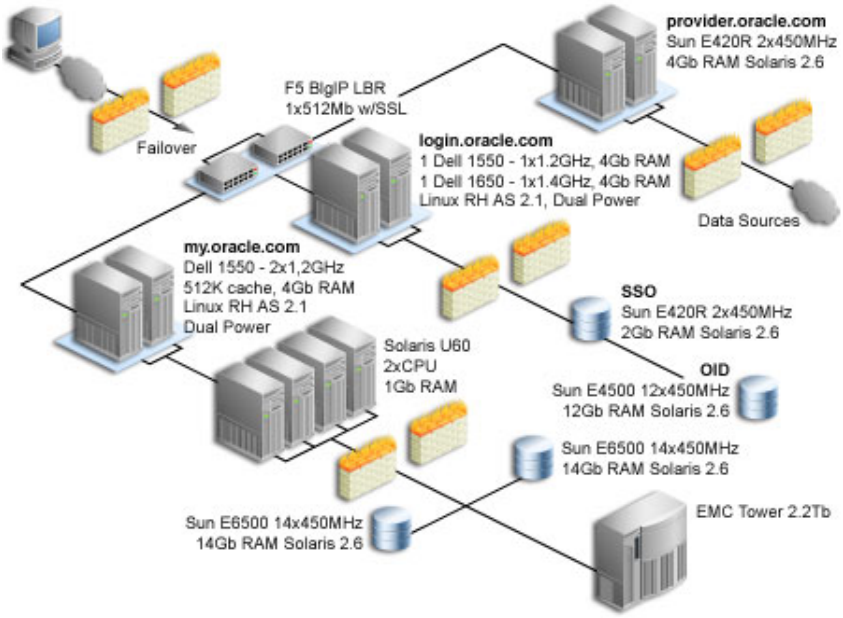

Figure 3: My Oracle Architecture

ORACLEAS WEB CACHE

OracleAS Web Cache has been closely integrated with OracleAS Portal to improve OracleAS Portal's overall scalability, performance and availability. Unlike legacy cache servers, which only handle static data, OracleAS Web Cache combines caching, compression and assembly technologies to accelerate the delivery of both static and dynamically generated OracleAS Portal content. OracleAS Web Cache also provides back-end Web server load balancing, failover and surge protection features, which ensure blazing performance and rock-solid up-time. With OracleAS Web Cache, OracleAS Portal can now serve rich content faster, to more users, using fewer computing resources than ever before.

In OracleAS Portal 10*g* release, OracleAS Portal functions as a Web Cache origin server to take advantage of the following Web Cache features:

- *Caching of dynamically generated, user-specific page and portlet content:* Web Cache's understanding of the contents of HTTP headers—including cookies—allows it to cache the personalized content that is uniquely maintained and generated for each OracleAS Portal user.
- *Fine-grained cache control:* Administrator-defined cacheability rules allow caching and routing decisions based on HTTP header information, including cookies. OracleAS Portal ships with a number of pre-defined caching rules that ensure optimal use of Web Cache capabilities. Web Cache controls have been built into the OracleAS Portal user interface and can also be specified by providers through the PDK.
- *Invalidation-based caching:* Cache invalidation can be based on an expiration period, or by having an invalidation message sent by an application when content needs to be refreshed. OracleAS Portal automatically sets the correct caching policy for every object managed in the Portal Repository, and ensures that invalidation messages are sent to Web Cache when content is modified or security privileges are altered.
- *Layer 7 load balancing and failover detection:* Web Cache can be used to balance the load between multiple OracleAS Portal middle-tier servers and/or providers in a cluster. Cache misses are directed to the most available, highest-performing middle-tier server or provider in a server farm. Cookies can be used to maintain persistent, or "sticky", connections to a specific server when necessary to preserve state.

- *Performance assurance and surge protection:* Web Cache offers patent-pending techniques to guarantee site performance and scalability, even when Web server loads surpass capacity levels. A surge protection mechanism detects system overload conditions, providing a crucial buffer against traffic spikes and denial-of-service attacks.

WEB CACHE DEPLOYMENT

When OracleAS Portal is installed, a Web Cache instance is automatically created with pre-defined cacheability rules and cache configuration settings. Portal sites can choose from the following deployment options:

- *Co-located:* Web Cache runs on the same physical server as the OracleAS Portal middle-tier. This configuration is appropriate for smaller, low-volume sites where the scalability of the middle-tier is not a concern.
- *Dedicated:* Web Cache is deployed on a dedicated server that sits in front of one or more OracleAS Portal middle-tier servers. Dedicated deployments are usually preferable to co-located, as there is no risk of resource contention with other server processes. Web Cache performs excellently on commodity hardware, so a dedicated deployment need not be costly in terms of hardware expenditure.

For very high-volume sites, and to avoid a single point of failure, two or more nodes running OracleAS Web Cache may be deployed behind a third-party network load-balancing device.

If you have multiple deployments of OracleAS Portal, each OracleAS Portal site can have its own Web Cache server, or one or more sites can share a single Web Cache. Similarly, a Web Provider can share a Web Cache with an OracleAS Portal site, or a dedicated Web Cache can be deployed in front of the Web server that hosts the Web Provider. The decision to deploy more than one Web Cache will depend on the following factors:

- *Performance:* Multiple Web Caches will distribute the request and cache management load and reduce the network latency between a cache and its origin server, helping to eliminate performance bottlenecks.
- *Administration:* Different sites may have different requirements for cacheability, such as maximum cache expiration time.
- *Firewall Issues:* Web Cache uses special ports (i.e. not the standard HTTP ports) for invalidation messages and administrative functions. This usually means that a firewall can't be used to separate Web Cache from its origin servers.
- *Access Control Changes:* Changes to access control privileges invalidate Web Cache entries for the affected objects and users. However, the invalidation messages are only sent to the local Web Cache. If Web Cache is shared across installations, cached content will be invalidated for all. But if each installation has a dedicated Web Cache, only the local cache will be invalidated when privileges are changed. Although rare, this could result in users seeing content for which they were previously authorized, but for which they have lost their access privileges.

Web Cache caches all content in memory. To avoid swapping documents in and out of the cache, it is crucial to configure enough memory for the cache. The amount of memory (maximum cache size) for OracleAS Web Cache should be set to at least 256 MB. By default, the maximum cache size is set to 500 MB, which is sufficient for most caches.

A browser-based console, OracleAS Web Cache Manager, is used to administer all aspects of Web Cache, including configuration of cacheability and load-balancing rules, security, manual and automated invalidation, monitoring, and logging.

For more information on Web Cache, please refer to the Web Cache documentation. A number of technical white papers on Web Cache are also available at http://otn.oracle.com/products/ias/web_cache/content.html.

MY ORACLE WEB CACHE CLUSTER

There are two OracleAS Web Cache machines configured as a single cluster. The machines are installed with OracleAS Web Cache 9.0.2.2. The repository now issues dynamic caching and invalidation messages to the OracleAS Web Cache clusters to ensure that the most up-to-date version of the content is stored in OracleAS Web Cache.

OracleAS Web Cache Machine Specifications	
Model	Dell 1550 Rackmount
Number of CPUs	2
CPU Speed	1.2GHz
Total Physical RAM	4096 MB
Power Supply	Dual

PORTAL MIDDLE TIER

The OracleAS Portal middle tier handles all requests from users, forwards those requests to the appropriate Portal database or provider, assembles Portal pages, and manages some of the caching of Portal content. The middle tier consists of:

- Oracle HTTP Server, powered by Apache.
- mod_plsql, which redirects HTTP requests to PL/SQL code running in a Portal database and provides a native connection to the Oracle database. Mod_plsql also provides pooling of database connections for efficient management of database resources.
- mod_oc4j, which routes all servlet requests to the OracleAS Containers for J2EE (OC4J) servlet engine. OC4J offers a number of performance characteristics that enable it to scale from very small sites running on inexpensive, low end single processor machines to very large sites running on high end SMP clusters.
- Parallel Page Engine.
- portal cache.

ORACLE HTTP SERVER

The middle tier includes an Oracle HTTP listener that handles all the incoming HTTP requests to the portal. If the incoming request asks to display a portal page, the listener hands the request to OC4J running the Parallel Page Engine (PPE). The PPE splits up all the portlet requests that make up the page and sends them off in parallel to wherever the portlets are served from. If the request asks to execute a database portlet or a PL/SQL procedure, the listener hands it to mod_plsql to communicate to the database. As each of the requests comes back, the Parallel Page Engine reassembles the page and returns it to the user. If one of the portlets is too slow in returning, the Parallel Page Engine returns the rest of the page once the timeout duration specified in the provider has elapsed.

PARALLEL PAGE ENGINE

The Parallel Page Engine (PPE) runs as a *stateless* servlet. This means that the PPE itself does not maintain any stateful information about user requests. However, the PPE uses cookies to track state across requests from the same user session. This state is used to check the authentication status of the current user (i.e., logged in or not) and to retrieve user-specific entries from the portal cache and Web Cache. When balancing load across Portal middle-tier instances, it is important that sticky routing (also called server affinity) be used to ensure that each request from the same user is routed to the same instance of the PPE to avoid cache misses in the portal cache.

For the My Oracle deployment, no external service can ever call the middle tiers directly. They go through a virtual IP or a URL that exists only in the load-balancing router (LBR). The LBR routes the request directly to the Web Cache cluster, which in turn passes the request on to a middle tier. This provides a higher level of security and the ability to control specific types of hacker attacks in one central location.

FEATURES OF THE MIDDLE TIER

The following sections summarize the important features of the middle tier:

PORTAL CACHE

The portal cache is the persistence component of the My Oracle caching armory and is a key component in supporting very high request rates. The portal cache uses the file system on the middle tier to store fully assembled pages, as well as page definitions and rendered content from individual portlets. Retrieving content from the cache is much faster than having the database and providers regenerate it and the PPE re-assemble it into pages for every request. So if the page definition hasn't changed from the last time the user requested it, then the cached information will be served up very quickly. Furthermore, if the portlet content hasn't changed from the last time the user requested it, the portal cache serves up the already assembled page.

The portal cache entries can be configured on a page-by-page or portlet-by-portlet basis to check if a refresh is required (*validation-based caching*) or to refresh periodically (*expiry-based caching*). My Oracle uses expiry-based page level caching exclusively, with a refresh period of 10-30 minutes depending on the page. This means that content will be never be older than 30 minutes, though certain events, such as a change to the page definition, can cause the cache to refresh before the timeout period expires.

