Caching In on the Enterprise Grid

Turbo-Charge Your Applications with OracleAS Web Cache

An Oracle Technical White Paper
September 2005

INTRODUCTION TO GRID COMPUTING

Information Technology Challenges Organizations Face Today

The primary challenge facing organizations today is the high cost of their information technology infrastructure. This high cost arises from three related causes:

1. *Excess Computing Capacity* that is poorly utilized due to the need to build capacity for peaks, and the inability to use the spare capacity efficiently
2. *Expensive Capacity Growth* due to the inability to add capacity quickly, when needed, and in low-cost, modular units to avoid further compounding the problem of excess capacity
3. *High Cost of Management* due to the complexity of systems; the specialized tools, procedures, and skills required; and the large amounts of human intervention needed to manage systems.

Grid Computing and Oracle's Grid Computing Offering

Grid Computing is a new software architecture designed to effectively pool together large amounts of low-cost modular storage and servers to create a virtual computing resource across which work can be transparently distributed to use capacity efficiently, at low cost, and with high availability. The resources in a Grid can include storage, servers, databases, application servers, and applications. By pooling resources together, Grid Computing can offer dependable, consistent, pervasive, and inexpensive access to these resources regardless of their location and when needed, thereby fulfilling the need for computing capacity on-demand.

While Grid Computing has primarily been used by the scientific community to solve very specialized problems, the rapid evolution of cost-effective networked storage; high speed, high density blade servers; high speed network interconnects; and low cost operating systems; together with the evolving capabilities of systems software (databases and application servers) to exploit these advances have now made it possible for enterprises to leverage the benefits of Grid Computing.

Oracle offers a comprehensive solution to manage information and run Enterprise Applications on Grids using Oracle Database 10*g* and Oracle Application Server 10*g*. Both Oracle Database 10*g* and Oracle Application Server 10*g* can be managed in a Grid Computing environment using Oracle Grid Control. Together these products address the challenges faced by IT organizations today:

- *Radically Reduce or Eliminate Excess Computing Capacity* by automatically load balancing workloads to use spare capacity efficiently, eliminating "islands of computation"
- *Provide Inexpensive Capacity Growth* by adding capacity on-demand in low-cost modular units

- *Radically Lower Cost of Management* by centralizing administration of the resources in a Grid and automating provisioning and administration tasks across these resources

Oracle Application Server 10*g* and its Benefits

Oracle Application Server 10*g*, the next generation of Oracle's Integrated Software Infrastructure for Enterprise Applications, has been designed to enable Grid Computing. It has been designed to effectively pool together large numbers of low-cost servers to create a virtual computing resource across which enterprise applications can be transparently distributed to use capacity efficiently, at low cost, and with high availability. Any existing application that runs on Oracle Application Server can transparently take advantage of Grid Computing without any changes. Service-Oriented Applications will find additional benefits when deployed in a Grid. Oracle Application Server 10*g* provides a number of Grid Computing features, most importantly:

- *Radically Reduce or Eliminate Excess Computing Capacity* through Policy-Based Resource Management; Metrics-based Workload Management; and a variety of advanced back-up, disaster recovery, and clustered fail-over solutions to provide maximum availability in a Grid.
- *Provide Modular, Inexpensive Capacity Growth* through Automated Installation, Configuration, and Software Provisioning (including both software cloning and patch management) across hundreds of nodes in a Grid.
- *Radically Lower Cost of Management* and eliminate human errors in management through Centralized Systems Monitoring, Unified Application Server Cluster Management (including Cluster Monitoring, Cluster Optimization, and Cluster-wide Application Deployment), and centralized Identity Management across a Grid.

PERFORMANCE, SCALABILITY, AND QOS ON THE GRID

IT managers and software developers face strict performance, scalability, and quality of service (QoS) requirements for Web-based applications, whether or not their applications are deployed in a Grid environment. To be successful, IT administrators must protect against poor response times and system outages caused by peak loads; at the same time, they must also contain costs. For their part, successful application developers must consider performance and scalability at design time, not as a post-deployment tuning exercise.

Today's IT managers and software developers face three main performance-related challenges:

1. Dynamic, Web-based applications are compute-intensive, yet IT personnel are asked to do more with less
2. Unexpected traffic surges can cause delays and outages, yet planning for peak loads is cost-prohibitive

3. Users are demanding sub-second response times, yet visibility into end-user service levels is poor

The Dynamic Content Dilemma

While fast response times are crucial for revenue generation, companies must also retain users and control costs if they want to become (or remain) profitable. The problem of user retention is most often addressed by delivering dynamic, personalized content to each user. Yet as the Aberdeen Group aptly warns, customizing each Web page raises the ever-present cost specter. "Customized content generated at the moment of request often translates into high hardware, software and data management costs. Without an adequate infrastructure, customized content could do more harm than good – degrading the firm's brand, slowing customer service, or, in the worst case, causing a customer to turn away and go to a competitor."[1]

One alternative is to design Web pages using only static content. In terms of computation and resource utilization, static content is easy to generate and deliver, and most static Web sites will perform adequately under heavy load. One of the problems with this approach is that without a dynamic, database-driven infrastructure, content management becomes difficult. Every time an update is made, the static Web site has to be redesigned and republished.

While static content may have been sufficient for first-generation Web design, today's e-businesses must offer customers a more compelling user experience. E-business is anything but static. Companies must exchange data in real-time with other companies, and customer retention demands an interactive, one-to-one relationship with consumers. For these reasons and more, database-driven, dynamically generated content is at the heart of today's Web-based application architectures.

Despite its prominent architectural role, dynamic content generation poses significant challenges for e-business managers who struggle to control costs without sacrificing performance. To paraphrase the Aberdeen Group report cited earlier, generating content on the fly involves several steps that inevitably utilize a large amount of computing power and, under load, can lead to performance bottlenecks.

The typical steps can be summarized as follows. In order for a Web browser to request content from a Web server, the user's client machine must first connect to the Web server. Once connected, the browser's HTTP request has to be parsed by the Web server. The HTTP request may contain parameters and header information that must be passed to a "presentation" mechanism for appropriate content retrieval and formatting. If the requested content requires formatting by a servlet or JSP, then the Web server must connect to a runtime environment (e.g., Apache Tomcat), which may be running on a separate machine. Once invoked, the servlet may instantiate a number of Java classes which query a database, requiring further network connections, since the database is typically running on a dedicated machine of its own. Ultimately, the formatted content is returned to the Web

[1] Aberdeen Group, *Accelerating Web Site Performance by Caching Dynamic Content*, January 2001.

server, which finally delivers the full page to the browser. The browser may then initiate several subsequent HTTP requests to retrieve the embedded (static) page elements.

For any given Web-based application, a large percentage of dynamic requests are made for identical or similar content, meaning that *repeated requests for dynamic pages place a significant strain on application infrastructure.* The higher the user load, the more accentuated the inefficiencies. To deal with this challenge, IT managers need state-of-the-art caching and compression mechanisms that enable more efficient use of existing, low-cost hardware.

Peak Loads, Flash Crowds, Lights Out

Considering the runtime, disk I/O, and network operations just described, one can appreciate how the computational overhead associated with building pages "on the fly" for thousands of concurrent users can result in increasing delays and failures in data delivery. Many enterprise data centers try to counter this problem by adding more servers to their existing architectures. In order to sustain performance under ever-mounting loads, a successful CIO may need to multiply her investment in hardware and software by a factor of ten or more on a yearly basis. Capital outlay of such inordinate magnitude is not unheard of, and many online businesses and enterprise IT departments have disproportionately large data center infrastructures for this reason.

But capacity planning is not a simple task. Even very large application infrastructures sometimes fail to produce the scalability, availability, and response time service levels demanded by a growing customer base, as illustrated by a number of high-profile Web site outages over the past few years. Not only is the number of Internet and enterprise users increasing, but also the volume and frequency of denial-of-service attacks are on the rise.

The days when hardware companies offered free equipment to high-profile dotcom startups are long gone. In today's profit-conscious economy, the ultimate challenge for software developers and systems managers is to make dynamic applications perform and scale while at the same time lowering the infrastructure costs required to meet capacity targets. Caching and compression can help, but IT managers also need sophisticated workload management tools at their disposal in order to cost-effectively contain flash crowds and other unexpected surges in traffic.

The End User is King

End users *expect* Web-based applications to be fast. A 2001 Zona Research report entitled *The Need for Speed II* highlights the importance of adhering to what has become known as the "eight second rule": if consumers cannot download a Web page within eight seconds, they may jump to a competitor's site or take their business off the Web entirely. With regard to the economic impact of slow Web sites, Zona finds "more than $25 billion [annually] in potential lost business due to Web performance issues."[2]

[2] Zona Research, *The Need for Speed II*, April 2001.

"…when the numbers are all added up, this amounts to a stunning condemnation of the Web site content design practices of many of the largest electronic retailing sites in the country. With an average page response time of more than 17 seconds running across the group (as seen by 56kbps modem users), it is clear that retailers are much more concerned about delivering fancy graphics and dynamic banner ads to Internet buyers than they are with delivering acceptable performance levels to potential buyers. The huge bulk of dialup modem users will continue to abandon nearly one out of every two buying attempts on retail Web sites. The total dollar value of that abandonment could mount to over $25 billion this year."

Zona Research, *The Need for Speed II*

Economic impact aside, eight seconds is actually a long time to wait for a Web page to load. Business applications users, in particular, are accustomed to sub-second response times serviced by legacy client/server and terminal/mainframe applications.

And yet most developers and IT managers have *no idea* what service levels their end users are experiencing. This is because existing monitoring tools and services are insufficient:

- *The data provided by services and tools is misleading.* End users do not sit in service providers' data centers. And client software, hardware, connection speed, and physical location vary from user to user.
- *The data is incomplete.* Traditional monitoring solutions only monitor some of the pages some of the time. The solutions do not monitor the entire application all of the time. Nor are they capable of tracking the experience of individual, named users, such as the company President or CEO.
- *The solution scope is limited.* Too often, intranet applications are simply ignored. Many IT shops stress test an application before putting it production, but then wait until users complain before they realize there is a performance issue with a live system.

