ABSTRACT
Caching is a necessity for serving dynamic Web pages at high speed and low cost. 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, Oracle9i Application Server contains state-of-the-art Web caching technology – Oracle9iAS Web Cache – designed to increase throughput, shorten response times, and lower infrastructure costs for e-business application deployments. Specifically, the Web Cache uses industry-standard Edge Side Includes (ESI) tags to provide partial-page caching, personalization, and dynamic page assembly functionality. This paper provides a brief overview of Web Cache and ESI as well as a detailed case study on the use of these technologies at a large European media company.

INTRODUCTION TO ORACLE9IAS WEB CACHE AND ESI
Dynamic, personalized Web pages are computationally expensive to create. This makes them valuable, yet challenging to cache. Edge-Side Includes (ESI) is an open specification for a simple markup language that enables dynamic page fragmentation, caching, and assembly. It improves the performance and reduces the cost of dynamic content creation. Oracle9iAS Web Cache, an integrated part of Oracle9i Application Server (Oracle9iAS), is the world's leading dynamic content caching solution. As the first caching product to support the open ESI specification, Oracle9iAS Web Cache offers powerful ESI extensions and numerous advanced features designed specifically for caching dynamic content. Web Cache can be deployed locally as an application server accelerator for Oracle9iAS (or any other HTTP-compliant application server) to reduce content creation cost, or in a distributed enterprise content delivery network (eCDN) to reduce both content creation cost and data transmission delay to browsers.

In a typical deployment, Web Cache sits 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, 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 servers and database servers, also known as the origin servers, generate the response and send it back to the Web Cache. (The response may be generated using JSP, Servlet, CGI, PL/SQL, ASP, or any other technology capable of outputting content for transmission over HTTP.) Web Cache stores any cacheable objects returned by the origin server and forwards the response to the browser. Simple to moderately sophisticated caching policies may be configured by an administrator using “caching rules” and require zero application modifications. Developers may also opt to express caching policies in response headers, as well as leverage the ESI markup language for expressing these policies at the partial-page level. As the Web Cache becomes populated, it is able to serve more of the requested content itself. This, in turn, frees up processing resources on the origin servers.

Oracle9iAS 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 verify the consistency of Web pages with the data sources used to create them. When deployed as server accelerators, traditional static 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 leading dynamic Web sites and Web-based applications. In contrast, Oracle9iAS Web Cache was built from the ground up to cache volatile Web content. 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.

**CASE STUDY - NRK.no**

NRK (Norwegian Broadcasting Corporation) is the biggest radio and television broadcaster in Norway, with several radio and television stations. NRK is owned by the government and funded by the residents of Norway, who pay an annual usage fee.

NRK has a lot of valuable content that is suitable for publishing on the Web. NRK.no, the online arm of NRK, has mainly focused on publishing textual information but is also providing streaming of its radio channels and some of its television programs. The company has also successfully used micro payment for historical television programs. In addition to its main office, NRK has 17 district offices that provide local news, as well as several news correspondents abroad.

The Oracle Consulting engagement with NRK.no began three years ago with a project to reengineer the company’s homegrown publishing system. Oracle Consulting has occasionally supported NRK.no with this system since completion of the project. In the meantime, NRK.no has experienced significant and constant growth in visitor traffic, resulting in problems related to increased load on the Web site’s software and hardware infrastructure. To address the scalability and performance challenge, the company tried several caching products, none of which provided a performant, stable and cost-effective solution. In Fall 2001, NRK.no decided to introduce Oracle9iAS Web Cache to solve these problems.

**PROJECT MISSION**

The project mission was to “turn NRK.no into a scalable and reliable content delivery Website using Web Cache”. There were 3 phases defined:

- **Phase 1:** Whole-page caching with time-based invalidation
- **Phase 2:** Introduce a micro payment solution using Edge Side Includes (ESI) and serve cached content that are paid for.
- **Phase 3:** Partial-page caching with event-based invalidation that is integrated into the publishing system.

**TIME-BASED INVALIDATION (EXPIRATION)**

Caching with time-based invalidation means that the content is cached for a configured period of time after cache entry. When the expiration limit is reached, the cached content is marked invalid, and hence the content will be fetched from the origin server upon next request.