With the introduction of the 9.0.2.6 repository, My Oracle architects will be able to take advantage of *invalidation-based caching*, which allows the repository to send an invalidation SOAP/HTTP message to the cache when and only when the content is no longer valid. This means that we reduce the overhead of a validation-based cache 'ping' which occurs every time the content is requested to see if it is still valid. We also reduce the reliance on expiry-based caching, which while effective and inexpensive in terms of processing overhead, can be a little too coarse-grained for some of the more dynamic page caching requirements. As My Oracle continues to evolve, developers of providers will be encouraged to use the invalidation-based caching mechanism as it will allow them to control exactly when their content is refreshed in the cache and leave it the page developers to worry about handling page level performance.

OC4J

Each machine runs two instances of OC4J for redundancy purposes. The OracleAS Process Management Service (OPMN) will detect and correct any OC4J process death. Running two separate instances allows the middle tier to continue to handle requests while the dead process is restarted should that event occur. The OC4J count is managed using the parameter numProcs, which can be found in $ORACLE_HOME/opmn/conf/opmn.xml

```
<oc4j maxRetry="3" instanceName="home" numProcs="2">
  <config-file path="/disk2/oracle/midtier/j2ee/home/config/server.xml"/>
  <oc4j-option value="-properties"/>
  <port ajp="3000-3100" jms="3201-3300" rmi="3101-3200"/>
  <environment>
    <prop name="LD_LIBRARY_PATH" value="/disk2/oracle/midtier/lib"/>
  </environment>
</oc4j>
```

Importantly, the middle tier machines are not installed in a cluster as this promotes unnecessary chatting between machines. The OracleAS Web Cache cluster will hand requests on to the middle tier machines as they are received. The sticky routing discussed earlier will ensure that the request is returned to the correct middle tier. The middle tiers do not use the OC4J clustering available in 9.0.2.

To ensure that we don't suffer from memory leaks and orphan process problems, one of the four middle tier machines is restarted each day using a CRON script.

Middle Tier Machine Specifications	
Model	Sun Ultra 60
Number of CPUs	2
CPU Speed	450MHz
Total Physical RAM	1024 MB

SCALING THE MIDDLE TIER

Any OracleAS Portal system can contain one or more instances of the middle tier. The middle tier can be co-located on the same server machine as a Portal database, or on dedicated server(s). Generally speaking, it is usually preferable to run each middle tier instance on its own server. Inexpensive middle tier servers can be easily added to increase system throughput and to handle higher concurrent loads. The load-balancing capabilities of OracleAS Web Cache can be used to distribute requests over the different instances. A third party load-balancing product, such as Cisco Local Director, BigIP, or Alteon, can also be used.

THE MY ORACLE LOAD BALANCED FRONT-END

All traffic from the public internet passes through the Oracle firewall and is received and responded to by a BIG-IP load-balancing router (LBR) from F5 Networks Inc. (www.f5.com) to load-balance requests among the Web servers located in the middle tier of the De-militarized Zone (DMZ). To ensure high availability and fault tolerance, a second BIG-IP router is in place and active should the primary router fail.

In computer networks, a DMZ is a computer host or small network inserted as a "neutral zone" between a company's private network and the outside public network. It prevents outside users from getting direct access to a server that has company data. A DMZ is an optional and more secure approach to a single firewall and effectively acts as a proxy server as well.

In a typical DMZ configuration for a small company, a separate computer (or *host* in network terms) receives requests from users within the private network for access to Web sites or other companies accessible on the public network. The DMZ host then initiates sessions for these requests on the public network. However, the DMZ host is not able to initiate a session back into the private network. It can only forward packets that have already been requested.

The LBR is configured to add its own cookie to any request for content, to ensure that a subsequent request for content is directed to the same middle tier that served the previous request. This is called server affinity, and it is necessary, because we maintain four separate file caches (see the section later on the Portal Cache) on each of the middle tier boxes and routing a second and subsequent request to a different box could result in previously fetched content being refetched, leading to performance loss

PORTAL DATABASE

Each installation of the Portal middle tier is matched to a single Portal database. The Portal database is an instance of the Oracle database in which OracleAS Portal and database providers are installed. Each Portal database includes Portal application code (written in PL/SQL stored procedures), a portal repository (the content and metadata that are managed by OracleAS Portal), and the code for database providers, including the database portlets (forms, reports, charts, and other components) defined with OracleAS Portal's application building tools. The data that is accessed by database portlets may also reside in the Portal database, or be located in remote databases.

The distributed OracleAS Portal architecture allows multiple Portal databases to be federated. Databases communicate among themselves to allow multiple physical instances to appear as a single logical database to the end user. Any database that participates in the distributed OracleAS Portal model can access any provider, and content from multiple databases can be aggregated on a single portal page. All databases and middle tiers in a distributed Portal system must share the same Single Sign-On Server.

A Portal system can be scaled by distributing load across multiple Portal databases. Each database will own its own set of content and applications. If a particular database is stressed, some of its load can be transferred to another database by exporting and importing content areas or providers. Of course, conventional database scaling techniques can also be used, including the addition of CPUs and memory and by taking advantage of Real Application Clusters.

MY ORACLE PORTAL DATABASE SERVER

The My Oracle Portal repository resides within the internal Oracle network behind the DMZ. This means that only the middle tier machines are allowed to make requests directly to it. If a request comes from anywhere else, it is denied.

Database Machine Specifications	
Model	Sun Enterprise 6500
Number of CPUs	14
CPU Speed	450MHz
Total Physical RAM	14336 Mb
Database Size	20 Gb
SGA	270 Mb
Database Version	9.0.1.3
Disk Configuration	EMC Tower (2.2Tb)

For high availability of the databases, Oracle Dataguard is installed and a hot standby database runs on a second, identical database server (this second server is also used for other applications, so only a portion is allocated to running the My Oracle standby database). The synchronization of the log files is intentionally delayed for a period of 3 hours. This way, if there ever is any corruption, the bad information is not replicated to the failover system.

For additional reliability, the file system is mirrored on the servers' EMC storage systems, and standard database backups are also taken. The backups are first written to disk, and then streamed to tape. This has less impact on database performance and allows the tape backup to proceed at its own pace.

The My Oracle repository has been upgraded from 3.0.9 to 9.0.2.6 as part of the testing of the 3.0.9 to 9.0.2. The processes learned and the issues encountered will be included in the upgrade release (9.0.2.6) currently available in controlled release. See the upgrades section on PortalCenter for further details.

SINGLE SIGN ON

OracleAS Single Sign-On (SSO) technology provides single sign-on for Web users. It is designed to work in an environment such as that provided by Oracle Application Server, where multiple Web-based applications are accessible through a portal. The Oracle strategy for SSO encompasses a variety of technologies. For the growing field of Web-based applications, Oracle has developed an SSO framework, OracleAS Single Sign-On, which is specifically designed to provide Web SSO.

The OracleAS Single Sign-On approach has a number of benefits. It provides a framework for secure SSO from browser clients to Web-based applications, including Oracle applications and tools, through standard protocols. It supports both partner applications, which take full advantage of the SSO framework, as well as external applications for support of legacy and third-party products. Partner applications work within the SSO framework and rely on the SSO service for authentication of users; external applications continue to use their own user names and passwords. The OracleAS Single Sign-On approach is based on cookies, which are created both by partner applications and by a centralized OracleAS Single Sign-On Server.

LOGIN.ORACLE.COM ARCHITECTURE

The SSO solution for My Oracle runs on separate hardware as a separate logical entity called Login.Oracle.Com for the following reasons:

- Security for the SSO servers can be much tighter than that for the My Oracle and provider.oracle.com machines
- Administration of SSO servers should be separate from that of My Oracle. Oracle's internal Global IT division are solely responsible for the maintenance and management of the SSO architecture, and this was easier to achieve with this split domain approach.
- Tuning the SSO repository on a separate machine from the portal repository allows for separate tuning parameters to be set in line with the requirements of the SSO application profile.

SSO SERVER MIDDLE TIER

The SSO Server middle tier machines run the same configuration as the portal middle tier machines. Web servers were specifically added to service just the SSO Server as other groups outside My Oracle need to share the user repository. With these dedicated Web servers, the performance for the authentication service can be tuned and adjusted independently from the portal service. Again, if you were considering building a similar type of OracleAS Portal deployment, then this function could be shared with the same middle tier machines used to front-end the Portal database.

The SSO service is accessed via SSL. The SSL encryption is handled by an accelerator board in the LBR. This is important as the process of encrypting the outbound traffic will add an unreasonable overhead to the Web server; offloading the encryption to the LBR means that My Oracle can be configured to operate in full HTTP mode and rely on the LBR to deal with the encryption. Techniques for configuring these types of setups are discussed in the OracleAS Server Security Guide.

Each middletier is running

- Oracle HTTP Server
- mod_plsql 9.0.2.6
- OC4J 9.0.2

The middletiers are running the 9.0.2 versions of the listener and the Java environment. Unnecessary pieces of the default V2 environment are removed after the installation; these include DAV, OC4J_Demos, OC4J_Portal and OC4J_Wireless. For an in-depth definition of the components removed and further configuration, refer to the document titled 'Configuring a Secure Oracle Application Server Environment' on OTN.

A custom login page is used to enable the branding of the SSO Server look and feel. The login.oracle.com page is written using JSP, but this could just as easily have been written in Perl or PHP. Two instances of OC4J are running on each of the middle tiers for the same redundancy reasons discussed earlier.

There are two middle tier machines configured as follows:..

SSO Middle Tier Machine Specifications		
Model	Dell 1550 Rackmount	Dell 1650 Rackmount
Number of CPUs	2	2
CPU Speed	1.2GHz	1.4GHz
Total Physical RAM	4096 MB	4096 MB

SSO SERVER DATABASE

login.oracle.com uses the core security and authentication features of OracleAS 9.0.2. This means we use the SSO Server to pass the initial authentication requests for logging in to Oracle Internet Directory (OID) and to manage retrieval of user names and passwords for external applications that may be stored in the SSO schema.

Upon initial authentication, the user name and password submitted by the user are passed to the OID instance for an LDAP authentication. The clear text password entered by the user is run through a one-way encryption and the encrypted string is compared against that retained in the OID repository. If a match is made, then OID will return an authenticated token and the SSO may commence the login process. The SSO Server has a set of tables that it uses to manage users and passwords for external applications outside of the Oracle environment. All the Web sites run by Oracle utilize this OID authentication feature, which means that if you have an account on any of the Oracle Web sites today, you can log in to My Oracle.