In this equation, the quality of service experienced by actual end users is lost. IT managers and developers need better tools to gather real-world application performance metrics. Without this knowledge, performance problems remain unknown and unresolved, to the detriment of the ever-important end user.

INTRODUCING ORACLE APPLICATION SERVER WEB CACHE

Caching, compression, and workload management are key technologies that promise to alleviate the computational and economic burdens faced by today's overstrained application infrastructures. Nearly all applications benefit from having Web content cached on nodes between the consumers searching for content and the content source itself, known as the origin server. Compressing the content not only saves bandwidth but can significantly improve response times for remote users or users on low-bandwidth networks. And workload management ensures quality of service during peak loads and unforeseen traffic spikes.

The main challenges of dynamic Web page caching are the volatility and variation of the content. Some parts of a page may change more rapidly than others, and for the same page, individual users may see different content according to personal preferences or roles. To address these challenges, Oracle Application Server contains innovative Web caching technology – OracleAS Web Cache – designed to increase throughput, shorten response times, boost scalability, ensure reliability, and reduce infrastructure costs. Often referred to as a "reverse proxy", OracleAS Web Cache is a state-of-the-art *server acceleration* solution, yielding throughput rates of several thousand requests per second on commodity hardware.

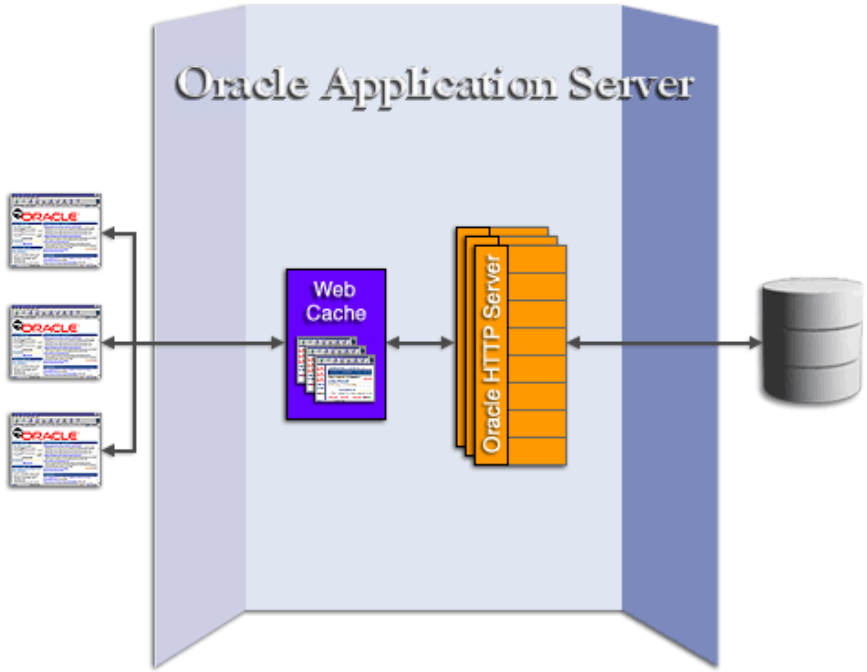


Figure 1: Deployed between browser clients and application Web servers, OracleAS Web Cache uses caching and compression technologies to speed the delivery of Web-based applications.

OracleAS Web Cache 10*g* is the software industry's leading application acceleration solution. Designed for enterprise grid computing, OracleAS Web Cache leverages state-of-the-art caching and compression technologies to optimize application performance and more efficiently utilize low-cost, existing hardware resources. Built-in workload management features ensure application reliability and help maintain quality of service under heavy loads. And new in this release, end-user performance monitoring features provide unparalleled insight into end-user service levels.

Key OracleAS Web Cache features include:

- Efficient use of low-cost, existing hardware
- Workload management and reliability
- End-user experience management
- Flexible deployment options and integration with the Oracle stack

The following pages will address the features listed above, making sure to highlight the benefits of each. Briefly, the benefits of OracleAS Web Cache can be measured by dramatic improvements in the following areas:

- *Resource usage* – higher throughput and scalability
- *User experience* – faster response times without sacrificing personalization
- *Availability* – intelligent workload management
- *Productivity* – no need to roll your own cache means faster time-to-market
- *Bottom line* – reduced infrastructure load translates into cost savings
- *Intelligence* – better visibility into end-user service levels

EFFICIENT USE OF LOW-COST, EXISTING HARDWARE

A high percentage of Web application requests are made for identical or similar content, meaning that repeated requests for dynamic pages place a significant strain on application infrastructure. The more users on the system, the more apparent the

inefficiencies become. To make more efficient use of existing, low-cost hardware, IT managers can turn to the intelligent caching and compression services of OracleAS Web Cache.

Static and Dynamic Content Caching

With its unique ability to cache both static and dynamically generated Web content, OracleAS Web Cache takes the pressure off busy Web sites and Web-based applications by storing frequently accessed pages in memory, eliminating the need to repeatedly process requests for those pages on mid-tier servers and databases.

In a typical deployment, OracleAS Web Cache is logically positioned in front of a farm of application Web servers, caching their content, and providing that content to Web browsers that request it. Acting as the virtual server for the application, OracleAS Web Cache intercepts all requests made to the Web site. Web Cache either returns the requested objects if they are present in the cache or, upon a "cache miss", forwards the request via HTTP to one of the application Web servers in the server farm. The application Web server, also known as the origin server, generates the response and sends it back to OracleAS Web Cache. The response may be generated using JSP, Servlet, CGI, PHP, PL/SQL, ASP, or any other technology capable of outputting content for transmission over HTTP. OracleAS Web Cache stores any cacheable objects returned by the origin server and forwards the response to the browser.

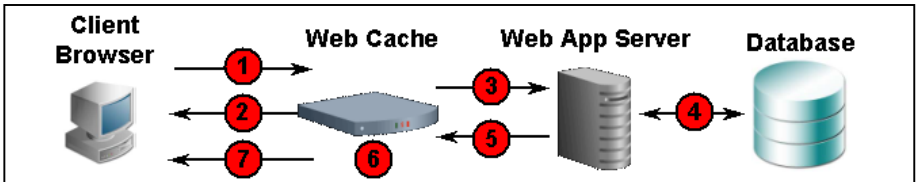
"With OracleAS Web Cache serving up dynamic content, the performance increase is great enough that Specialized can significantly reduce the need for increased bandwidth and server horsepower at the back end. And less processing – not just on the database servers, but also by Web server applications – means the customer has more time to get out and ride, which is what the Specialized experience is all about."

Ron Pollard, CIO, Specialized Bicycle Components, Inc.

Caching policies are configured by an administrator using "caching rules" and require no application modifications. However, developers may also opt to express cacheability policies in response headers. As OracleAS Web Cache becomes populated, it is able to serve more of the requested content itself. This, in turn, frees up processing resources on the Web, application and database servers.

OracleAS Web Cache is truly unique in the marketplace. Products and services that cache static content are typically unable to serve dynamic content because they lack the means to manage the consistency of Web pages vis-à-vis the data sources used to create them. When deployed as server accelerators, traditional (naïve) caching products force customers to rely on expensive and complex content propagation tools to update their caches with new content. Furthermore, these content propagation tools cannot handle the volume and frequency of content updates demanded by today's dynamic Web-based application architectures.

In contrast, OracleAS Web Cache was built from the ground up to cache volatile Web content. OracleAS Web Cache employs event- and time-based *invalidation* mechanisms to maintain consistency with origin data sources, such as file systems, content management tools, databases and content feeds. Using a combination of administrative commands, database triggers and programmatic interfaces, site administrators and application developers can purge cached content as frequently as the original content changes.



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

Figure 2: How Web Cache works.

"With OracleAS Web Cache serving up dynamic content, the performance increase is great enough that Specialized can significantly reduce the need for increased bandwidth and server horsepower at the back end. And less processing – not just on the database servers, but also by Web server applications – means the customer has more time to get out and ride, which is what the Specialized experience is all about."

Ron Pollard, CIO, Specialized Bicycle Components, Inc.

In addition to intelligent consistency management, an effective dynamic caching solution must address content disambiguation, session management and personalization. Disambiguating content is essential for properly storing and delivering documents and maintaining high cache hit rates. URLs, normally the unique identifiers for static content caches, are often insufficient for dynamic content due to request-specific content generated by applications. For example, a browser-aware application may generate different responses for different browsers for the same URL.

Likewise, Web-based applications often need to maintain session state (such as shopping carts) for every user. Some applications embed the session state within the URL and hyperlinks as an alternative to cookies. A naïve cache would interrupt the session establishment during cache hits and prevent new session creation. Additionally, pages with embedded session state become unique for each session and defeat simple caches.

Unlike traditional static caches, OracleAS Web Cache supports a number of advanced caching policies for dynamically generated content, such as caching multiple-version documents for the same URL, and session-aware rules for pages containing session information.

○	23	Regular Expression	^/robots\.txt$	GET	N/A	N/A	Don't Cache
○	24	Regular Expression	^/elog/*	GET with query string	N/A	N/A	Don't Cache
○	25	Regular Expression	\.pdf$	GET, GET with query string	N/A	N/A	Cache
○	26	Regular Expression	\.PDF$	GET, GET with query string	N/A	N/A	Cache
○	27	Regular Expression	\.pdf\?Cache$	GET with query string	N/A	N/A	Cache
○	28	Regular Expression	\.html?$	GET, GET with query string	N/A	N/A	Cache
○	29	Regular Expression	\.htm?$	GET with query string	N/A	N/A	Cache
○	30	Regular Expression	/$	GET, GET with query string	N/A	N/A	Cache
○	31	Regular Expression	\.(gif|jpe?g)$	GET, GET with query string	N/A	N/A	Cache

Figure 3: Sample static caching rules for www.oracle.com, as displayed in the Web Cache Manager administrative user interface.

When configuring OracleAS Web Cache, administrators use caching policies to specify which content to cache and which content not to cache.[3] OracleAS Web Cache supports caching of static content, such as GIF and JPEG images, as well as non-transactional content created using dynamic page generation technologies, such as JavaServer Pages (JSP), Active Server Pages (ASP), PL/SQL Server Pages (PSP), Java Servlets, Common Gateway Interface (CGI), PHP, Python, ColdFusion, and many others.