The goal of the first project phase was to replace a time-based system built on Squid Proxy Cache with a time-based system built on Oracle9iAS Web Cache. This was a very easy task to achieve. The Linux servers were reused, as well as the regular expressions used by Squid for caching rules. The rest of the system was not influenced with this change.
**SETUP TIPS FOR WEB CACHE RELEASE 9.0.2**

After the caching rules are defined, turn on verbose event logging on Web Cache. The event logs are very valuable to use when analyzing the defined caching rules. NRK.no made a script that went through the event log and pulled out a list of non-cached documents sorted by number of hits. This is shown below:

```bash
sed '/^10.* Reason: HTTP method or query/!d' ./event_log | 
sed 's/a^.* Insertion not done for //i' | 
sed 's/Reason: HTTP.*$//' | 
sort | 
uniq -c | 
sort -nr > notCachedUrls.log
```

UNIX script that pulls out a sorted list of uncached documents for the 10th day of the given month.

(Note that in Web Cache release 9.0.4, a script is no longer necessary to view popular non-cached documents. This level of analysis is now integrated into the Web Cache Manager administration console.)

**CHALLENGES WITH TIME-BASED INVALIDATION**

Key challenges with caching include the variation and volatility of the content. A page is often composed of static content (infinite expiration), short expiration content, and content that is dynamically generated. Examples include old articles and menus (static content) and newly created articles and front pages (short expiration). Personalized content, authorized content and poll results are examples of dynamic content that are difficult to cache.

When a page is composed of content elements with diverse behavior, its validity is governed by the element with least expiration time, i.e. if it contains dynamic or personalized information you cannot cache the page at all. Solving this dilemma requires the use of advanced caching techniques like partial-page caching and event-based invalidation, features only available in Oracle9i AS Web Cache.

**EVENT-BASED INVALIDATION**

Time-based invalidation is configured using administrator-defined rules in Web Cache that specify the lifetime of the cached objects. Time-based invalidation may also be specified in ESI response headers. Event-based invalidation relies on HTTP POST messages, where documents to be invalidated are specified in the body of the message. An invalidation message is sent from the origin server(s) to the Web Cache when the content changes. Event-based invalidation is intended for frequently changing content and/or content with update frequencies that are difficult to predict.

**INVALIDATION METHODS**

Invalidation is either performed manually from an OEM console or programmatically from within an application. Web Cache listens to a dedicated port for invalidation messages. The format is XML and is specified in a provided DTD. This means that invalidation messages can originate from every platform/programming language that supports XML over HTTP.

An invalidation message can specify URLs and the “severity” of invalidation. The URLs can be specified using Regular Expressions (regexp) or an exact URL (ADVANCED vs. BASIC invalidation). NRK.no is using ADVANCED invalidation, where a URL prefix is defined together with the regular expression. The “severity” specifies whether Web Cache should remove content immediately or allow a grace period during which content can be served stale. (Stale content is only served if the origin server is unavailable or too busy to refresh it.)

NRK.no has a publishing system based mainly on PL/SQL and Java in the database. Oracle Consulting created a PL/SQL package to handle the invalidation communication with Web Cache. In Oracle9iAS Web Cache 9.0.x, Java...
and PL/SQL APIs are provided for this purpose.

**PARTIAL-PAGE CACHING**

Web Cache provides dynamic assembly of Web pages with both cacheable and non-cacheable page fragments. It provides for personalized page assembly by enabling Web pages to be broken down into fragments of differing caching profiles. Each fragment is maintained as a separate element in the cache and can be fetched independently from origin servers.¹ Web Cache assembles the fragments into HTML pages as appropriate when requested by end users.

With partial-page caching, far more HTML content can be efficiently cached and delivered by Web Cache than with traditional caching solutions. By enabling dynamic assembly of Web pages on the Web Cache rather than on the origin servers, developers can choose to cache fragments that are shared across several pages, thereby reducing page creation and assembly costs when different users access the application. Furthermore, page assembly can be conditional, based on information provided in HTTP request headers or end-user cookies.

Developers provide partial-page caching instructions to Web Cache by using the standards-based Edge Side Includes (ESI) markup language to define fragments of a page. NRK.no first introduced partial-page caching for a micro payment solution, then for their publishing system.