OID Machine Specifications	
Model	Sun E4500
Number of CPUs	12
Processor Speed	450MHz
Total Physical RAM	12288 MB
Database Size	2 GB
SGA	270 MB
Database Version	9.0.1.3

SSO Machine Specifications	
Model	Sun E420R
Number of CPUs	2
Processor Speed	450MHz
Total Physical RAM	2048 MB
Database Size	0.5 GB
SGA	270 MB
Database Version	9.0.1.3
SSO Version	9.0.2.6

PROVIDER.ORACLE.COM ARCHITECTURE

provider.oracle.com is a separate installed architecture used to resolve requests for content from Web providers registered within the My Oracle repository. The use of this separated provider architecture allows for the scaling of component hardware according to specific request requirements, without having to worry about the trade off this may cause for other components of the portal architecture, as would be the case if the providers were running from within the My Oracle architecture. The distributed provider architecture also allows us to manage access and security independently from the My Oracle machinery. If, for example, a developer needs access to the provider.oracle.com architecture, it can be provided without having to compromise the security of the My Oracle or login.oracle.com architectures.

provider.oracle.com Machine Specifications	
Model	Sun E420R
Number of CPUs	2
CPU Speed	450MHz
Total Physical RAM	4096 Mb
Oracle HTTP Listener	2
Jserv Engines	4

There are many different portlets available to My Oracle from within the provider architecture. These primarily take the form of a piece of Java code utilizing the Portal PDK to import the portlet structure and provide its own piece of unique content. This content may be dynamic or static. For performance reasons, any content refresh is limited to a lifespan of 30 minutes, thus reducing the load on the portal infrastructure, cached content being much easier and quicker to serve up than dynamic content.

An example of a dynamic portlet is the ubiquitous stock quote portlet, where a request for a page will always result in a subsequent SOAP/HTTP call to the stock agent for an updated price quote.

An example of a static portlet is the file-based portlet, where content is registered through the normal provider.xml process, but each portlet is provided by a piece of file content that may be provided by a third party. Newsfeeds are a good example of this: we receive newsfeed information from various agencies as XML files, fed through to the provider machine, which will serve them up the next time the 30 minute content refresh time has elapsed.

ORACLE INTRANET

Oracle's online services (www.Oracle.com, Oracle Technology Network (OTN), and My Oracle) use a single user repository to authenticate users, called Oracle's online registration system. This system also handles registrations for the eBusiness Network, AppsNet, and Oracle's Partner Network. So it was key to leverage this large user population as My Oracle was rolled out. There are also a number of internal applications and content providers on the Oracle intranet. These providers are integrated into My Oracle, but are only accessible to Oracle employees. OracleAS Portal's flexible security architecture allows employees and customers to share the same portal while ensuring that each user only sees the information for which they have been authorized. Authorization privileges can be specified on each portal page, for each portlet, or by the applications exposed through the portlets.

RELIABILITY AND AVAILABILITY

As discussed above, all of the middle tier components have a redundant set of servers to ensure high availability. Oracle Enterprise Manager is used to monitor the system and issue alerts when anything goes wrong. In addition, some other utilities are used to check on system status.

t is planned for My Oracle to use Real Application Clusters (RAC) from the Oracle database. RAC is the evolution of the Oracle Parallel Server (OPS), offering the failover and load-balancing benefits of OPS with considerably less administrative complexity. This implementation of RAC will occur in the second half of 2003.

Every machine has at least two network interface cards (NICs) to ensure access to the machines is available for administrators at all times. One NIC is employed as the main interface; the other is used for doing online network backups. Both NICs are on separate switches and subnets. This setup is also exceptionally useful if a Denial of Service attack is suffered on the main NIC. Administration staff can access the target machine through the secondary NIC and deal with the problem.

Performance logging is switched on throughout the whole My Oracle architecture; mod_plsql, PPE, OC4J, Web Cache and DB logging are all utilized to give the fullest picture possible of the health of systems.

The performance logs are switched twice in a 24-hour period and these finalized logs are fetched by a specific log analysis system. The analysis system generates performance reports on a daily basis and retains seven days worth of data, approximately seven million rows of data. The data from these logs is then uploaded into an Oracle database for analysis. The steps for this are documented in the technical note 'Performance Monitoring with mod_plsql in Oracle9iAS Portal Release 2' and an overview of the types of reports available can be found in the technical note 'Object Access Reporting from the Performance Logs in Oracle9iAS Portal'

The outcome of the logs and some examples of the data they provide are shown in this paper.

The performance logs are analyzed to check for poorly performing pages, portlets and providers; however, there is also a failsafe monitoring system in place utilizing the monitoring capabilities of Oracle Enterprise Manager.

If an issue has arisen either via the performance logs or through database or system alerts, these alerts are emailed to My Oracle site administrators. Oracle's IT staff will be automatically paged and requested to either dial in or attend site to help resolve the problem.

Monitoring listener status is an automated process that issues the following series of ramped tests every five minutes to every middle tier machine.

- File request – a known file (JPG) is requested from My Oracle, it must be returned within a given time frame to pass this stage of the test.
- CBUF Test – the string 'htp.p?cbuf=test' is issued to the mod_plsql component of My Oracle. An HTML response of **`test`** should be received within a given performance threshold. Failure to do so would once again trigger a performance alert
- IsItWorking Servlet Test – OracleAS ships with a servlet called IsItWorking which is requested by the /servlet alias of Oracle HTTP Server. An HTML response should be received within a given performance threshold. Failure to do so would once again trigger a performance alert
- PPE Servlet Test – A page ID from the portal is requested by the parallel page engine servlet, a suitable response within a given time frame is expected.

If the first four tests are passed, then the tool moves on to performing an integrated test, executed via the LBR which consists of the following steps.

- Navigate to http://my.oracle.com
- Login
- Customize the screen to add a new portlet.
- View the new page.
- Customize the screen to remove the new portlet.
- View the new page.
- Logout.

If all the tests are passed, then the tool will sit idle for 5 minutes before starting again. If any of the steps fail, then a warning is issued and the testing steps are restarted. If the error reproduces on three occasions, then the alerting process mentioned above is put into place.

Internally there are other less specific tests that are performed on a regular basis. For example, we regularly run *iostat* and *vmstat* on our unix servers. During one of these runs of iostat we noticed an abnormally high value for `asvc_t`

```
                extended device statistics
 r/s    w/s   Mr/s   Mw/s wait actv wsvc_t asvc_t  %w  %b device
40.3    1.0    0.5    0.0  0.4  0.2   10.3    4.5   1  16 c0t0d0
40.3    1.0    0.5    0.0  0.4  0.2   10.3    4.5   1  16 c0t0d0s0
 0.0    0.0    0.0    0.0  0.0  0.0    0.0   24.2   0   0 c0t0d0s1
 0.0    0.0    0.0    0.0  0.0  0.0    0.0    0.0   0   0 gandalf
```

This value is defined as the average service time for active transactions in milliseconds. In our example we can see that this is a relatively low 4.5 ms, however in live run we were receiving values in excess of 1000ms for a given disk that was servicing the news feed provider. Upon further examination it transpired that for each request of the news feed portlet, the code was reading the news feed file that we receive from the news providers even if the file had not changed. Following this determination, the portlet was re-architected so that it read the news feed file once and retained the content in a memory structure. Thereafter, each request for the news feed portlet would merely validate the timestamp of the news feed file against that of the one held in memory: if the two matched, it was OK to deliver the content from memory.

This simple architectural change halved the response times for the news feed portlet, thereby speeding up the response time for each of the pages that had an instance of the news feed portlet defined.

CONFIGURATION SETTINGS

The configuration of the components within OracleAS for My Oracle are generally left at the defaults they ship with, however some components are either shut down or removed for security purposes while others are tuned for performance reasons.

For a single middle tier on My Oracle, the following setting are dependent on the site usage, site content, and modes of caching used by content providers. This can only be used as a reference, and the actual settings for a site have to be determined separately on a per-site basis

My Oracle Configuration for Oracle HTTP Server (OHS) (httpd.conf)

- KeepAlive Off
- MinSpareServers 30
- MaxSpareServers 100
- StartServers 30
- MaxClients 250
- MaxRequestsPerChild 2000

You can learn more about these configurations options in the Oracle Application Server Performance Guide on OTN.

My Oracle configuration for mod_plsql (dads.conf)

- DAD level NLS_LANG is configured to match the database NLS_LANG, avoiding unnecessary/unsupported requirements to have charset conversions for response being fetched through mod_plsql
    - PlsqlNLSLanguage AMERICAN_AMERICA.UTF8
- The same setting is done for the SSO DAD as well
- PlsqlIdleSessionCleanupInterval is defaulted to 15 minutes

Database TNS Listener is configured for DEDICATED connections. MTS mode has additional overheads, which we wish to avoid. If your portal has Multi-Threaded Server (MTS) enabled MTS, you should try running without it to see if a performance improvement is achieved. Typically there is little need to use MTS with the connection pooling and reuse model of mod_plsql, which you can learn more about in the OracleAS Portal Architecture white paper on PortalCenter.

As with all settings, it is advisable to check for yourself if MTS is running or not. Oracle Support regularly encounter situations where you may think it is disabled, when in fact MTS is enabled.

You can learn more about how to configure and tune MTS in the Oracle Administrators Guide and the SQL*Net Administrators Guide.

If you know exactly how to tune and configure MTS, then it can have some advantages. Low access sites/DAD can use fewer DB server processes.

In the regular model, if you have 50 HTTPD daemons, then you have 50 dedicated processes to handle them. If you have a low access site/DAD, then these connections can be concentrated down to a smaller number of shared servers. In a small scale configuration the SSO server (Login Server) might benefit from this, for example, as it tends to run behind the same Web server that OracleAS Portal uses and may not run with a high DB connection pool efficiency. Note however that if you don't get the MTS configuration correct, then it may become a bottleneck for the execution of the portal's PL/SQL code.

- The database is configured to allow for enough database sessions from each middle tier and other clients 2*MaxClients*NumberOfMidTiers.