Note that dynamically generated content was, until very recently, considered non-cacheable. This is in part due to the ephemeral nature of this content; another reason is that it is difficult to map HTML-formatted content to the relational tables and materialized views used to generate the content. As discussed later in this paper, the invalidation and capacity heuristics mechanisms supported by OracleAS Web Cache help circumvent this difficulty quite nicely.

[3] Note that most caching policies can be expressed at development time using surrogate-control headers, making the content self-describing and the configuration of rules unnecessary. Rules expressed in surrogate-control headers override those expressed in the configuration.

○	6	Regular Expression	^/cgi-bin/*	GET, GET with query string	N/A	N/A	Don't Cache
○	7	Regular Expression	^/forums/i_system_annc.jsp	GET, GET with query string	N/A	N/A	Cache
○	8	Regular Expression	^/forums/i_forum_annc.jsp	GET, GET with query string	N/A	N/A	Cache
○	9	Regular Expression	^/forums/i_droplist.jsp	GET	N/A	N/A	Cache
○	10	Regular Expression	^/forums/i_forumdisplay_esi.jsp	GET, GET with query string	N/A	N/A	Cache
○	11	Regular Expression	^/forums/i_dateview.jsp	GET, GET with query	N/A	N/A	Cache

Figure 4: Sample dynamic caching rules for www.oracle.com, as displayed in the Web Cache Manager administrative user interface

Examples of non-transactional pages that are dynamically generated include:

- product catalogs, where information on pricing and inventory might vary from one moment to the next
- auction views, which must be regenerated after each successful bid is processed
- search results, which can change as site content is added and removed
- portal pages, which display personalized content for each user

Specifying caching policies is a declarative process. When an administrator assigns caching rules, he specifies the *regular expression* for either a specific URL or a collection of URLs and whether or not the content matching those URLs should be cached. Examples of content that administrators would typically declare non-cacheable include update transactions, personal account views, and so forth.

In addition to the URL, administrators can specify optional selectors for more fine-grained caching rules. These additional selectors include the HTTP request method (`GET`, `GET` with query string, or `POST`) and, if `POST` is selected, the HTTP `POST` body of the documents.

If no caching rules are specified or if no rules match a particular request, then OracleAS Web Cache behaves just as traditional proxy caches do; that is, it relies on standard HTTP header information to determine what is cacheable. Naïve proxy caches typically only cache static content for reasons discussed earlier in this paper.

Invalidation for Cache Consistency

Clearly, caching dynamically generated Web content requires a set of features specifically designed for this purpose. One such requirement is an efficient means of maintaining consistency between the content in the cache and the content in the origin data repositories. OracleAS Web Cache uniquely supports invalidation and patent-pending performance assurance mechanisms that allow it to maintain consistency with origin data sources, even under heavy loads or when content is changing frequently. Using a simple combination of expiration policies and invalidation messages, a content administrator or application developer can purge cached content as frequently as the original content changes, thereby ensuring site accuracy and rapid response times. Because of its simplicity and openness, the cache consistency model introduced by Oracle is easier to use, more flexible and less expensive than that of any other solution on the market.

Expiration Policies

Expirations are used primarily in cases where content changes can be accurately predicted. A Web site that displays weather forecasts and current climate conditions is an example of an application that would benefit from invalidation using expiration policies. The Web pages relating to the climate conditions could be set to expire 30 minutes after the pages were created, thereby ensuring that customers never receive outdated information.

Invalidation Messages

For less predictable, more frequently changing content, a general message-based mechanism is needed for maintaining cache consistency. The invalidation message format and grammar must integrate readily with existing applications and content repositories, including databases, custom scripts, application logic, syndication servers and content management tools.

Oracle's invalidation protocol is based on open standards, namely HTTP and XML. As such, it can be rapidly integrated into application logic and existing content management systems. And because it is both non-proprietary and simple, Oracle's cache consistency model is easier to use, more flexible and less expensive than that of any other solution on the market.

OracleAS Web Cache invalidation messages are HTTP POST requests that carry an XML payload. The contents of the XML message body tell the cache which URLs to mark as invalid. Invalidation messages can be sent using any of the following methods:

- Manually, using either Oracle Enterprise Manager Application Server Control, OracleAS Web Cache Manager or Telnet (requires no code changes)
- Programmatically, using database triggers, scripts or application logic

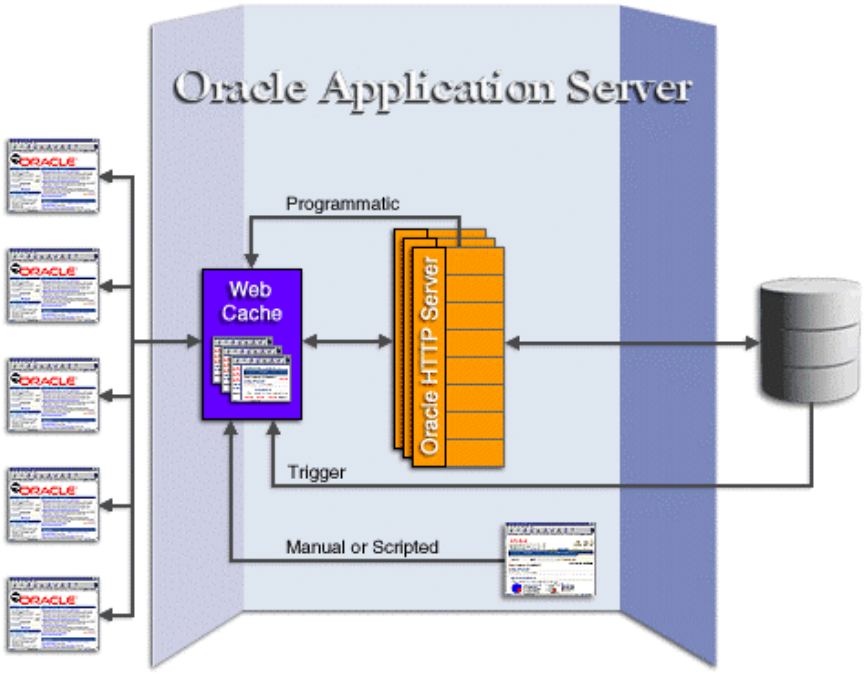


Figure 5: OracleAS Web Cache employs an XML/HTTP-based invalidation message model that allows easy integration with application logic, database updates, custom scripts, and off-the-shelf content management solutions.

With only moderate code changes, almost any application can automatically generate the XML/HTTP messages required for invalidating cached content. *For the sake of convenience, OracleAS Web Cache ships with Java and PL/SQL APIs that enable developers to embed invalidation logic directly into their applications.* Developers who wish to write their own invalidation routines can refer to the Document Type Definition (DTD) for invalidation requests and responses, which is installed with the product and is described in detail in the *OracleAS Web Cache Administrator's Guide*.

Partial-Page Caching and Personalized Page Assembly

Full-page caching becomes inefficient when pages are highly customized for each user. Personalized pages are often unique aggregations of common content. A single page may combine content fragments with different caching properties. Some are cacheable, some are not; some expire soon, some hardly expire. Some content expires at prescribed intervals, while other content expires as the result of an external event. The combination also creates "cache space explosion" for traditional full-page static caches due to the redundant caching of common fragments. There is little value in caching a personalized page as a whole page since the page can only be delivered to the user or users for whom it was designed, and only while all the content is valid.

For this reason, Oracle collaborated with partners to create a standard partial-page caching language called Edge Side Includes (ESI). As the first application server to support the innovative ESI specification, Oracle Application Server leads the industry with its ability to perform partial-page caching, personalization and

"ESI will help to solve the scalability problem with sites that create unique page configurations for such things as personalization. Without functionality such as ESI, the burden to dynamically generate pages falls solely on the origin server."

Neal Goldman, Research Director at The Yankee Group

dynamic content assembly from both the *edge* of the corporate network and the *edge* of the Internet.[4]

Invented by Oracle and Akamai, and endorsed by other leading players[5] in the Internet infrastructure and services industries, ESI is an XML-style specification that has been published as a W3C Note. Web developers use ESI markup to identify fine-grained page elements, called content fragments, for dynamic assembly at the network edge.

"Our goal is to continue to extend the richness and functionality of our site while ensuring the highest levels of performance. That requires being able to deliver dynamic Web content in a reliable manner, while also maximizing the efficiency by which constantly changing and personalized content reaches our end users. We are in support of an open specification that enables application server infrastructure and content delivery capabilities to interoperate for, not only a good user experience, but reduced total cost of ownership."

Eric Schvimmer, Vice President of Technology at Washingtonpost.com

The basic structure a developer uses to create dynamic content in ESI is a template page containing HTML fragments. The template consists of common elements such as a logo, navigation bars, and other "look and feel" elements of the page. The HTML fragments represent dynamic subsections of the page. (See Figure 6.)

Traditional Web architectures tend to be expensive because they require a single infrastructure for the creation, assembly and delivery of content. ESI, on the other hand, separates a Web site's content generation mechanism from its assembly and delivery mechanisms. The ability to assemble dynamic pages from individual page fragments and deliver them from inexpensive edge servers, such as OracleAS Web Cache, means that only non-cacheable or expired fragments need to be fetched from the origin Web servers. Thus, the ESI model reduces the need to retrieve complete pages and decreases the load on a Web site's content generation infrastructure.

With ESI,

- e-businesses can now develop highly personalized Web-based applications that are assembled at the edge of a company's main data center and remote offices (with OracleAS Web Cache) and/or at the edge of the public Internet (with ESI-compliant Content Delivery Network services), for improved performance;
- the aggregation and assembly of content in edge servers dramatically reduces the cost of infrastructure required to deliver fast, scalable and fault-tolerant applications.

[4] OracleAS Web Cache is a full-fledged ESI processor. Use of a content delivery network like Akamai is ***not*** required in order to take advantage of ESI and partial-page caching.
[5] ESI endorsers include Oracle, Akamai, ATG, BEA Systems, Circadence, Digital Island, IBM, Interwoven, Macromedia, OpenMarket, ATG, Network Appliance, Mirror Image, Digital Island, Silverstream and Vignette. Please visit www.esi.org for more information.