**EDGE-SIDE INCLUDES (ESI)**

Co-invented by Oracle and Akamai in April 2001, Edge-Side Includes (ESI) is an open standard specification for a simple markup language aimed at enabling greater levels of dynamic content caching. The most important function of ESI is page fragmentation and assembly. ESI include and inline markup tags divide Web pages into multiple independent fragments. The original page consists of a page template and several page fragments. Templates and fragments can all have their own independent caching properties. Some may change every minute or hour; others may change every month. Certain fragments and templates may be broadly shared, while others may be unique. Although some Web pages are not cacheable with or without ESI, in many dynamic Web sites an ESI-capable cache can store templates and fragments as separate objects and assemble customized dynamic responses directly out of the cache without incurring any cost at the mid-tier application level or database level.

ESI and its Java derivative, ESI for Java (JESI), are designed to enable high performance Web applications. While there are probably more than enough technologies to do page assembly in databases, application servers or client browsers, ESI page assembly was introduced to expose inherent caching values in dynamic content so that expensive and complicated business applications can also enjoy the benefit of high-performance caching. Every Web-based business application determines its own internal caching policies. Unfortunately, as soon as the response leaves that application, it is very hard for a generic cache to recognize the caching values. With proper ESI tags, an ESI-enabled cache like Oracle9iAS Web Cache can understand the caching properties of each Web page as much as the application itself can. Indeed, thanks to the introduction of ESI, application developers no longer need to build their own page fragment caching mechanisms and can instead rely on Web Cache for page assembly.

Detailed information about ESI and JESI can be obtained by visiting [www.esi.org](http://www.esi.org) and by referring to the *Oracle9iAS Web Cache Administration and Deployment Guide* available on [OTN](http://otn.oracle.com).

**MICRO PAYMENT SOLUTION BASED ON ESI**

NRK.no introduced a micro payment solution for one of its most popular youth programs that includes Web content and video. The challenge for NRK.no was that they could not bypass the cache mechanism and pull content directly from the database since this already had enough load.

¹ Note that the `<esi:inline>` model enables partial-page caching even when fragments cannot be fetched independently from the origin server.
Besides the requirement of zero performance degradation, the solution had to be based on the exiting publishing system. That is, Oracle Consulting could not build a custom system for the payable content.