My Oracle File System (FS) Cache configurations

- mod_plsql FS cache is moved to a separate file system to avoid IO contention. NOTE : If this is changed, make sure that web.xml is also changed correspondingly. You can learn more about this process in the document Configuring Session Cache for Performance on PortalCenter.
    - PlsqlCacheDirectory /some/other/file/system
- Configure the session cache for mod_plsql to reside on a faster file system
    - My Oracle has changed the directory $OH/Apache/modplsql/cache/session to be a symbolic link pointing to a RAM disk
- Resized the middle ttier file system cache size (default is not enough for My Oracle)
    - PlsqlCacheTotalSize 8192000000 (8 Gigs)
- Configured mod_plsql to cleanup all cached content older than 15 days
    - PlsqlCacheMaxAge 15
- Configured cache cleanup to happen every day at 11:00 PM in the night
    - PlsqlCacheCleanupTime Everyday 23:00

Portal OC4J configuration

- Configured two OC4Js running per OC4J_Portal. This gives you a backup if/when the first falls over.
    - numProcs="2" in opmn.xml for the "OC4J_Portal" instance
- Removed the "-xincgc" line from the opmn.xml since it's a performance hit, and increase the mx and ms settings.
    - The -mx option is the maximum memory that is allowed for the JVM. Beyond this, the JVM will not allocate any more memory and report "Out of memory errors" in the logs.
    - The -ms option is the amount of memory to start with
    - -Xincgc : Enables the incremental garbage collector (IGC). The IGC, which is off by default, will eliminate occasional garbage-collection pauses during program execution.
- Configured "cacheDir" parameter in web.xml to match that of the mod_plsql's PlsqlCacheDirectory setting

Configure SQL*Net for better performance

- Allow for an SDU set "large enough". My Oracle sets this to 8761
- TCP.NODELAY set to "yes"

You can learn more about these TCP/IP and SQL*Net settings in the TechTip Tuning Oracle Net Services to optimize modPLSQL Database access times on PortalCenter

PERFORMANCE LOGGING ANALYSIS

What are the benefits of all this careful architecture selection and machine configuration?

Through using the performance logging features of mod_plsql and the PPE, it is possible to obtain detailed log entries that may be analyzed, providing an opportunity to gauge the performance of a portal.

Oracle9*i*AS Portal, Release 1 (3.0.x) provided portal administrators with a collection of reporting and charting portlets, available by default on the Monitor tab, for reporting object access information. The information in these reports and charts was retrieved from the Oracle9*i*AS Portal Activity Log tables, which were located in the Oracle9*i*AS Portal Repository. In addition, a set of public views were made available for anyone who wished to issue their own SQL queries directly against this data.

With the move to OracleAS Portal, and the introduction of OracleAS Web Cache into the portal architecture, the information in these tables became inaccurate. Therefore, all the pre-built reporting and charting portlets were removed from Portal. The Oracle9*i*AS Portal Activity Log tables and views still remain in the OracleAS Portal Repository for backward compatibility, however, these tables may be removed entirely in a future release.

This reporting functionality will be restored in a future release of the Oracle Application Server; however, in the interim, it is possible to use a number of reports that execute against the data collected by the performance logging service of mod_plsql. For a full description of how to implement this logging service, please see the technote 'Performance Monitoring with mod_plsql in OracleAS Portal ', which is available on Portal Center.

Examples of the types of reporting that can be obtained from the performance logging scripts follow.

UNIQUE LOGINS PER DAY

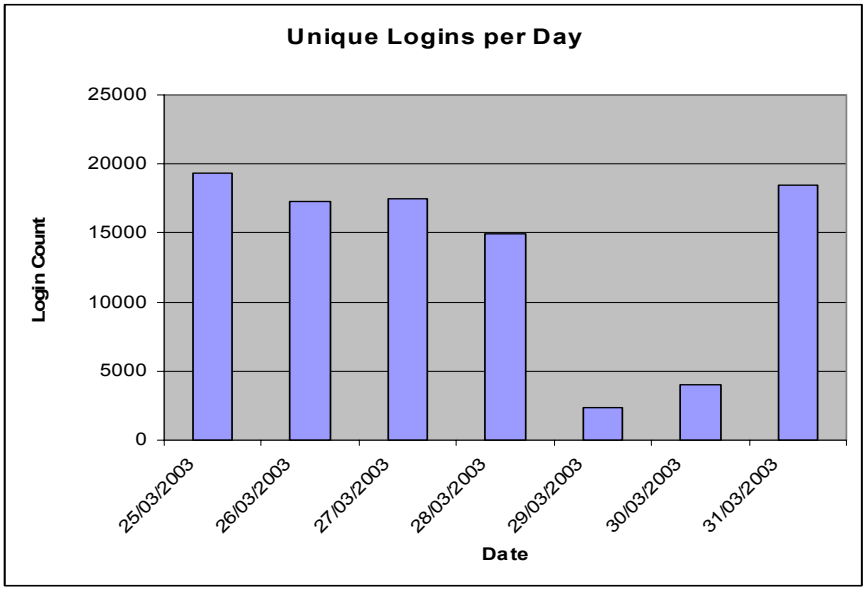

Figure 5 : Unique logins per day

The chart in Figure 5 indicates the quantity of unique successful logins over a 7-day period. A successful login is composed of one or more successful calls to wwptl_login.login_url and wwsso_app_admin.ls_login, terminated by a successful call to wwsec_app_priv.process_signon.

The chart shows that there is significantly less demand for login access during the weekend and that demand is higher at the beginning of the week than the end of the week.

UNIQUE LOGINS PER HOUR

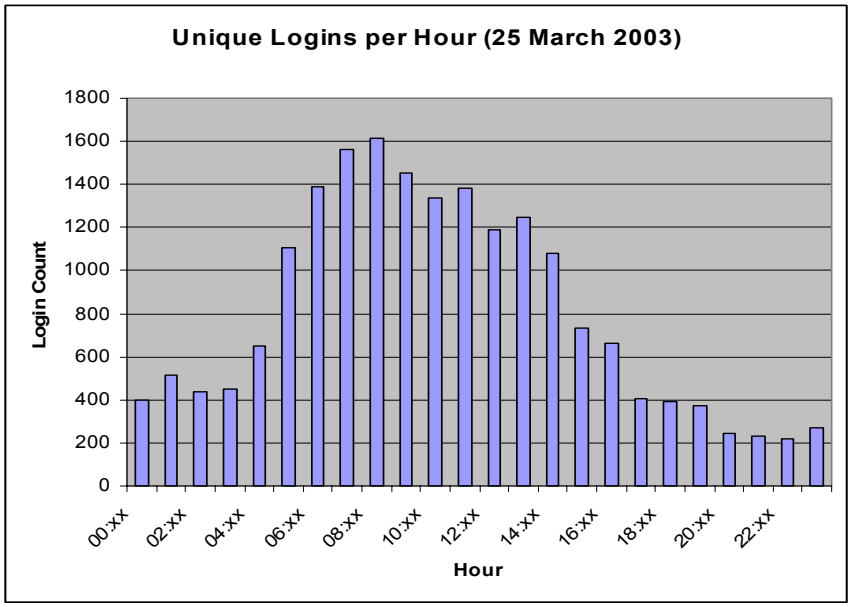

Figure 6 : Unique logins per hour

The chart in Figure 6 drills down further on the unique login data to show the unique successful logins for a given hour. The time measurement is based on the timestamp of the log for the SSO machines, these machines are based in the PST time zone (GMT-8), so it reasonable to conclude that the majority of the logins are from the US rather than other timezones around the world.

The peak login time is between 6 am and 10 am, with logins tailing off through the day.

PAGE VIEWS PER DAY

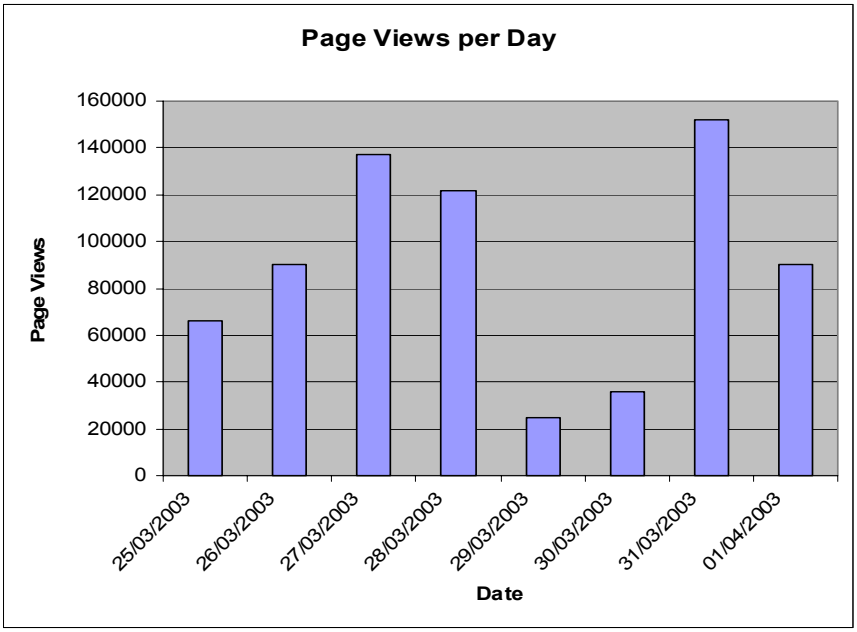

Figure 7 : Page views per day

As shown in the chart in Figure 7, total page views in a 7-day period can vary greatly, in this example almost twice the amount of pages were requested on Monday March 31st than were requested on Tuesday March 25th.

Similar to the daily login rate, this chart shows that demand for content is significantly reduced during the weekend, but here content demand has a peak on Monday and another peak on Friday.

PAGE VIEWS PER HOUR & CPU LOAD

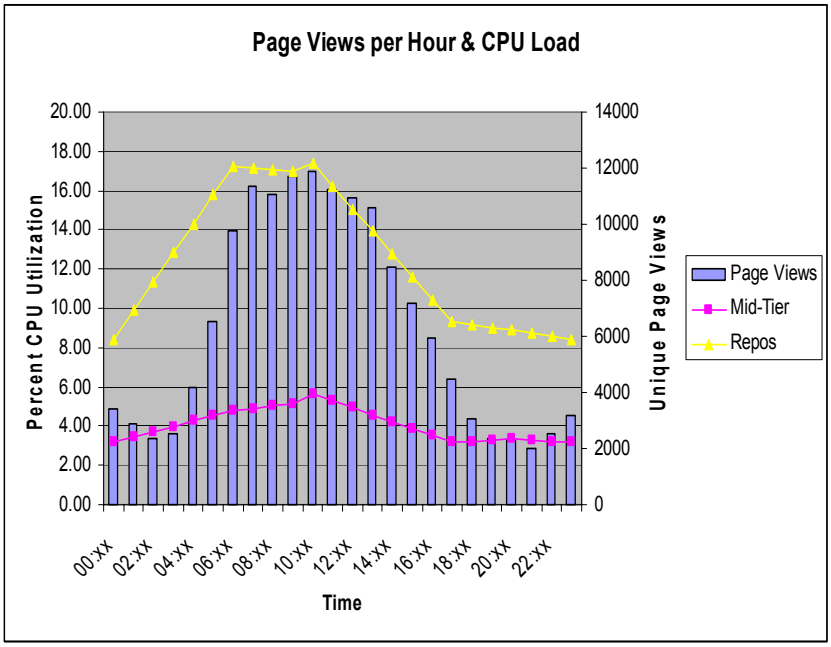

Figure 8 : Page views per hour & CPU load

The Page View chart in Figure 8 gives an insight to the loading that the My Oracle machines are under during a 24-hour period.