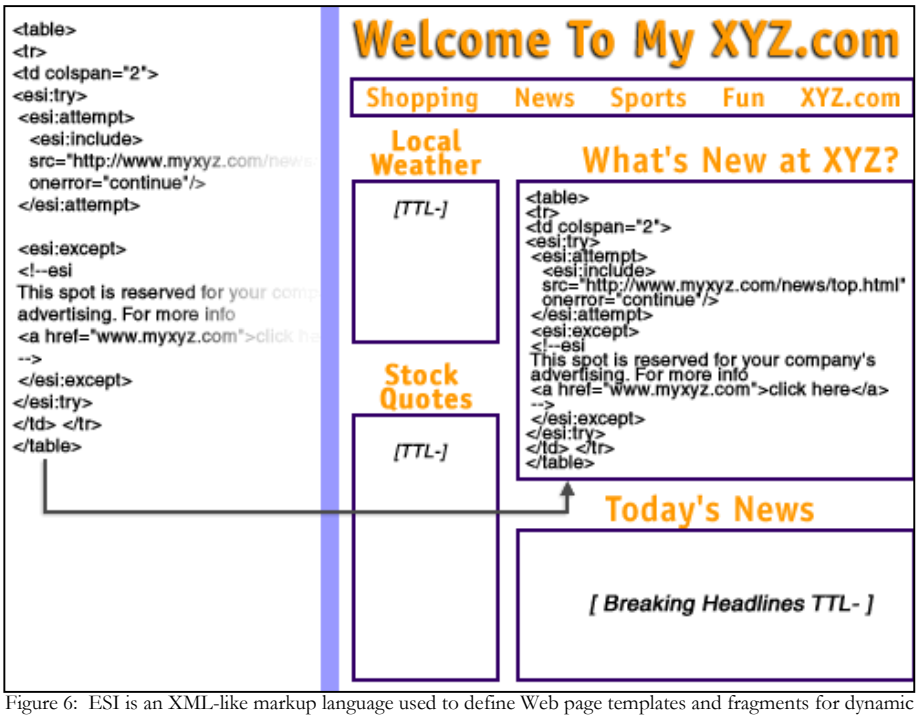


Figure 6: ESI is an XML-like markup language used to define Web page templates and fragments for dynamic assembly at the network edge. JESI is a specification and custom tag library that JSP developers can use to automatically generate ESI code.

ESI for Java (JESI) – JSR 128

To encourage rapid adoption of ESI by Java developers, the creators of ESI have also published the Edge Side Includes for Java (JESI) specification. JESI is actively making its way through the Java Community Process standards body as JSR 128. As a specification and custom JSP tag library that developers can use to automatically generate ESI code, JESI facilitates the programming of JavaServer Pages (JSPs) using ESI.

While developers can always use ESI tags directly within their JSP code, JESI represents an easy way to express the modularity of JSPs and the cacheability of those modules, without requiring developers to learn a new programming syntax. JESI generates the appropriate ESI tags and headers in the JSP output that instruct ESI processors, such as OracleAS Web Cache, to cache (or not) templates and fragments for the appropriate duration. JESI also facilitates the partial execution of JSPs when an ESI processor requests fragments and templates. To further simplify the invalidation process for JSP developers, the JESI custom tag library supports automatic invalidation for pages and page fragments through easy-to-use <jesi:invalidate> tags.

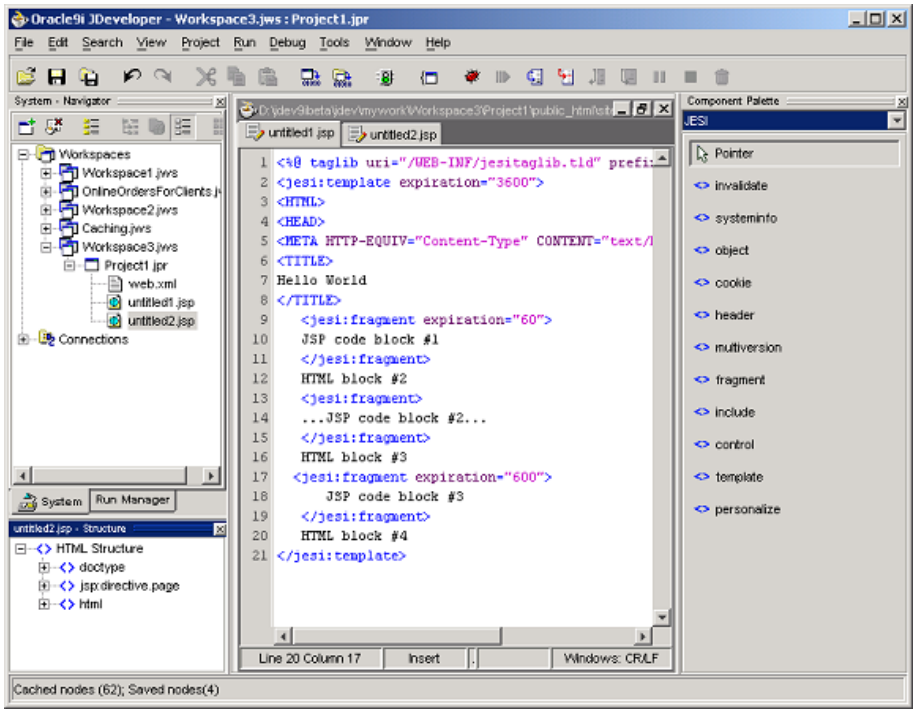


Figure 7: With its JESI component palette and ESI Servlet Filter extension, JDeveloper is a friendly environment in which to develop and test cache-enabled JSPs.

Both OC4J and Oracle JDeveloper support the use of ESI and JESI, and both currently ship with the JESI tag library. An easy-to-use ESI Servlet Filter extension is also available for JDeveloper. The filter allows developers to create JSPs with ESI and JESI tags (using the component palette), and test them within the development environment. Without the extension, JSPs developed with ESI or JESI will not be rendered properly when previewed in JDeveloper.

Automatic Compression

OracleAS Web Cache offers automatic compression of dynamically generated content. On average, using the standard GZIP algorithm[6], OracleAS Web Cache is able to compress text files such as HTML and XML by a factor of 10. Because compressed objects are smaller in size, they require less bandwidth to transmit and can be delivered faster to browsers. With compression, everyone benefits: service providers, corporate networks and content providers reduce their transmission costs, while end-users enjoy more rapid response times.

[6] All major browsers since 1997 support GZIP expansion.

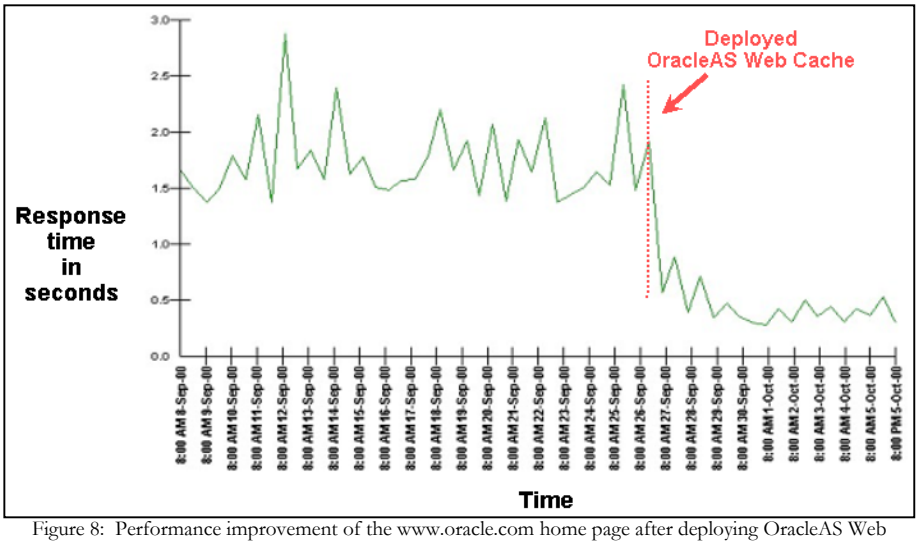


Figure 8: Performance improvement of the www.oracle.com home page after deploying OracleAS Web Cache in September 2000, as measured by Keynote Systems, Inc. Response times fell from an average of approximately 2 seconds to approximately 0.5 seconds.

"Oracle Application Server's caching technology delivered a 15-fold improvement in Web page response times, and enabled us to immediately triple the maximum capacity of our Web site to approximately 600,000 users per day, right out of the box."

John Perkins, Vice President of Engineering for Rentals Inc.

Most application Web servers on the market are capable of serving compressed pages, but few enable caching of compressed output. With OracleAS Web Cache, compression is a simple "Yes/No" option that an administrator selects when specifying a caching rule. Because OracleAS Web Cache supports regular expression for caching rules, compression can be applied to responses using criteria other than just file extension. Regular expression makes it very easy to select which pages to compress and which pages not to compress, as well as whether or not a particular browser should receive compressed content.

And unlike the typical application Web server, OracleAS Web Cache offers compression and caching for pages that have been dynamically generated. *By caching compressed output, OracleAS Web Cache reduces the processing burden on the application Web server, which would otherwise have to re-generate and compress dynamic pages each time they are requested.*

Linear Scalability on Commodity Hardware

OracleAS Web Cache was designed from the ground up to provide high performance, reliability and scalability on low-cost commodity hardware. A single OracleAS Web Cache instance can be configured to support thousands of concurrent inbound HTTP connections. The throughput (requests/second) that a single cache instance can sustain scales linearly with CPU speed. Additionally, OracleAS Web Cache fully supports the HTTP/1.0 and HTTP/1.1 protocols, including keep-alive. The keep-alive directive reduces the frequency of connection establishment and improves performance and scalability under heavy load.

When clustered, the capacity (i.e., the amount of content stored in RAM) of the cache tier scales linearly, as well. Cache clustering achieves high availability by failing over among cluster nodes, as well as high scalability by utilizing memory and processing power of multiple inexpensive computing hardware units in parallel.

Cache clustering is also discussed in greater depth in the Workload Management section of this paper.