With ESI, the solution was quite simple. Oracle Consulting used ESI tags to check whether a given cookie is set (in the example below, this is hard coded for sake of clarity). If the cookie is present, the content is shown; otherwise a login screen is displayed where the content normally appears. The login screen connects to a servlet that performs the authorization via a third-party vendor (Payex, the most used micro payment system in Norway) and sets the cookie if authorized.

```html
<html>
...
<esi:choose>
    <esi:when test="\{HTTP_COOKIE\{userAuthorized\}\}=='True'">
        ...
        <!-- original section/article appears here -->
        ...
    </esi:when>
    <esi:otherwise>
        <!-- If cookie not set show login -->
        <esi:include src="/j2ee/nrk/betalingsskjerm.html" onerror="continue"/>
        <!-- The login-servlet will set the cookie if authorized-->
    </esi:otherwise>
</esi:choose>
...
</html>

Example of ESI tags used for viewing content based on authorization.
Login screen when the user is not authorized.

Everything but the dynamic content is shown when the user is not logged in. When using this method, the solution is very integrated since the login is within the rest of the page. It is also possible to use

\[
<esi:when test="\$\{(HTTP_COOKIE\{userAuthorized\})=='True'\}"
\]

to exclude other information for a user that is logged in.
Rudebilstasjonen rives

Til tross for protestet setter "kulturbyen" Arendal utfordringer i gang med å jevne sitt mest kjente landsmerke med jorden. I følge profittegernes voldsligse minutt for minutt. (OBS: For kun kr 10,- får du tilgang til videoserien, der du blir finne hele min Drammenserie.)

Hovedartikkelen

Epokegjørende DVD-utgivelser


Samlingens tilsette

Alt om "Det er boka mi"

Midsommerfigurerne forvandler med sterke lokaliserede. Har satt spor etter seg. Her finner du ordene sammen, i tillegg til noen "utskåret".

The same URL when the user is authorized.
PARTIAL-PAGE CACHING AND EVENT-BASED INVALIDATION AT NRK.NO

NRK.no has built their own publishing system, which is used by approximately 1000 content editors. When introduced to Web Cache as a solution for scaling their content delivery platform, NRK.no set forth some specific requirements. First, every change made with the publishing system should be captured and the cache should be properly notified of the change (i.e. if an editor changes a picture in the multimedia gallery, every component that is using the picture has to be updated, including front pages, articles, links, etc). The other requirement was to have a notification system that is loosely coupled to the publishing system.

The publishing system is based upon use of components (very similar to portlets used in Oracle9iAS Portal) that the administrator creates when a page is composed. Every cacheable component, including the template of the page, is stored as a fragment in Web Cache. The template references the fragments by using <esi:include> tags.

Dependencies can be registered for every defined component in the system. There are several pre-built dependency classes and a couple of custom-built ones that could not be generalized. These are implemented as component notificators in Java. Examples of these include PageNotificator, SingleArticleNotificator, and ObjectNotificator. The events are fired from triggers in the database, which check whether there are dependencies registered on these tables/columns. If so, an event is sent to the DependencyDispatcher, which applies the appropriate notificator to the event according to the dependencies. The notificator generates the necessary XML-formatted invalidation message(s) and sends this to Web Cache.

Design of Notification System for the Publishing System
INVALIDATION OF FRAGMENTS

Invalidation is an atomic operation. This means that if you specify a regular expression that invalidates several documents, they are simultaneously marked invalid. This is particularly important when invalidating fragments in a partial-page caching scenario. If there are several article fragments on the front page of a news portal, and the order and position of those articles needs to change for some reason, it is important that every fragment is invalidated simultaneously so the end user does not see the same article fragment occur twice. A better solution would be to invalidate the “template” that includes those article fragments. When building a system for partial-page caching it is very important to carefully design the template, fragments and their corresponding URL structures so that the invalidation mechanism is efficient and effective. While the invalidation methods are atomic operations, the ESI page assembly in Web Cache is not (due to performance reasons).

CONTENT DELIVERY ARCHITECTURE

NRK.no serves its traffic easily through a single 1-CPU Linux box where Web Cache is installed. The machine has little or no load in ordinary production settings. Oracle Consulting has been running tests (for a longer period) with about 600 requests per second without problems. A peak of about 1100 requests per second for one Linux machine was obtained, however the network and/or the client emulator were the bottlenecks, not Web Cache.

In the current production architecture there are two standalone Web Cache machines with manual failover mechanisms. NRK.no will cluster these machines using Web Cache’s built-in clustering feature as soon as another test machine is set up.

