How WebSphere Caches Dynamic Content for High-Volume Web Sites

Authors: High-Volume Web Site Team
Web address: ibm.com/websphere/developer/zones/hvws
Management contact: Larry Hsiung larryh@us.ibm.com
Technical contact: Catherine C. Diep cdiep@us.ibm.com
Date: December 15, 2002
Status: Version 1.0

Abstract: Optimizing for scalability and personalization remains a significant challenge for e-businesses as they balance the demands for instantaneous changes, high performance, availability, reliability, and security. Vendors are responding with infrastructure options and supporting hardware and software platforms that address these requirements. This white paper introduces a new feature of WebSphere Application Server Version 5 that can improve Web Site performance.

Executive summary

Web site visitors expect increasingly quick response time. Web site designers are always trying to improve response time and, at the same time, provide content that personalizes each visit and attracts visitors to become regular customers. Sooner or later customer expectations and designer efforts to meet them come into conflict as a result of severe network and server bottlenecks caused by the constant need to retrieve data from the back-end Web application and database servers. This is especially true for high-volume Web sites.

Server-side caching techniques have long been used to improve Internet performance of Web applications. In general, caching improves response time and reduces system load. Until recently, caching has been limited to *static content*, which is content that rarely changes. Opportunities to improve performance are greatest when *dynamic content* is also cached. Dynamic content is content that changes frequently, or data that is personalized. And, caching dynamic content requires proactive and effective invalidation mechanisms, such as event-based invalidation, to ensure the freshness of content. Implementing a cost effective caching technology for dynamic content is essential for the scalability of today's dynamic, data-intensive, e-business infrastructures.

This paper introduces the dynamic cache service that comes with WebSphere® Application Server, Version 5.0. WebSphere Application Server is the first server to include a dynamic cache service that is not only comprehensive and easily implemented, but also compatible with existing caching mechanisms. Using WebSphere Application Server dynamic cache service to cache dynamic content can reduce server-side bottlenecks and maximize system resources, thus boosting performance and reducing infrastructure cost. Given customer expectations and the need to retain them and attract more, dynamic content caching with WebSphere Application Server can be a competitive advantage for high-volume Web sites.

Contents

Introduction

Server-side caching techniques have long been used to improve Internet performance of Web applications. In general, caching improves response time and reduces system load. Most techniques cache static content, which is data that changes rarely, if at all, such as graphic and many text files. While the solutions for caching static content have resulted in excellent performance for some Web sites, they have little or no value in enhancing the performance of Web sites with dynamically generated pages. Dynamic content is the data that changes over time (such as stock prices, sport scores, and weather forecasts) or is personalized (such as shopping carts). Dynamic content can also be cached.

For years, IBM Research has developed and refined technologies that enable the caching of dynamic content. These technologies were implemented, deployed, and verified at various high-volume sport event sites such as the Olympics [1,2]. The success of the sport sites [3] demonstrated the feasibility and significance of caching dynamic content [4,5], and confirmed the scalability and reliability of the caching technologies. Based on these proven and scalable caching technologies, IBM developed a dynamic content caching solution for Java™ 2 Enterprise Edition (J2EE) applications running on WebSphere Application Server Version 5.0.

WebSphere Application Server offers a built-in dynamic cache service for serving dynamic content and caching data. There is no time-consuming installation and integration work needed to activate the dynamic cache service. The cache is enabled /disabled declaratively using simple XML configuration files or using the WebSphere Application Server's Administrative User Interface; these methods not only allow caching to be brought up quickly and easily, but also provide great flexibility and control at runtime. Also, you can define your existing caches, such as the caching component of WebSphere Edge Server or IBM HTTP Server, as *external caches* and use them in conjunction with WebSphere's dynamic cache service.