WORKLOAD MANAGEMENT AND RELIABILITY

"The Web caching and load balancing capabilities of the Oracle Application Server have produced significant decreases in page response times and also greatly reduced CPU load on back-end database servers. We find that we can now achieve all the scalability we need in the middle-tier. That means that we don't have to purchase millions of dollars worth of new database server hardware. The bottom line: we're able to save millions of dollars."

Marty Boos, CIO, Digital River

Caching and compression optimize performance and create efficiencies in hardware resource usage, but these technologies in isolation do not address the costly load distribution and capacity planning issues arising from unpredictable traffic patterns. For this reason, OracleAS Web Cache extends the boundaries of content acceleration to include workload management features that enable IT organizations to cost-effectively contain flash crowds and other unexpected surges in traffic.

Web Server Load Balancing, Failover, and Connection Pooling

Most Web sites are powered in part by mid-tier application Web servers running on a collection of co-located computers. This deployment is commonly known as a server "farm" and can vary from as few as two machines to as many as 1,000 or more. Distributing the load across multiple servers provides better scalability and reliability and allows administrators to take machines offline for maintenance without impacting availability. The traffic management function is typically provided by a hardware load balancing device deployed between the browser clients and the farm of Web servers.

Situated between the network load balancer and the Web servers, OracleAS Web Cache introduces a new tier into the traditional Web farm architecture. To reduce complexity and to avoid the cost of purchasing additional load balancing hardware, OracleAS Web Cache includes built-in load balancing and failover detection features to ensure that "cache misses" are directed to the most available, highest performing Web server in the farm. The cache supports both stateless and stateful load balancing mechanisms, including the use of cookies and URL parameters to maintain server affinity when required.

Just as network load balancers do, OracleAS Web Cache can determine when an origin Web server has failed and then automatically redistribute the load over the remaining servers. OracleAS Web Cache periodically checks to see if the failed Web server has returned to a functional state and is capable of serving dynamically generated content. Layer 7 status checking, as this mechanism is often referred to, not only verifies the health of the Web server, but also that of the application logic, database and other repositories used to store and create content. As soon as the failed server returns to operation, OracleAS Web Cache will once again include it in the distribution mix.

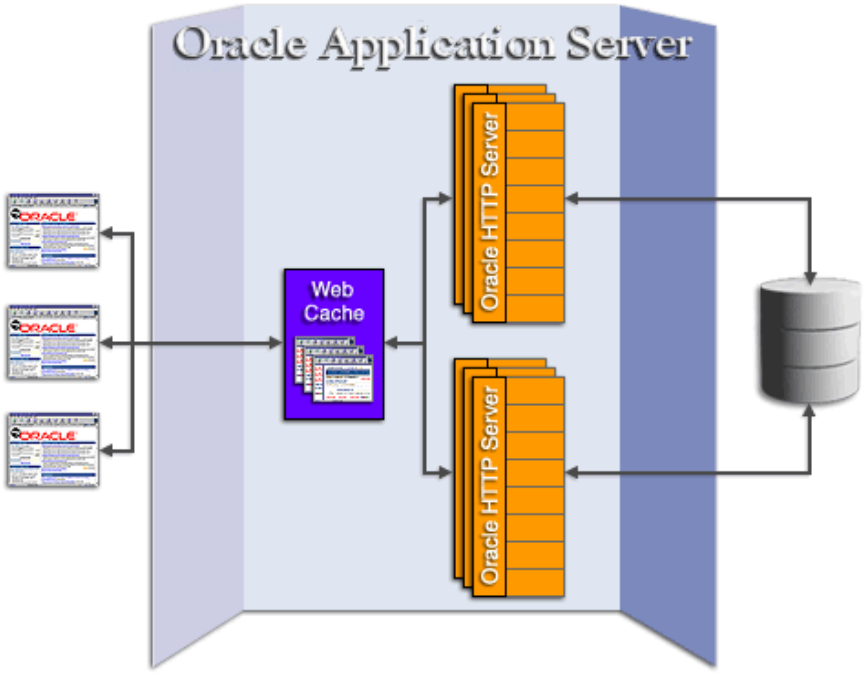


Figure 9: Load balancing and failure detection ensure that cache misses are directed to the most available, highest performing Web server in the farm. The cache maintains a connection pool between itself and the origin Web servers for faster update transactions and retrieval of new or changed content.

Additionally, OracleAS Web Cache maintains a pool of HTTP connections between the cache and the origin Web servers. The connection pool leverages HTTP keep-alive to reduce TCP connection establishment overhead and improve cache miss performance.

Surge Protection and Throttling

Peak loads and request patterns on Web sites are seldom predictable. It is difficult to foresee when requests for new content will generate more traffic than a site's Web servers have capacity to handle. Such traffic spikes or "surges", as they are known, are even more troublesome when the requested content must be dynamically generated or when the content is changing frequently. Because OracleAS Web Cache can sustain orders of magnitude greater throughput than the average Web application server, it provides a layer of defense against such surges.

To prevent an overload of requests on the origin Web server(s), OracleAS Web Cache enables administrators to set a limit on the number of concurrent connections that the origin Web server(s) can handle. When the capacity limit is reached, subsequent requests are queued to wait up to a maximum amount of time. If the maximum wait time is exceeded, OracleAS Web Cache rejects the request and serves an apology, or error, page to the Web browser that initiated the request.

Quality of Service Assurance

OracleAS Web Cache uniquely supports invalidation and patent-pending performance assurance mechanisms that allow it to maintain consistency with origin data sources, even under heavy loads or when content is changing frequently.

Using a combination of expiration policies and invalidation messages, a content administrator or application developer can purge cached content as frequently as the original content changes, thereby ensuring site accuracy and rapid response times.

Fine-grained invalidation (i.e., invalidating a single cached document identified by an exact URL) can be a painstaking process for an administrator or developer. It is particularly difficult to map content stored in rows and columns in a database to content formatted in HTML on a Web page, especially when relational content is partitioned or resides in materialized views. A change to a product table, for example, might affect countless search result pages. For this reason, broad-brush invalidation (i.e., invalidating a large set of documents matching a regular expression) is a welcome addition to an administrator or developer's arsenal, provided that performance can be guaranteed. Only OracleAS Web Cache makes this possible.

OracleAS Web Cache provides administrators with the flexibility to invalidate individual documents or broad ranges of documents, such as the entire set of cached search results. Yet, for all its convenience, this approach to maintaining cache consistency has its challenges. For instance, if a large number of cached documents are invalidated, the retrieval of new documents could result in overburdened origin servers, which do not possess the scalability or throughput capacity of a high-performance cache. *For this reason, OracleAS Web Cache intelligently serves stale versions of invalid objects until the origin servers have the capacity to refresh them. The result is that overall Web site performance remains constant (i.e., quality of service is assured) at the higher throughput levels sustainable by the cache, even with frequent content changes on the origin Web servers and databases.*

OracleAS Web Cache uses patent-pending performance assurance heuristics that determine which objects to serve stale and which objects to refresh immediately, with minimal tradeoff between Web site performance and content consistency. *When faced with the choice of serving some stale content or no content at all, most Web site administrators and application developers opt for the former.* The person or program performing the invalidation can specify the grace period during which a page may be served stale. For situations where consistency is more important than performance, such as in a banking application, administrators and developers may specify which responses *never* to serve stale.

Cache Clustering

Multiple independent OracleAS Web Cache instances can be deployed in order to scale the cache tier horizontally. However, IT managers face two key limitations when deploying independent cache instances:

1. Cache capacity is limited to individual machine resources
2. Hot content is lost if a cache instance fails or is taken offline

To combat these limitations, OracleAS Web Cache features native clustering capabilities that enable multiple cache instances to work together as a single logical cache. Cache clustering is both hardware and operating system independent.

Clustering OracleAS Web Cache instances together increases the capacity of the cache and the number of requests that can be served concurrently.

OracleAS Web Cache intelligently provisions content across a cache cluster. By automatically *partitioning* the application content across multiple cache peers and *replicating* the most popular content among those peers, OracleAS Web Cache can store more content, ensure that hot content is always in cache, and support more concurrent users.

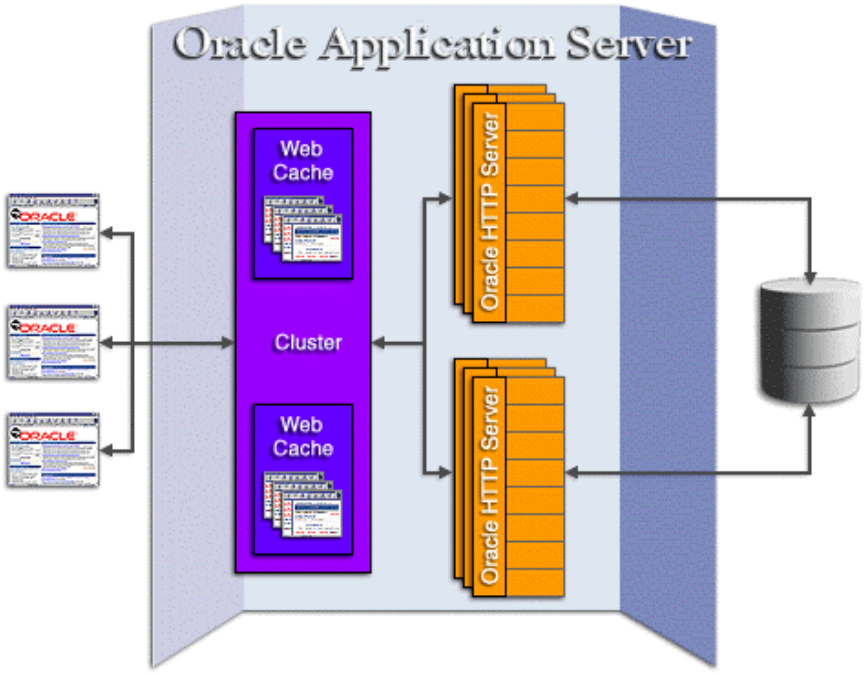


Figure 10: Cache clustering functionality enlarges capacity for content storage and ensures availability of cached content during node failure scenarios.