The chart shows three elements:

- Unique Page View count in each hour
- CPU utilization averaged for all My Oracle middle tiers
- CPU utilization for the My Oracle repository server

The graph shows that there is a distinct capacity for increased load on the My Oracle servers. Even at peak page load (~12,000 page views) around 10am, the middletier servers never exceed 6% utilization and the repository server never exceeds 17% utilization.

TOP TEN PAGES ON MY ORACLE

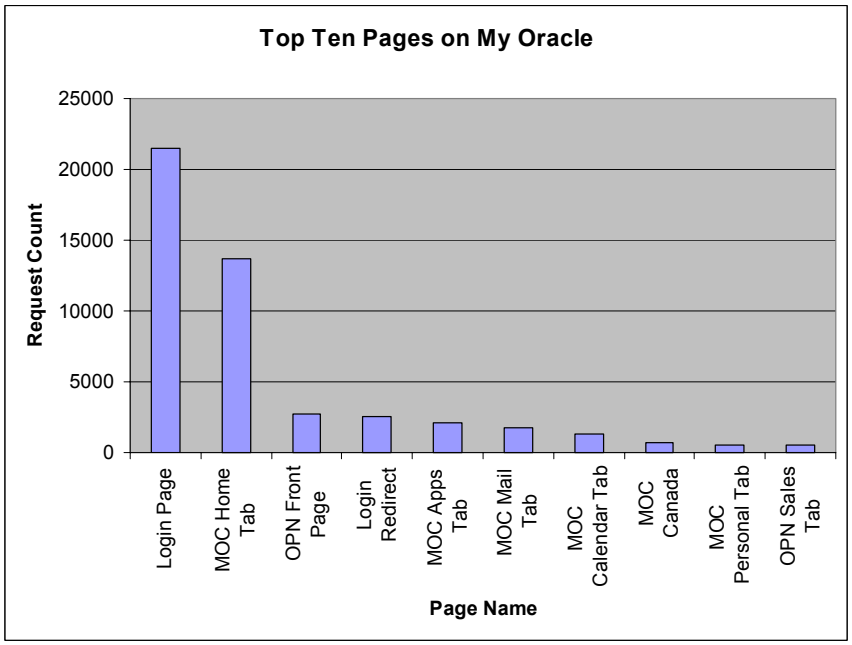

Figure 9 : Top ten pages on My Oracle

The chart in Figure 9 shows that there is a large number of My Oracle users logging in and viewing the home page. The very similar page view numbers for the next eight pages fit with the classic portal navigation model. This model follows a 'landing pad' approach where all users hit a single page and then navigate off to a page specific to them. Caching the home page in its entirety and reducing the customization options available on the home page will ensure that a cache hit is obtained whenever possible. Subsequent pages navigated to by the user may then be more customizable as it is likely to receive fewer requests than the main landing pad page.

TOP TEN PORTLETS & RESPONSE TIME

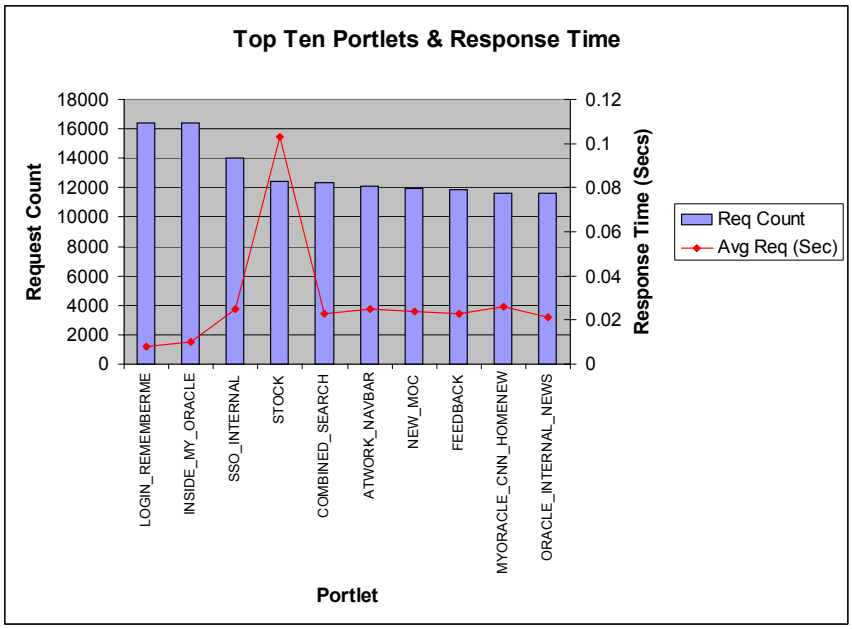

Figure 10 : Top ten portlets & response time

The chart in Figure 10 shows the ten most popular portlets in My Oracle. It is interesting to note that the ubiquitous Stock portlet remains as popular as ever, and that despite this portlet having no caching at all, it still manages to execute on average in 0.1 secs.

RESPONSE TIMES

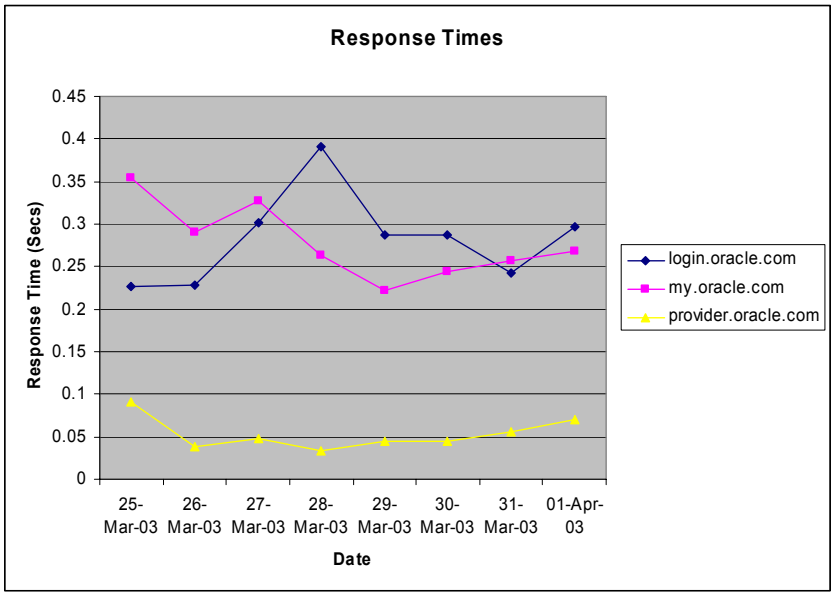

Figure 11 : Response times

The chart in Figure 11 shows the average response times for each of the logical divisions that make up My Oracle.

The average response time for My Oracle over the seven day period is between 0.2 and 0.35 seconds. The exceptional response times for the provider.oracle.com servers can be attributed to the high level of static content caching that is occurring in-memory of the OC4J engine.

SOFTWARE PERFORMANCE CONSIDERATIONS

So far we have examined the aspects of physical configuration that affect performance. During the development of any portal it is possible to design the content to be as performant as possible from the outset, through judicious use of caching and content design.

PAGES, PORTLETS AND PROVIDERS

Portlets are the fundamental building blocks of an OracleAS Portal page. Each portal page contains content presented through one or more portlets. At a technical level, a portlet is a standardized and secure object that produces a live area of HTML.

Providers act as containers for portlets; each portlet communicates with OracleAS Portal through its provider. Each provider may contain one or more portlets, with each portlet exposing an underlying application or information source. Each provider is defined, registered and managed within a single OracleAS Portal installation. Once a provider is registered with an OracleAS Portal installation, its portlets are available to be placed on portal pages. As explained in the following sections, providers can be distributed across multiple physical nodes, and a single page can contain portlets from multiple provider locations.

OracleAS Portal allows many types of providers and portlets to be created declaratively. Using the content publishing and application development tools built into OracleAS Portal, users can generate portlets that allow portal content elements (pages, navigation bars, banners, etc.) and application components (forms, reports, charts, dynamic pages, etc.) to be added to pages. The content areas and applications that contain these objects are automatically exposed as providers. Generating a built-in portlet or provider simply requires selecting the "Publish to Portal" check box in a Portal wizard.

Custom providers and portlets can also be built programmatically with the public API contained in the Portal Development Kit (PDK). Two options are available for development of custom portlets and providers: Database and Web:

- ***Database Providers*** — Portlets and providers are implemented in Java or PL/SQL and executed as stored procedures within the Oracle database. The OracleAS Portal middle tier communicates with these providers in two ways: through mod_plsql, if the provider resides in the local Portal database; or by SOAP over HTTP, if the provider resides in a remote database (in Oracle9*i*AS Portal Release 1 (3.0.x), Net8 connections were used in both cases - SOAP over HTTP makes it easier to communicate with remote database providers through firewalls). Note that execution in the database does not place any restrictions on the functionality of a portlet; database facilities allow for external communication in many ways, including HTTP connections to external content. Database providers are particularly appropriate for portlets that require significant interaction with the database and in situations where the development team has extensive Oracle PL/SQL development experience.
- ***Web Providers*** — Portlets and providers are implemented in any Web deployment environment (e.g., Java, ASP, Perl, etc.) and executed as an application external to OracleAS Portal, which also communicates with these providers using SOAP over HTTP. Web providers are most appropriate for external information sources (e.g., Internet news/business information) and in environments with experience using Java and other Web development languages.