```
Webcache1.nrk.no
Webcache2.nrk.no
Linux Suse, 1 CPU
```

```
gordon.nrk.no
IAS
mod_plsql, mod_rewrite
```

```
mor.nrk.no
2 CPU
DB 8.1.7.3
```

Architecture with Oracle9iAS Web Cache

In addition to improved system availability and capacity, cache clustering has a number of advantages. The problem with adding standalone Web Cache machines is that every new machine almost doubles the load on the application servers (due to cache misses). This is not the case when the machines are clustered into one logical cache, where cache
instances request much of the content from their peers instead of from the origin server, and where there is one place to invalidate and update the cache using event-based invalidation. For more information on cache clustering, readers should refer to the Oracle9iAS Web Cache Administration and Deployment Guide and other content available on the Web Cache page on OTN.

Web Cache caches HTML content delivered from the publishing system. Images, which reside on a dedicated image server, are also cached. Virtual hosting functionality in Web Cache is used to achieve this. Serving content from several hosts is useful when creating an integrated solution across several applications that will be included into the same Web pages. Retrieving ESI fragments from several hosts and aggregating them in Web Cache at the front-end is a solution that appeals to NRK.no architects.

NRK.no also sees potential in additional ESI use in order to extend their solution. For instance, the \texttt{<esi:environment>} tag is very useful for adding additional logic to the Web Cache tier.

\textit{Improved User Experience Due to Caching}

The user experience of a Web site is crucial to its success. Users do not want to wait for more than a few seconds before a page appears in their browser. Along with dynamic, functional and appealing Web pages, speed of delivery is essential for success on the World Wide Web. Web Cache clearly improves the performance of dynamically generated pages, as evidenced by the experience at NRK.no and elsewhere. Web Cache improves the user experience in the following ways:

- Web Cache automatically compresses HTML content to reduce outbound traffic size (reduces download time for users).
- Web Cache serves its pages from memory; hence page delivery is extremely fast (even compared to other cache systems that are usually disk-based).
- Web Cache reduces the time to update content via event-based invalidation. This is more flexible than time-based invalidation and provides the end-users with consistently fresh information with no delays.
- The possibility to have cached content with mixed behavior (static and dynamic) on the same page also gives a better user impression of the site.

\textit{HTTP Error Caching}

NRK.no needs to simplify/beautify their URLs due to campaigns on TV / Radio. To achieve this they are using \texttt{mod_rewrite} (a module to Apache). \texttt{mod_rewrite} returns a HTTP 302 code that redirects requests to the correct page. With Web Cache, NRK.no is also caching these responses, which constitute about 1\% of the total traffic to the site.
THROTTLE TRAFFIC TO THE BACK-END

For NRK.no, one very important feature of Web Cache is the ability to continue to serve documents even when the origin servers are busy at the moment, or cannot deliver anything at all (i.e. when origin servers have been taken down for maintenance). This functionality ensures both high availability and performance of the NRK site.

- Set maximum number of connections to App. Server
- Serves stale documents when necessary

Web Cache
App. Server
Database

Throttle back-end systems by using Web Cache

In order to configure this throttling mechanism, administrators set a maximum number of connections to the application server(s). When this limit is exceeded or Web Cache cannot reach the application servers, Web Cache starts to serve stale documents. A stale document is a document that is marked invalid but not removed from the cache. Of course, serving stale content is something one prefers to avoid, but it is usually better to serve something that is not 100% fresh rather than not serving anything at all. With Web Cache, you can select to remove invalid documents immediately, or base the removal on the current load of the origin servers.

PERFORMANCE ASSURANCE HEURISTICS

Web Cache provides minimal trade-off between performance and consistency through patent-pending performance assurance heuristics that determine which documents can be served stale. These heuristics are based on a number of factors including:

- Validity of documents (based on the expiration time, invalidation time, and removal time of an object)
- Popularity of documents
- Availability of origin server(s)

This ensures that the most popular documents are refreshed first in a case where stale documents have to be served.
WEB CACHE RESULTS AT NRK.NO

NRK.no is a very satisfied user of Web Cache. It has given them a stable, scalable and performant platform for the delivery of dynamic content, which was their main purpose. They now have full control of the load on their back-end servers, regardless of the amount of incoming traffic to the site. This has also given them much more flexibility on the hardware side of the equation. Without Web Cache, NRK.no would need to scale its application server and database hardware infrastructure to handle much greater loads (since two relatively idle Web Cache machines handle almost 99% of the traffic today). With partial-page caching and event-based invalidation they also have a solution where fresh, yet cached, content is available immediately to editors and end-users. For their part, end-users enjoy better responsiveness from the site and experience fewer site outages. Finally, dynamic content caching yields great cost savings compared to static caching or no caching at all. (Readers should contact an Oracle sales representative or Oracle Consulting for estimated return on investment for this and other deployments of Oracle9iAS Web Cache.)

SUMMARY

Dynamic content caching offers key performance benefits in today’s Web-based business applications. Static content caching, while simple and efficient, does not solve the performance problem in this area. The key challenges in dynamic content caching are the variation and volatility of the content. ESI is a superior caching technology aimed at increasing cache-hit ratios and reducing storage requirements in dynamic content caches. It is co-authored by several major ISVs and service providers including Oracle, Akamai, ATG, BEA Systems, Digital Island, IBM, Interwoven, Macromedia and Vignette. Oracle9iAS Web Cache, as part of Oracle9i Application Server release 2, is the industry’s leading solution for dynamic content caching. It is the first cache that supports the open standard ESI specification. It also offers powerful ESI extensions, advanced caching rules, automatic compression and a flexible event-driven cache invalidation mechanism. Using Oracle9iAS Web Cache and its ESI features, your business application’s performance can improve by several orders of magnitude with 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 hardware cost savings.

CONTRIBUTORS

The authors would like to thank Per Anders Jørgensen, Regine Vergon, and Jørn Lager Hansen from the Oracle Consulting team in Norway, as well as Xiang Liu, Ming Lei, Ric Goell and Rajiv Mishra from the Oracle9iAS Web Cache development team in California.