In addition, cache clustering extends application availability by supporting failure and recovery detection of cache instances. If a cache instance fails, other members of the cluster detect the failure and the cached content is automatically re-partitioned among the remaining cluster peers. Because the most popular content is always resident in each member's cache, the benefit of this failover mechanism is that system administrators can service the "downed" cache at their convenience, without disrupting service. Furthermore, management is simple: a cache cluster uses one set of caching and expiration rules for all cluster members, and invalidation messages are automatically propagated among the cache instances in the cluster.

Partitioning the Web Object Space for Linear Capacity

A cache cluster is a loosely coupled collection of cooperating OracleAS Web Cache instances. It is "loosely coupled" in the sense that global information, including topology and object partitioning, is maintained locally by each individual node, and

more importantly, co-ordination among caches is not based upon active message exchange, as is widely used in other cluster or cache hierarchy schemes (e.g., Internet Cache Protocol). On the other hand, all nodes in a cluster are symmetric. They have equal chances to serve any incoming request, and each node of the cluster has the same global image of cluster topology, configuration and Web object distribution at steady state.

Yet members of a cache cluster configuration are not purely replicas of each other. At cluster startup, the URL namespace is randomly partitioned across the membership and a single node hosts a single partition. That node is called the "owner" of objects residing within that partition space. Instead of a "value-based" partition, where all pages identified by user-specified application criteria (e.g., "all pages for a particular department", "all English-based pages" or "all product pages") are maintained in a particular cache member, OracleAS Web Cache uses a deterministic algorithm to automatically partition site content across cache nodes, resulting in a linear increase in capacity as nodes are added to the cluster. When combined with "content provisioning" (see next section), this deterministic partitioning scheme helps avoid content "brown-outs".

Content Provisioning

In a cache cluster, OracleAS Web Cache uses a hybrid-caching scheme called "content provisioning". *Content provisioning is a combination of both partitioning and replication*. With this scheme, an object is cached by its owner node, as well as cached "on demand" by all other nodes. Because the network load balancer distributing requests across the cache cluster is unaware of the ownership partition, cluster members will receive a percentage of requests for non-owned content, resulting in peer (i.e., on demand) requests to owner caches. Peer caches tend to be greedy, meaning they will keep cacheable, non-owned content in their local repositories if they have the capacity to do so. *In this way, popular content is replicated across the cluster, yielding a higher cache hit rate at any single node, as well as higher availability of cached content in cases of individual node miss or node failure.*

Due to the replication of content, the tradeoff is a less effective usage of the combined capacity of the whole cluster. The solution to this issue is to prioritize ownership and demand-based replication caching in terms of cache replacement (i.e., "garbage collection"). This prioritization is leveraged by a heuristics mechanism. The heuristics mechanism automatically achieves an optimization of both cache capacity usage and hit rate of individual nodes. More popular objects tend to populate more cache nodes and remain in those nodes longer, while less popular objects are likely to be cached only by their owner nodes and flushed due to the heuristics applied at cache replacement.

Failure Detection and Auto-Restart for Hands-Off Manageability

Detection of node failures in a cache cluster is accomplished either by a separate monitoring mechanism (i.e., a process monitor or "watchdog" process) or by inter-cache, on demand requests by individual nodes. At detection of any node failure or removal, ownership needs to be redistributed to accommodate the topology

change. Since the ownership image is maintained locally on each member, a deterministic algorithm is employed to recalculate the partitioning of the URL namespace. This ensures the synchronization of the partitioning image at each node once the cluster has reached a steady state for a given topology. In addition, a process monitor automatically restarts failed cache instances, thereby improving availability and manageability of the cluster.

On-line Reconfiguration Without Loss of Service

To provide uninterrupted service, OracleAS Web Cache supports on-line reconfiguration of the entire cache cluster using a rolling upgrade and restart mechanism. This is implemented on top of an on-line topology change and configuration propagation mechanism. Briefly speaking, cluster nodes check each other's configuration version during a request for a demand-based object and update the topology image if a configuration mismatch is detected. A mismatch segments the cluster into two, one with the new configuration and the other with the old configuration. When an administrator propagates the configuration change to each member and initializes the rolling restart mechanism, the new configuration will be deployed. A newly started node has the new topology view of the cluster. Eventually, it will demand objects from all other nodes, which will have the chance to re-examine its configuration version and add it to their topology image if the configuration matches. In this way, nodes with old configurations leave the cluster and then re-join with new configurations, until all nodes have been restarted and the new cluster is complete. Throughout this process, there is always at least one running node.

Reduced Load on the Origin Servers

At various times during the operation of OracleAS Web Cache, cacheable document requests must be forwarded to the origin Web, application and database servers in order to be "refreshed". These so-called "cacheable miss" or "refresh" requests can occur during cache startup population, after a document has been invalidated, after document expiration, or if a document has been removed from the cache for capacity reasons. With a collection of non-clustered cache instances, refresh requests for sufficiently popular cacheable pages will be sent by *each* of the caches. For instance, suppose `/index.html` is a popular cacheable document and there are four non-clustered caches. Each of these caches will, at least once, request this document from the origin servers. With a cache cluster, in contrast, only the owner cache will make the request to the origin servers. All the other cluster members will request the document from the owner. Therefore, when buffered by a cache cluster, the origin servers are able to devote more time to processing non-cacheable, transactional requests, and less time to refreshing cacheable requests.

END-USER EXPERIENCE MANAGEMENT

Workload management helps maintain system availability and service levels, yet most developers and IT managers may not always know what service levels their end users are actually experiencing. This is because existing monitoring tools and services are insufficient.

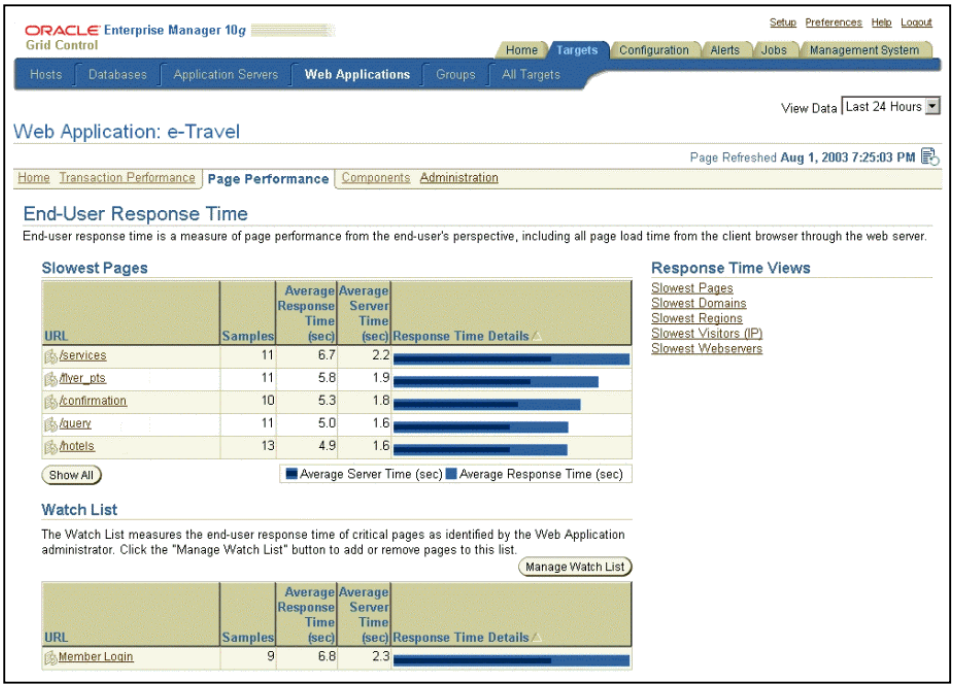


Figure 11: End-User Performance Monitoring statistics, as reported in Oracle Enterprise Manager. Administrators can view the response times experienced by end-users as they access the URLs of a Web application.

Most end-user and Web application monitoring tools require that the administrator manually configure the URLs they wish to monitor. Consider that a Web site like www.oracle.com has over 80,000 URLs. It is, in practice, virtually impossible for an administrator to actively monitor the entire site. Together, OracleAS Web Cache and Oracle Enterprise Manager make what has been extremely impractical and time-consuming into a simple task.

Administrators can configure OracleAS Web Cache to measure end-user response times for individual URLs, sets of URLs, or even entire Web-based applications, regardless of whether the URLs are cached. For each instrumented request, the complete user experience is recorded, from the time a user clicks on a link until the time the page fully renders in the user's browser. The raw measurements are collected in the OracleAS Web Cache access logs. This data is then aggregated,

cleansed and analyzed by Oracle Enterprise Manager using its Application Performance Management (APM) functionality.

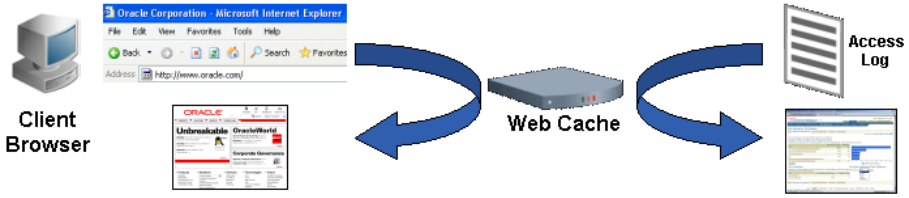


Figure 12: For each instrumented request, the complete user experience is recorded, from the time a user clicks on a link until the time the page fully renders in the user's browser. Raw measurements are collected in access logs; the data is then cleansed and analyzed by Oracle Enterprise Manager.

Visitor tracking ensures that key customers, CEO's, and all other important visitors are receiving adequate response times. Reports show where traffic originates and how much traffic stems from a particular location, so administrators can quantify the impact of performance problems. This information is invaluable when prioritizing repairs for system problems, empowering administrators to focus on problems having the largest impact while placing less critical issues at a lower priority. The Analyze functionality lets you view detailed reports in context by group, URL, domain, visitor, or application as well as in a daily, weekly, or monthly context. Further drill-downs provide administrators with response time and load distribution information to help balance Web server resources.

The result is the most comprehensive and accurate end-user performance monitoring system in the industry, providing click-to-render measurement and reporting in a manner completely transparent to the end-user.