PAGE AND PORTLET CACHING IN ORACLEAS PORTAL

As described in the previous sections, portal pages are not defined as static HTML files - they are generated from content that is either stored in a database or retrieved from a Web provider. Dynamically assembling a page from multiple sources can be a very expensive operation, in terms of both CPU consumption and network latency. Therefore, OracleAS Portal uses sophisticated caching mechanisms to minimize database and provider calls. Page metadata, portlet content, and rendered pages are all cached to achieve a high degree of scalability and performance while still enabling dynamic personalization and protecting the security of portal content.

CACHING TYPES

Three types of caching are used for OracleAS Portal content:

- *Invalidation-based caching* uses OracleAS Web Cache to cache content. When various events occur that require that content in the cache be refreshed, the portal repository or providers send invalidation messages to Web Cache. When an entry in Web Cache is invalidated, it will be refreshed the next time it is requested. Web Cache entries are also invalidated when their time in the cache exceeds the maximum expiration time set in Global Settings. *Invalidation with expiry-based caching* is an extension to the invalidation-based caching method, offering portlet developers finer-grained control over when and how their portlets expire. The portlet developer may indicate that a portlet will use invalidation-based caching and can also provide a time after which the content should expire if no invalidation message has been received by the time that the expiration definition has elapsed.
- *Validation-based caching* uses the portal cache to cache content. Before an entry in the portal cache is used, the Parallel Page Engine or mod_plsql contacts the portal repository or a provider to validate that the cache entry is still valid.
- *Expiry-based caching* also uses the portal cache. Expiry-based cache entries use an expiration period to specify the length of time they are valid, known as the *cache retention period*. Note that pages that use expiry-based caching may also be cached in users' browsers.

Validation-based and *invalidation-based caching* both ensure that users are seeing the most up-to-date information. These types of caching are most appropriate for content that can change frequently or on a random basis and for which the cost of the validity check or sending the invalidation message is low. For example, a document or folder can be quickly checked for changes by comparing timestamps between the cache and the repository.

Expiry-based caching, on the other hand, pins a cache entry to the cache for a specified period of time. Expiry-based caching can result in better performance, since certain checks and refresh operations are eliminated; however, users may be seeing stale content if the underlying data changes before the expiry-based cache entry expires. Expiry-based caching is most appropriate when content is not time-sensitive and the cost of validating or invalidating the content is high. For example, use expiry-based caching where an expensive query is required to see if a new product has been added to a catalog, yet the catalog is updated infrequently and users don't need to see the updates immediately.

USER AND SYSTEM LEVEL CACHING

OracleAS Portal supports user customizations of pages and portlets. It also filters out portlets and content items that a user is not authorized to view. Because of these customization and security features, the view of a page can vary from one user to the next. OracleAS Portal's caching is designed to allow content to vary on a per-user basis, even though the physical URL being cached might be the same across users.

A portlet developer can specify either of two levels of caching for a portlet: *user level* and *system level*. User level caching is for a specific user—the cache entries stored are unique for that user only. System level caching allows users to share a single cache entry.

System level caching requires that an object be accessible to all users (i.e., the object does not enforce access privileges) and is not customizable. A big advantage to specifying system-level caching for an object is that a single copy of the object can be cached and shared by all users. Examples of content that are suitable for system-level caching include page banners and news portlets.

User level caching must be used for objects that can be customized or that enforce their own access privileges. Portal cache and Web Cache cache user-level entries on a per-user basis; these cache entries, are protected from unauthorized access and can therefore be shared across multiple sessions for the same user. For example, if a user logs out of OracleAS Portal and then returns the next day, that user will use valid cache entries from the previous day's session that are still stored in the cache.

BROWSER SETTINGS

To maximize the effectiveness of OracleAS Portal caching, and to ensure that users always see the most recent content, it is recommended that browsers be set to use the following options:

- In Internet Explorer, under Tools > Internet Options > General Tab > Temporary Internet files (Settings):
  - [ ] Check for newer versions of stored pages:
  - [x] Every visit to the page
  - [ ] Every time you start Internet Explorer
  - [ ] Automatically
  - [ ] Never
- In Netscape, under Edit > Preferences > Advanced > Cache:

Document is compared to the document on the network:

- [ ] Once per session
- [x] Every Time
- [ ] Never

CACHING OPTIONS FOR PORTAL CONTENT

The caching options for the different types of portal content, including pages, portlets, documents, and events are described in detail in the following sections.

PAGE CACHING OPTIONS

From release 9.0.4 onwards, pages can now be cached at the system level, which places a single copy of an object in the system cache (on the middle tier) for all users. Cache a page's *definition* when the page contains highly dynamic content; cache a page's *definition and content* when the page contains more static content that is unlikely to change within the amount of time you specify. Caching at the system level is recommended only when page customization is not important, as the same page is used for all users.

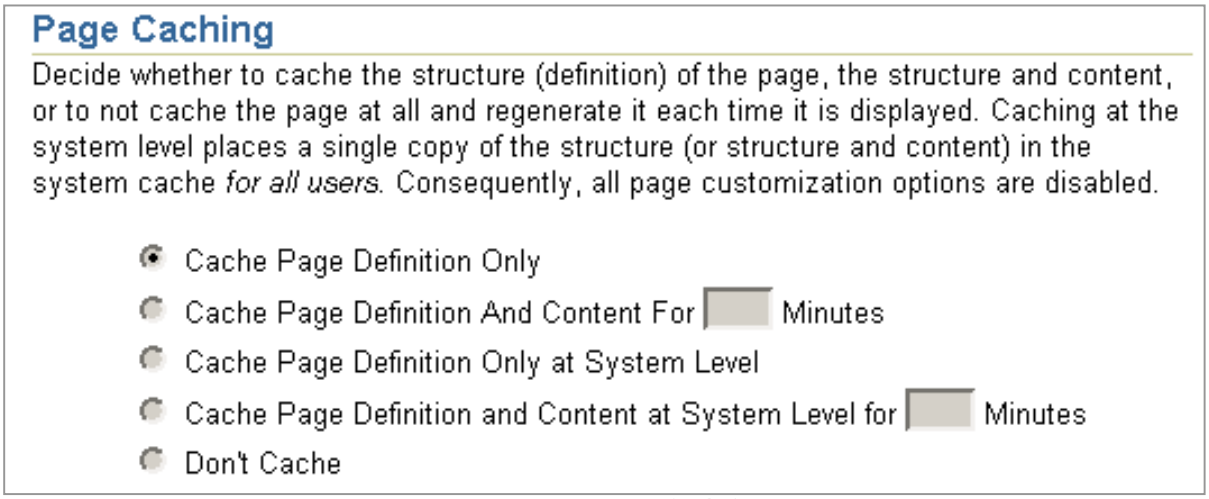

Figure 7: Page Properties for Caching

Cache page definition only

This option enables caching of the page definition only. The page definition includes the metadata that describes the page structure and identifies the portlets the page contains, plus any items that are managed on the page. When this caching option is selected, the page definition will be cached in both Web Cache and the portal cache for each user. The page definition includes the following information:

- Metadata describing the page structure
- Identification of the portlets that the page contains
- Items that exist on the page excluding the page content
- Style information for the page

You can choose this option for the following types of pages:

- Pages with highly dynamic content
- Pages that contain portlets having short expiration periods
- Pages that contain portlets using validation-based caching or invalidation-based caching

If the Web Cache entry is invalidated, the Parallel Page Engine will contact the page repository to see if the portal cache contains the newest version of the page definition. If the cached definition is valid, the Parallel Page Engine will use the portal cache entry to assemble the page. If the portal cache entry is not valid, the page definition will be regenerated and both cache entries will be refreshed. Generating the page definition is a CPU-intensive operation that occurs in the Portal database, so either type of caching will help to reduce the load on the database In previous versions, the style of the page (exposed as a cascading style sheet or CSS) was cached with the page definition. This is not the case in OracleAS Portal 10*g* (9.0.4). The CSS is cached in Web Cache as a unique object and is referenced in the page definition, thus when the page is rendered by the browser, a request to the Web Cache will made for the cached CSS file.

Cache page definition only ensures that users will always see the most up-to-date definition of a page. This option eliminates the expensive step of regenerating the page definition, but does not eliminate portlet validity checks (for validation-based portlets), portlet content retrieval (for expired or invalid portlets), or page assembly.

Note that portlet validity (for validation-based portlets) is checked on each request, unlike expiry-based caching, which only performs these checks when the page has expired. Therefore, *cache page definition only* should be used when the page contains portlets that must use validation- or invalidation-based caching, or that have short expiry periods.

Cache page definition and content for N minutes

This option enables expiry-based caching for a page: the page definition will be cached as for the previous option, but the assembled page containing the rendered content of all of its portlets will be cached for the specified period of time (minutes). With expiry-based caching, the fully assembled page is cached in the portal cache and the user's browser. OracleAS Portal also instructs the browser to cache the page. Web Cache is *not* used to cache the fully assembled page, since it would be redundant to cache a page there in addition to caching it in the browser.

Expiry-based caching ignores the caching options for the individual portlets on the page, so it is possible that some content will be stale when the page is viewed from the cache. However, *cache page definition and content for N minutes* eliminates the need for the page to be dynamically assembled with every request, so it is the best choice for minimizing the load on the Parallel Page Engine and the middle tier servers.

Note that the Parallel Page Engine always checks the validity of a page definition before serving a page. Therefore, if the page definition changes, OracleAS Portal ensures that the page will be regenerated even if it has not expired (see Caching Scenarios, below).

Cache Page Definition Only at System Level

By using this option, you can create a single cached copy of the page definition in the system cache for all users. Because the page definition is the same for all users, page customization options are disabled. This caching option greatly reduces storage requirements and improves performance. You can select this option for pages that have highly dynamic content, but do not require customization.

Cache Page Definition And Content at System Level for [ ] Minutes

By using this option, you can create a single cached copy of the page definition and page content, including the rendered content of all portlets. The cached information remains the same for all users in the system cache for a specified period. Page customizations are not possible because the page definition and content is the same for all users. You can select this option for pages that are more static and are unlikely to change within the specified period. When you select this option, portlets display the public content only. If you select this option, you can include a Refresh link on the page to regenerate the page content from the database.

Don't cache

This option disables page caching. This option should be used sparingly, as dynamic page generation can affect the performance of your portal by placing a heavy load on both the database and the middle tier. However, you may want to disable page caching in the following circumstances:

- Your page contains PL/SQL items or URL items that are rendered in place (i.e., displayed directly on the page) and that produce dynamic content that must always be up-to-date (a portlet that is not cached or that uses a short expiration would be a better choice for this kind of content).
- You are concerned that a soft-invalidation event, such as a change to user group membership (see Invalidation Events, below), may result in a cached version of a page being viewed by an active user who has recently lost privileges on the page.