- Enterprise Manager can be configured to notify administrators proactively when response thresholds are violated.
- Additionally, this comprehensive performance data is rolled up and stored in the Enterprise Manager Repository, enabling robust reporting on the end-user experience.
- Response times for individual users can also be tracked to ensure that key customers, CEO's, and all other important visitors are receiving adequate quality of service.

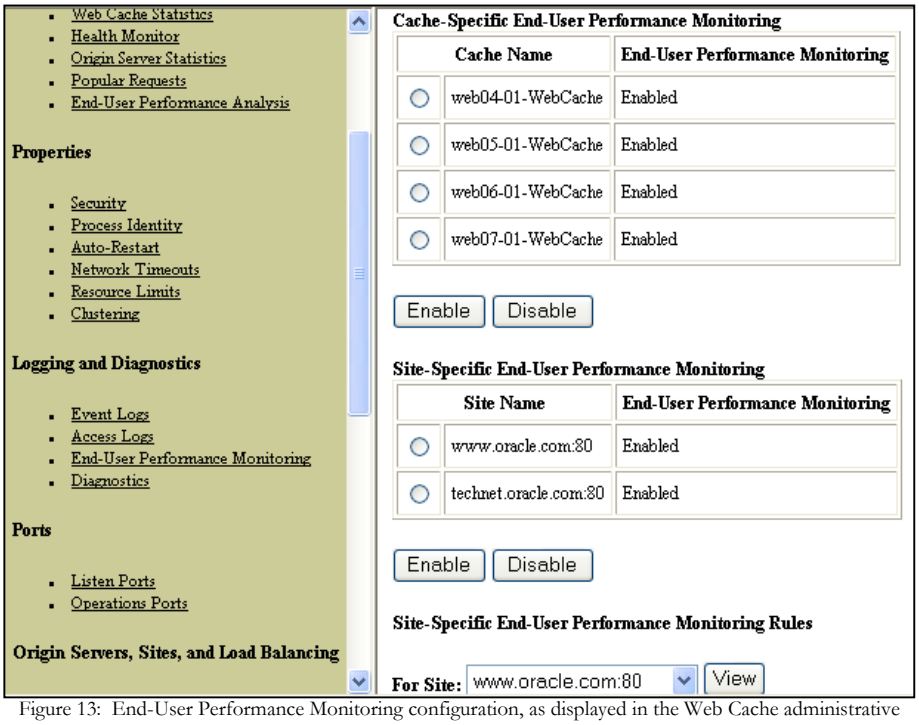


Figure 13: End-User Performance Monitoring configuration, as displayed in the Web Cache administrative interface. Administrators can instrument entire applications, individual pages, or sets of pages.

Now administrators can quickly identify those URLs that are troublesome and those that are responsive. More often than not, administrators and developers are surprised to learn that performance is sub-par in many areas of the Web application, especially for users accessing the application from branch offices and distant geographic locations. *In turn, a more complete and accurate understanding of the end-user experience creates awareness among administrators and developers of the need for performance-enhancing technologies like OracleAS Web Cache, which can be deployed both centrally and remotely.*

FLEXIBLE DEPLOYMENT OPTIONS

OracleAS Web Cache is suitable for the full range of application deployment environments – from small departmental servers, to enterprise data center Grids, to distributed branch office hierarchies.

In terms of hardware, OracleAS Web Cache is designed to run on commodity, one- or two-CPU machines. Customers can deploy OracleAS Web Cache on the same node (or nodes) as their application Web server(s). Customers can also deploy OracleAS Web Cache on dedicated hardware. The latter topology is often preferable in order to avoid resource contention. OracleAS Web Cache performs well on inexpensive hardware, so a dedicated deployment need not be a costly on e in terms of hardware expenditure. For very high-volume Web sites and applications, and to avoid a single point of failure, two or more hosts running OracleAS Web Cache may be deployed behind a third-party network load balancing device. Customers wishing to avoid the use of an external load balancer can take

advantage of high availability features built into the host operating system (e.g., the virtual IP, load distribution, and failover functionality of the Network Load Balancer service for the Microsoft Windows platform.)

In a Grid environment, OracleAS Web Cache instances can be provisioned on an as-needed basis. And enterprise customers with users in remote office locations can move cached content closer to end users by deploying OracleAS Web Cache in a distributed hierarchy.

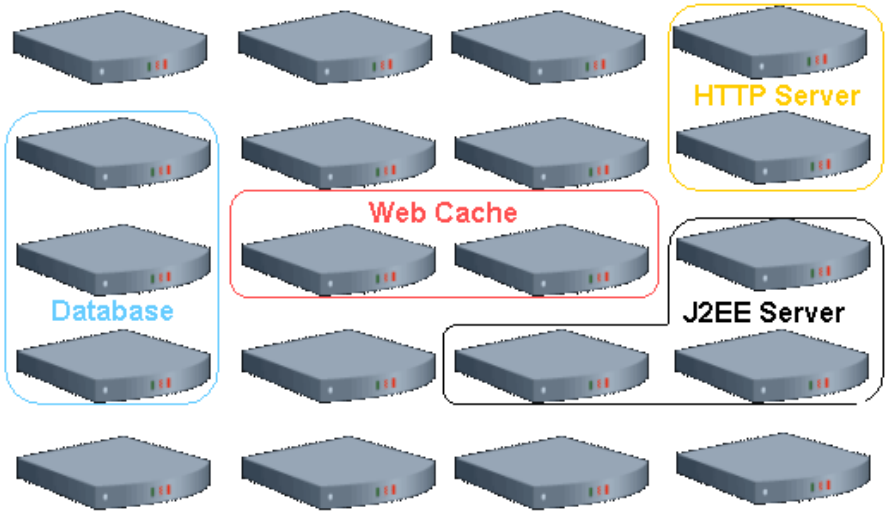


Figure 14: OracleAS Web Cache is a central component of Oracle Application Server 10*g* deployments.

"Benchmarking of Oracle Application Server proved the quality of the product in terms of features, performance, and stability. As it had been developed to accommodate a wide range of applications, Oracle Application Server was found to have outstanding flexibility, and its Web caching added tremendous performance advantages."

Jong-Ho Park, IS Team Senior Manager, CJ39 Shopping, Korea

Integrated with the Oracle Technology Stack

By designing Web caching functionality into the Oracle Application Server, Oracle makes it easier for customers to develop, generate and deliver content with an integrated single-vendor platform. To that end, OracleAS Web Cache provides out-of-the-box integration with:

- *other Oracle Application Server components*, such as OracleAS Portal, OracleAS Discoverer, OracleAS Forms, OracleAS Wireless, Oracle HTTP Server, OC4J, OracleAS Single Sign-On, and more
- *the Oracle Database*, including trigger-based invalidation
- *Oracle E-Business Suite applications*, for compressing application output
- *Oracle Enterprise Manager*, for management and end-user performance monitoring
- *Oracle JDeveloper*, including the JESI tag library and the ESI Servlet Filter extension

Heterogeneous Environments

Although integrated, *OracleAS Web Cache is by no means proprietary.* OracleAS Web Cache functions independently of the technology used by an application to generate HTTP responses.[7] Because OracleAS Web Cache fully supports HTTP/1.0 and HTTP/1.1, it can be deployed with any HTTP-compliant application Web server, including those available from BEA, IBM, ATG, Sun, Microsoft, Apache and others.

In addition to HTTP, OracleAS Web Cache leverages non-proprietary languages like XML and ESI for invalidation and page assembly. As such, OracleAS Web Cache works seamlessly with third-party databases, content management systems, content delivery network services, and load balancing devices.

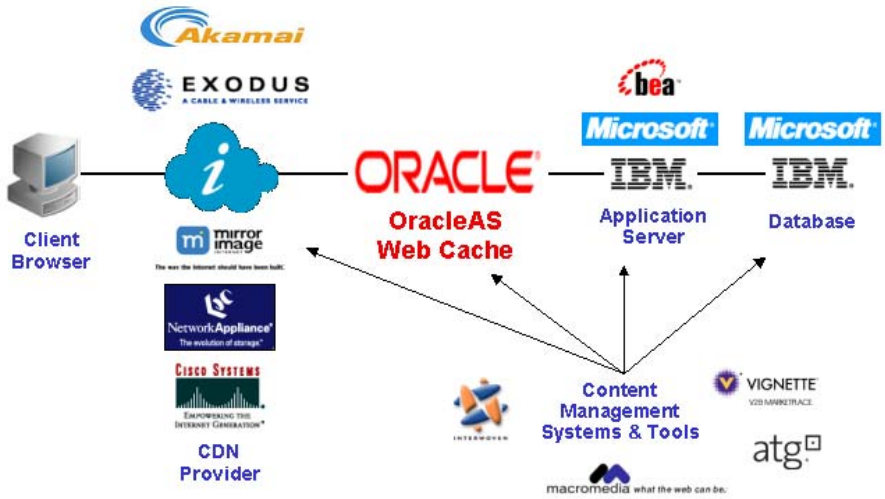


Figure 15: OracleAS Web Cache supports multi-vendor interoperability via HTTP, XML, and ESI.

Branch Office Hierarchies

OracleAS Web Cache offers hierarchical caching features that enable customers to easily create Content Delivery Networks (CDNs). For high availability and performance, many Internet businesses mirror their Web sites in strategic geographical locations. Caching is an excellent low-cost alternative to full-scale mirroring. Caching may also be used to serve local markets in order to shorten response times to these markets, and to reduce bandwidth and rack space costs for the content provider.

Within the corporate intranet, so-called "Enterprise" CDNs (eCDNs) provide shorter response times for branch office users of e-business applications. Compared to application mirroring and database replication, eCDN is a more manageable and cost-effective model of distributed computing. Using OracleAS Web Cache, customers can distribute the content assembly and delivery functions

[7] Pages that can be cached include those generated by JSP, Servlet, CGI, PHP, PL/SQL, ASP, or any other technology capable of outputting content for transmission over HTTP.

of their applications to key network access points, while maintaining centralized management of application logic and data.

When configuring an eCDN, customers typically deploy OracleAS Web Cache in each branch office data center as well as in the central office where the application and database are maintained. For example, a U.S.-based company might deploy instances of OracleAS Web Cache in its U.S., Japanese and Australian offices. The central cache residing in the U.S. serves as the content source for the caches in Japan and Australia. Using various commercially available DNS routing techniques, requests are transparently handled by the cache that is closest to the end user. A browser request made by a Japanese employee, for instance, is handled by the cache instance in Japan, *thereby reducing WAN traffic and eliminating long-haul network latencies.*

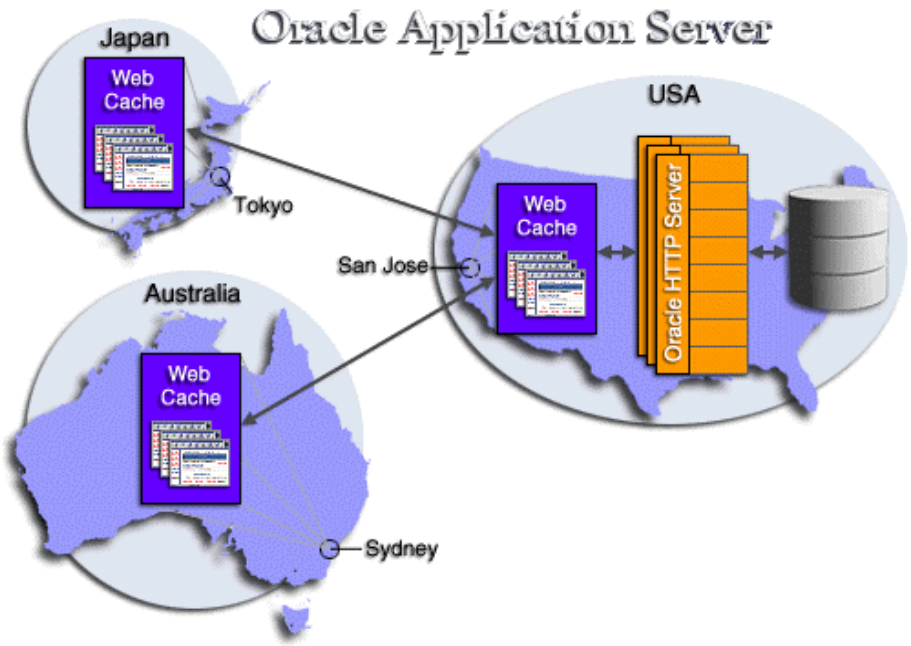


Figure 16: OracleAS Web Cache supports both intranet and Internet CDN deployments, enabling enterprises to distribute cached content to the edge of the network and closer to end users.

In a distributed cache hierarchy, the central cache is aware of the branch office caches. As a result, any content invalidation messages sent to the central cache automatically propagate to the remote caches. This invalidation propagation feature ensures content consistency across the CDN and simplifies the management of cache hierarchies.

CUSTOMERS AND RETURN ON INVESTMENT

With its powerful feature set, OracleAS Web Cache is the clear choice for accelerating e-business applications and Web sites of all kinds. Applications and services that stand to benefit from OracleAS Web Cache include, but are not limited to:

- Internet and enterprise portals, both traditional and wireless
- Web-based CRM and ERP applications, and the enterprise content delivery networks (eCDN) used to deploy these applications in branch offices
- Business-to-consumer and business-to-business e-commerce applications
- Business intelligence tools
- CDN and application hosting services

ROI: A Case Study of Digital River

"So far the Web Cache technology has helped us tremendously. In fact, as we hit peaks throughout the year we really couldn't have done it without the latest Web Caching technology that Oracle has provided us. We're seeing anywhere from two to 20 times performance improvements on some of our pages. On our very complicated pages that were dynamically generated in the past, we've taken some of those page times from 15 seconds down to a sub-second, so that's a huge benefit."

Marty Boos, CIO, Digital River

Digital River, a leading global e-commerce outsource provider founded in 1994, offers more than 32,000 companies the ability to cut costs and grow their businesses by using its complete e-commerce systems and services. The company's world-class infrastructure and e-marketing services are proven to grow businesses quickly and profitably while reducing risk. Digital River's commerce services include e-commerce strategy, site development and hosting, order and transaction management, system integration, product fulfillment and returns, e-marketing and customer service. Digital River's clients include Symantec, Motorola, Fujitsu, 3M, Siemens, Polaris, Novell, Autodesk, SONICblue, Adaptec and Staples.com. For more details about Digital River, visit the corporate Web site at www.digitalriver.com.

Digital River processes up to 40,000 orders and generates 1.5 million dynamic Web pages on a daily basis. Traffic volume doubles in the last quarter of the year with the holiday shopping rush. To sustain ever-mounting user loads and to support a growing client base, CIO Marty Boos needed a more scalable, high-performance application server architecture platform that is guaranteed to work seamlessly with the Oracle database. A previous OAS customer, Digital River also wanted to preserve its investment in Oracle PL/SQL, keeping open the option to migrate their applications to J2EE for a future date.[8] Boos needed to achieve these objectives and save money at the same time.

Due primarily to the dynamic generation of pages, Digital River's databases were forced to handle 300 concurrent connections at any given time. Boos considered throwing hardware at the problem but knew that this would be an expensive approach. With no time to redesign the application before the 2000 holiday season began, Boos and his staff evaluated a number of Web caching appliance solutions in an attempt to reduce the burden on their backend databases. After these static HTTP caching products proved ineffective in dampening the traffic burden, it looked like the expensive hardware approach was the only remaining alternative. Finally, in November 2000 Digital River heard about OracleAS Web Cache. Boos' team deployed the cache two weeks later and Digital River has been running the software successfully ever since.

[8] Digital River recently moved from a PL/SQL architecture to a J2EE architecture. Digital River had originally selected BEA WebLogic as the J2EE engine but in Fall 2002 migrated to OC4J. The reason? According to Digital River, "it cost two-thirds as much and is twice as fast."

Boos and his staff were able to reduce planned hardware and software infrastructure expenditures and improve performance by making the critical decision to migrate their middle-tier Web infrastructure to Oracle Application Server. Immediately, Boos saw dramatic performance improvements in his server farm configuration and Digital River's clients have enjoyed higher sales volumes thanks to faster download times.

Turning Cache into Cash

OracleAS Web Cache allowed Digital River to save millions of dollars on hardware. "It reduces the additional hardware I'll need to support additional e-commerce sites," says Boos. Digital River was able to eliminate 2 Sun Enterprise 6500s in their application server tier. Digital River had also anticipated purchasing 2 Sun Enterprise 10000s for their backend database tier because of the increasing traffic to their Web site. By offloading the number requests hitting the backend database, OracleAS Web Cache allowed Digital River to avoid this hardware purchase and save millions of dollars in the process. "Oracle Application Server has greatly improved the stability as well as the performance of my application server infrastructure," says Boos. "The Web Cache has moved much of the load off my back-end database tier and onto my application servers, where it's easier to manage and scale."

In total, the addition of OracleAS Web Cache translated into cost savings of over $3.3 million in the first year alone.

Today, the overall hit rate on the cache tier is approximately 86 percent, reducing the number of concurrent connections to the databases from 300 to 30. In addition to the 10-fold load reduction on the database tier, dynamic page delivery is as much as 20 times faster since deploying OracleAS Web Cache. Prior to using the cache, page response time for sites hosted by Digital River ranged from 3 and 15 seconds. With caching, delivery is now between 0.5 and 3 seconds. More importantly, since deploying OracleAS Web Cache, Digital River has been able to grow its business by a factor of four, from just over 8,000 customers to over 32,000, with minimal incremental investment in software and hardware.

SUMMARY

Oracle Application Server 10*g*, the next generation of Oracle's Integrated Software Infrastructure for Enterprise Applications, has been designed to enable Grid Computing.

Whether or not IT organizations choose to deploy applications on the Grid, they face three key performance-related challenges:

1. Dynamic, Web-based applications are compute-intensive, yet IT personnel are asked to do more with less
2. Unexpected traffic surges can cause delays and outages, yet planning for peak loads is cost-prohibitive

3. Users are demanding sub-second response times, yet visibility into end-user service levels is poor

OracleAS Web Cache, as part of Oracle Application Server 10*g*, is designed to address these challenges by providing:

1. More efficient use of low-cost hardware;
2. Sophisticated workload management; and
3. End-user performance monitoring.

OracleAS Web Cache is the industry's leading solution for dynamic content caching and page compression. The key challenges in dynamic content caching are the variation and volatility of the content. To overcome these challenges, OracleAS Web Cache features the most flexible event-driven cache invalidation mechanism on the market. And it is the first cache to support the open standard ESI specification for partial-page caching and personalized page assembly.

Additionally, OracleAS Web Cache offers sophisticated, yet easy-to-use workload management features, including load balancing, surge protection, performance assurance, and clustering. Together, these features ensure quality of service and system reliability under high load.

End-user service levels are no longer a mystery thanks to Oracle Application Server 10*g*. IT managers and application developers can utilize the end-user performance monitoring features of OracleAS Web Cache and Oracle Enterprise Manager to gain valuable insight into the collective and individual experience of end users.

OracleAS Web Cache is designed to run on inexpensive, commodity hardware and can be deployed in departmental, grid, and hierarchical topologies. OracleAS Web Cache is available on all the standard operating systems certified by Oracle Application Server. Nevertheless, heterogeneous environments involving third-party application servers, databases, and content management systems are also supported. *A standalone installation option is available for download from Oracle Technology Network*. In short, deployment options are flexible.

With OracleAS Web Cache, your business application's performance can improve by several orders of magnitude with zero or very little development effort. The return on investment is also significant, both in terms of developer resources (you no longer need to build your own dynamic caching solution) and infrastructure cost savings. As Grid Computing takes hold in IT organizations, a sophisticated server acceleration solution like OracleAS Web Cache becomes a necessity for deploying dynamic, high-performance Web-based applications at low cost.

MORE INFORMATION

For technical information on OracleAS Web Cache – including white papers, sample code, documentation, and trial downloads – please visit http://otn.oracle.com/products/ias/web_cache/

ORACLE FUSION MIDDLEWARE

Caching In on the Enterprise Grid:
Turbo-Charge Your Applications with OracleAS Web Cache
An Oracle Technical White Paper
Sep 2005

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com