Page caching works identically for normal pages and for JavaServer Pages (JSPs) that have been loaded into the repository (internal JSPs). The OracleAS Portal caching mechanisms do not cache external JSPs—JSPs that contain portlets but that run externally to OracleAS Portal; however, caching is managed by OracleAS Portal for the portlets contained on external JSPs. (Internal and external JSP pages are a new feature in OracleAS Portal 10*g* (9.0.4).)

The following table summarizes the recommended page caching options for different types of content:

Content Contained on the Page	Recommended Page Caching Option
Static (e.g., items that are manually updated).	Cache page definition and content with a long retention period. Changes to the page definition (including modifications to items) will force a refresh, even is the page has not expired.
Portlets with expiry-based caching.	Cache page definition and content. Retention period should be less than or equal to the shortest portlet retention period.

Portlets with invalidation or validation-based caching where some stale content is acceptable.	Cache page definition and content with a short retention period.
Portlets with invalidation or validation-based caching where content must always be up-to-date.	Cache page definition only to ensure that portlet validity is checked when the page is assembled.
Dynamic items (PL/SQL, URL) where some stale content is acceptable.	Cache page definition and content with a short retention period.
Dynamic items where the content must always be up-to-date.	Don't cache. It's a good idea to put this content on pages that are not frequently accessed, or use portlets instead of items and choose Cache page definition only.

PORTLET CACHING OPTIONS

Like pages, portlets can use invalidation-based, validation-based, or expiry-based caching. Database portlets can also use a combination of invalidation and validation. The significant enhancement in OracleAS Portal 10*g* (9.0.4) that improves portlet performance is the ability to cache portlets at the system level. Caching a portlet at the system level places a single copy of the portlet in the system cache for all users.

To improve page performance, perform the following steps:

Figure 7: Portlet Properties for Caching

Click the Actions icon that is located above the portlet in the Graphical editing view.

Select the Edit Portlet Instance link.

Select the "Cache Portlet at the System level" check box.

The following points must be considered before caching portlets at the system level:

Caching a portlet at the system level disables all customization options for the portlet.

Caching a portlet at the system level does not enforce access privileges for the portlet.

Caching a portlet at the system level displays only public data. Therefore, portlets, such as the Recent Objects portlet or the External Applications portlet that do not contain public data are not displayed.

If the page that includes a portlet is cached at system level, then the portlet is also cached at the system level.

If the Web Provider specifies system-level caching for a portlet, then the portlet is cached at the system level. In this case, you can change the cache setting for the portlet manually because it is already set.

If a portlet is not cached at the system level and the page that includes this portlet caches both the page definition and the content, then this portlet will be cached at the user level. With user-level caching, this portlet can be customized and access privileges can be enforced.

Therefore, you must not cache a portlet at the system level if that portlet includes sensitive data or other information that you do not want to display to all users. Examples of content that may be suitable for system-level caching include page banners and news portlets.

All types of caching function the same as for pages, with the following differences:

- With all types of caching, the portlet *content* is cached, not the portlet metadata. This means that only the rendered view of the portlet is cached, but not any metadata that is used to generate or personalize the portlet.
- When expiry-based caching is used for a portlet, the provider is not contacted until the cache entry expires (pages are always validated, even if the page cache entry has not expired).

Each type of portlet has different ways of defining and implementing caching, as described in the following sections.

PAGE PORTLETS

Page portlets follow the same caching behaviors as described above for pages.

With Oracle9*i*AS Portal Release 2 (9.0.2), page portlets are now used to implement banners and navigation bars.

PORTAL DATABASE PROVIDERS

The following Portal database provider portlets use a combination of validation and invalidation -based caching with a user-specified expiry period:

- Calendars
- Dynamic Pages
- Hierarchies
- Menus
- Reports
- Charts
- XML Components
- URL Components
- Data Components

To enable caching on any of these components, enter a value greater than 0 in the "Expire after (minutes)" field of the Component Properties, found under the Display Options tab. Although an expiration period is specified, these portlets do not use expiry-based caching. The expiration period is used to invalidate the Web Cache entry and to validate the portal cache entry. If the content has expired, Web Cache marks the content as invalid and contacts mod_plsql. Mod_plsql refreshes the portal cache and returns the content to Web Cache.

These portlets are also cached at the user level. All cache entries for these portlets are stored in Web Cache on a session-basis. User-level cache entries are stored in the portal cache. Web Cache entries can be created or refreshed from the portal cache if the portal cache entry is still valid.

Cache entries for these portlets will be refreshed if any of the following events have occurred since the last refresh:

- The portlet definition has changed
- The portlet has been customized by the requesting user

- The cache entry is older than the expiration period specified for the portlet, or the maximum retention period specified for Web Cache
- The portlet is explicitly invalidated by its owner
- The portlet's access control list changes

Changes to underlying data will not be displayed until the cache entry is refreshed.

Caching is not enabled for Links, List of Values, Forms, and Frame Drivers.

Note: OracleAS Portal database providers were known as applications in previous releases. .

CUSTOM PORTLETS

Custom portlets generated from database and Web providers support all types of caching. The portlet developer specifies the type of caching that is used through the APIs of the Portal Development Kit (PDK).

As with all other types of portlets, content from custom portlets is cached in either Web Cache or the portal cache and refreshed as necessary by the provider.

The following table summarizes the recommended portlet caching options for different types of content:

Content	Recommended Portlet Caching Options
Static, infrequently updated, shared by all users (e.g., page banner, or company tombstone data like mission statements and office locations).	Expiry-based with long retention period. System level.
Static, periodically updated, shared by all users (e.g., company news bulletins).	Expiry-based with short retention period (depending on update frequency). System level.
Static, periodically updated, personalized (e.g., list of favorite Web sites, customizable by user).	Invalidation-based, or combined invalidation and validation-based. User level.
Dynamic (generated from database), does not require real-time updates, shared by all users (e.g., product catalog).	Expiry-based. System level. Choose retention period to meet business requirements for timeliness.
Dynamic, personalized, does not require real-time updates (e.g., delayed stock quotes).	Expiry-based with short retention period. User level.
Dynamic, personalized, requires real-time updates (e.g., active trouble tickets in a helpdesk portlet, real-time stock quotes).	Don't cache or use combined validation/invalidation-based if the application can trigger an invalidation message. User level.

Use invalidation or invalidation-with-expiry based caching instead of expiry-based caching if your application can be modified to send invalidation messages to Web Cache when content needs to be refreshed, or if the user can manually refresh the content (e.g., by clicking a Refresh link in the portlet).

If appropriate, try to combine invalidation-based with validation-based caching (the combination is currently available for database portlets only). The only drawback to combining the two types of caching is that the portlet developer must implement the PDK interfaces for both.

Seeded Portlets

OracleAS Portal ships with a large number of built-in portlets, known as *seeded portlets*. Seeded portlets implement different types (Invalidation, Validation, Expiry) and levels (System, User) of caching, as appropriate for the content that they contain. The caching for these portlets is predefined and cannot be modified by OracleAS Portal users or administrators. However, it is helpful to keep the following points in mind when using these portlets:

- Many of the seeded portlets are provided for administration of OracleAS Portal users and objects. Most of these use validation-based caching (or a combination of invalidation/validation) since they are not customizable and seldom change.
- Certain seeded portlets are useful productivity tools for OracleAS Portal users and can be customized. These include Favorites, Recent Objects, the HTML portlet, and Set Language. These portlets all use combined invalidation/validation, except for Set Language, which changes infrequently and uses validation-based caching only. The content for these portlets will be refreshed when a user customizes them.
- Portlets that implement search results (including Category, Perspective, Saved Search, and Search Results) are not cached, as their content is highly dynamic. Placing these portlets on a page that does not use expiry-based caching could affect response times, as the search will be executed for each page request.
- The Notification portlet, which tracks notifications for content approvals, uses invalidation-based caching. It will be refreshed for a user when the status changes for an item that needs their attention (e.g. an item is edited and the user is in the list of approvers).

Documents and Images

Documents (including images) that are stored in the portal pepository are cached in the portal cache using validation-based caching.

By default, Web Cache also caches all images (*.gif, *.jpeg, and *.jpg) until they are invalidated or Web Cache is cleared (the default rule actually specifies that images be cached "forever"). However, because portal images may be subject to access control and not accessible to all users, OracleAS Portal defines a Web Cache rule that disables caching for images from the portal repository (everything under the path "/docs"), except for the images that are associated with Categories, Perspectives, and Styles. (Note that this does not mean the images that are classified by a category/perspective, but rather the images that belong to the category/perspective definition and are used as a visual identifier of the category/perspective.) OracleAS Portal does not place access controls on categories, perspectives, or styles, nor on their associated images.

OracleAS Portal also makes extensive use of images in the "/images" directory on the middle tier file system. When OracleAS Portal is installed, a Web Cache directive is created that sets the cache lifetime of these images to 2,592,000 seconds (30 days). This directive overrides Web Cache's default "cache forever" rule for images.

Events

Events are a new feature in OracleAS Portal 10*g* (9.0.4). Portlets can define events, and those events (such as a button being pushed or a data value selected) can be mapped to page parameters to enable inter-portlet and inter-page communication.

Event metadata (the data that describes the behavior of an event) is cached in both Web Cache and the portal cache. The caching behavior for events is the same as for page definitions - a combination of invalidation-based and validation-based caching. Event definition cache entries will be invalidated when they are modified.

Login Metadata

Providers can be passed login metadata that describes the current session so that they can track state for an OracleAS Portal user. This session data is cached in Web Cache and passed to providers along with requests for portlet content.

INVALIDATION EVENTS

When an object is cached, its cache entry will be used until the cache entry expires, the cache is cleared, a message is sent from OracleAS Portal or a provider to invalidate the entry (Web Cache), or the cache entry is determined to be invalid (portal cache).

OracleAS Portal traps two types of invalidation events:

- *"Hard" invalidations* are events that require an immediate invalidation of a cache entry.
- *"Soft" invalidations* are events that will generate an invalidation message some time after the event has occurred.

HARD INVALIDATIONS

The following events will cause a hard invalidation. In addition, some of these events will also force a refresh of the portal cache on the next validation check of an affected object.

Event	Initiator	Web Cache Entries Invalidated	Portal Cache Entries Refreshed
Clear the Entire Cache	Portal Administrator (in Global Settings)	Entire Cache	All pages for all users
Clear the Cache for User	Portal Administrator (in Global Settings)	All for the specified user (including Login Metadata)	All pages for the specified user
Invalidate My Cache	Any User (by using the supplied Clear Cache Portlet)	All for the requesting user (including Login Metadata)	All pages for the requesting user
Clear Cache on Object	User with Manage Privilege on Object or Provider (through PDK call). Portal provides this option on the Access tab for the following objects: • Page Template • DB Provider • DB Provider Portlet • Page Group • Page • Any Portlet (in Portlet Repository)	All for the specified object	All pages that contain the specified object, where the object is of the type listed under Initiator Objects invalidated by a PDK call will not cause a page refresh
Click "Refresh" link on Page	Any User	Page definition for the current user	Page for the current user
Modify Page Properties	Page Owner/Manager	All page definitions for modified page	Affected pages for all users
Add, Edit, Delete Item on Page	Page Owner/Manager	All page definitions for modified page	Affected pages for all users
Modify Default Page Layout (including editing of region properties and adding/removing portlets)	Page Owner/Manager	All page definitions for modified page	Affected pages for all users
Modify Page Template	Template Owner/Manager	All page definitions that use the modified template	Affected pages for all users

Event	Initiator	Web Cache Entries Invalidated	Portal Cache Entries Refreshed
Modify Style	Style Owner/Manager	All page definitions that use the modified style	Affected pages for all users
Modify Provider Properties	Portal Administrator	All page definitions that contain portlets from the modified providers All portlet content managed by the modified provider Any Login Metadata that pertains to the provider	Affected pages for all users Affected portlets for all users
Page expires	Automatic (part of page definition)	N/A (page definition does not expire)	Assembled page for all users (page must be reassembled for each user)
Customize page	Any User	Page definition for the current user	Affected page for the current user
Edit Defaults (portlet)	Page Owner/Manager	Non-customized instances of the affected portlet for all sessions (if invalidation-based)	Non-customized instances of the affected portlet for all users (if validation or expiry-based)
Customize portlet	Any User	Affected portlet for the current user (if invalidation-based)	Affected portlet for the current user (if validation or expiry-based)

SOFT INVALIDATIONS

Soft invalidations are required when a change to security can potentially change the access rights to a large number of objects. Rather than invalidating the affected objects immediately, which could cause an undesirable spike in the load on Web Cache or OracleAS Portal, the invalidation messages are deferred. A background OracleAS Portal process manages the queue of deferred soft invalidation events.

Web Cache's surge protection feature can also be used to minimize the impact of bulk invalidation. Refer to the Web Cache documentation for more information on this feature.

The following table describes the events that will trigger soft-invalidation of cache entries:

Event	Cache Entries Invalidated
Delete user	All for the deleted user
Add user to group	All for the added user
Remove user from group	All for the removed user
Add group to group	All for every user in the added group
Remove group from group	All for every user in the removed group
Delete group	All for every user in the deleted group
Change privilege on object in portal repository	All for the affected object (see Clear Cache for Object, above)

If you are concerned that soft invalidation may allow a user to view content for which they are no longer authorized (because of a change to privileges or group membership), you have two options:

- Initiate a hard invalidation message for specific objects or specific users, as described under Hard Invalidations, above; or
- Do not use caching for objects that need to be protected from a change in privileges.

If a custom provider manages its own privileges, it is responsible for sending invalidation messages if those privileges are altered.

PORTAL PERFORMANCE FACTORS

The previous sections discussed the importance of caching for achieving good performance in OracleAS Portal. However, a number of other factors can influence the performance of your portal. The most important performance factors are described in this section.

PAGE CACHE HIT RATE

The Page Cache Hit Rate is the number of page definitions or page assemblies you can retrieve from the cache compared to the number of pages that must be regenerated. The Page Cache Hit Rate is affected by the following factors:

- How often pages are modified by their owner
- How often pages are customized by end users
- Whether you are using invalidation-based, validation-based, or expiry-based caching
- The number of times pages are revisited by the same user

Pages are cached at the user level, which means that a distinct cache entry is created for each page for every user that has visited the page. The hit rate will be higher if users tend to revisit the same pages several times, rather than visiting each page only once.

Page caching minimizes the number of times the following operations are performed:

- *Page Definition Generation:* Generating the page definition is done with PL/SQL stored procedures and is CPU-intensive on the Portal database. Fetching of cached page definitions from either Web Cache or the portal cache is a relatively inexpensive operation, and 'pinging' the portal repository to verify that the page remains valid is a much less expensive operation than a full generation of the page. The Hit Rate on page definitions will usually be high (e.g. 95% or greater) since page definitions change infrequently.
- *Page Assembly:* Page assembly, performed by the Parallel Page Engine, is CPU-intensive on the middle tier. Page assembly is minimized when you use expiry-based caching.

Performance will be best if you can use expiry-based caching, especially for frequently visited pages. When expiry-based caching is used, the page can be pulled directly from the portal cache or the browser cache without an expensive assembly step. The tradeoff with expiry-based caching is that content may be stale.

PORTLET CACHE HIT RATE

The Portlet Cache Hit Rate is the number of portlet requests that can be satisfied from the cache compared to the number of portlet requests that must be handled by a provider. The Hit Rate is affected by the following factors:

- How often portlet content changes
- How often portlets are customized by end users
- Whether you are using invalidation-based, validation-based or expiry-based caching
- Whether you are using system-level or user-level caching

In general, your best performance will result from expiry-based caching. The portlet content is cached by the portal cache and does not have to be validated or regenerated until it expires. System-level caching allows for a single copy of the portlet content to be cached (in either Web Cache or the portal cache) and shared by all users.

The placement of portlets will also have an impact on system performance. If you place a portlet that does not use caching (or that expires or is invalidated frequently) on a frequently visited page, your overall hit rate will suffer as the portlet content must be frequently refreshed. Highly dynamic portlets should therefore be placed on less visited pages, if you are concerned about system throughput.

HTTP SERVER

The HTTP Server is not usually a source of bottlenecks, but there are a number of things that can be done to tune its performance. For more information, refer to the *Oracle Application Server Oracle HTTP Server Performance Guide* for your platform.

LOGIN RATE

The Login Rate is the rate at which users log in to the portal, thereby placing a load on the OracleAS Single Sign-On (SSO) Server and Oracle Internet Directory (or other LDAP server, if used). Explicit logouts will also place a load on the SSO Server and directory. Implicit logouts (closing a browser) do not impact the SSO Server.

PORTLET EXECUTION SPEED

The portlet execution speed is the average time required to execute all (uncached) portlets on a page. Since portlets execute in parallel, this measure will be equal to the execution speed of the slowest portlet, plus page assembly overhead. The portlet execution speed will differ from site to site, since each site will have a different mix of content and applications in their portal. Estimating this number will require running your own benchmark tests (if you already have applications that you plan to expose through portlets, you will already have a sense of the execution times). In general, the speed of page assembly will be limited by the execution speed of the slowest portlet on the page (when the portlet output is not cached).

PAGE COMPLEXITY

Page security and the number of items, tabs, and portlets on a page will affect the time it takes to generate page metadata. The number of portlets on a page will affect page assembly times in the middle tier, especially if each portlet must be contacted for a validity check or content refresh.

Page complexity, a function of page security and the number of tabs, items, and portlets on a page, affects throughput by increasing the amount of metadata that needs to be generated as well as the number of security and validity checks. Page complexity does not affect page assembly time in the middle- ier, but may affect the time it takes to validate and refresh portlet content.

NETWORK BANDWIDTH

The availability of network bandwidth will affect response times (as experienced by the browser), but will not affect server throughput.

LOAD DISTRIBUTION

The distribution of system load across servers will affect overall system performance. Will all components (middle tier, database, providers, etc.) reside on the same server, competing for system resources, or can the components be distributed across multiple servers? If load is distributed, will it be evenly balanced, or will one server do more work than another?

OTHER PORTAL ACTIVITY

The impact of other portal activity, such as content management, application development, and logging must be

considered. If the Portal database is heavily used to maintain content or build database portlets, that activity will affect the throughput of the page generation process. Reducing the amount of logging activity may also help to improve database throughput.

HARDWARE RESOURCES

Available memory could limit the performance of a server. When memory is constrained, disk input/output could also become a factor affecting performance.

CPU PERFORMANCE

Since page generation performance is CPU-bound, the performance of the CPUs on server machines will affect page throughput.

TYPE OF CONTENT

The amount and type of content that is served could affect system throughput. Multimedia content (large image files, video, Flash, etc.) could place an additional load on your HTTP server, network bandwidth, Web server file system, or database (if stored there).

Images that appear frequently on a site, such as company logos and banner icons, should be stored on the middle tier file system, rather than in the database. Images that are managed in the portal repository will not be cached in Web Cache, for security reasons (with some exceptions, as noted earlier in this paper).

SSL

SSL will place an additional load on the HTTP Server.

CONCLUSION

Oracle9*i*AS Portal pioneered the use of intelligent caching combined with a scalable, cross-platform architecture to enable deployment of secure, personalized content to a large audience. OracleAS Portal has further extended its scalability with the following features:

- Memory-based caching in OracleAS Web Cache to augment the file-based portal cache.
- Invalidation-based caching, which eliminates the need to contact the database or providers until content actually needs to be refreshed.
- Deployment on OC4J, the OracleAS lightweight container for J2EE that has set performance and scalability benchmark records and that features support for the latest J2EE specifications.
- Integration with the native load-balancing and failover capabilities of OracleAS Web Cache.
- Support for SOAP connections to remote database providers, allowing for more flexibility in distributing portal databases.

Without the unique combination of caching and scalability features offered by OracleAS Portal, it would be impossible to scale a portal site in a cost-effective manner. OracleAS Portal's architecture truly minimizes the hardware resources required to deploy a high-traffic portal.

The configuration and performance data from My Oracle demonstrates the use of OracleAS Portal technology to enhance an organization's online presence. As a portal that delivers compelling, customizable business content and applications to a targeted group of users, My Oracle helps Oracle to attract and retain customers with an increased level of online services while lowering the cost of marketing programs and service delivery. Flexible security enables the same portal infrastructure to also serve employees, and a robust architecture allows the portal to scale to large numbers of users. The most important aspect is that everything used to produce My Oracle is the exact same software customers use and buy in Oracle Application Server.

By understanding the OracleAS Portal architecture, and the process used to generate, assemble, secure, and cache personalized portal pages, you can make the right decisions when designing and building your enterprise portal.