This paper introduces the presentation level caching features of the WebSphere Application Server dynamic cache service. These features are:

- *Servlet/JSP Result Cache,* which is nonintrusive, effortless, and ready to cache any existing whole page or fragment generated by a servlet or Java Servlet Page™ (JSP).
- *Command Cache,* which is used to cache dynamic data that is "expensive to re-create." This data requires either back-end requests, such as back-end Java database connectivity (JDBC) calls, or additional CPU intensive computation/manipulation at a later time.
- *Replication Support,* which enables cache sharing and replication among multiple servers and tiers (Figure 3).
- *Invalidation Support,* which includes rule-based, time-based, group-based, and programmatic cache invalidation techniques to ensure the freshness, consistency, and accuracy of the content.
- *Edge of Network Caching Support,* which extends the WebSphere Application Server caches into network-based caches, through the implementation of external cache control and distributed fragment caching and assembly support.
- *Tools*, which assist in configuring the cache and monitoring runtime

More information on additional features of WebSphere Application Server, such as EJB caching and Web Service caching, can be found at the WebSphere Application Server V5 InfoCenter [7].

Caching dynamic content

The key issue with caching dynamic content is to determine what should be cached, where caching should take place, and how to invalidate cached data.

What should be cached?

A candidate for dynamic content caching is content or data that is changing and at the same time must be stable over a long enough time for meaningful reuse to occur. If frequent access is high, such as pricing information of a popular stock, then even a short time of stability may be enough to benefit by caching dynamic content.

All dynamic Web pages consist of smaller and simpler page *fragments*. Some fragments are static (such as headers, footer), while others are dynamic (such as fragments containing stock quotes or sport scores). Breaking a page into fragments or components makes effective caching possible for any page, even a highly dynamic page. The goal of creating fragments or components is to maximize fragment reusability and cache utilization.

For example, the personalized page shown in Figure 1 contains user-specific elements applicable to only one person. Not much benefit can be realized by caching this whole page.

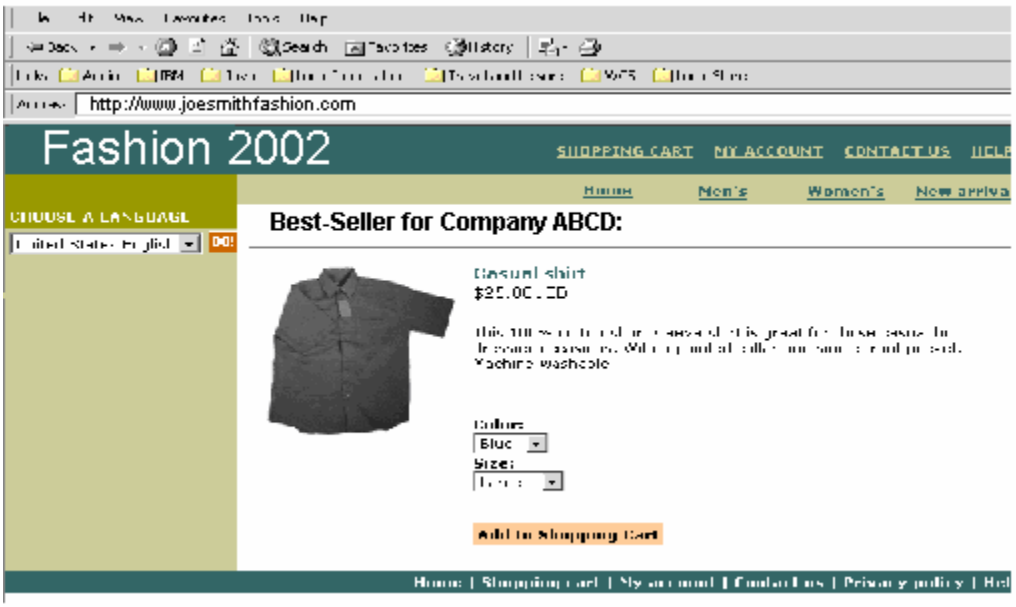

Figure 1. Example of a dynamic page containing personalized data

Figure 2 shows that when the same page is broken down into fragments based on reusability and cacheability, some or all of the fragments (for example, headers, footers, navigation bars for all users; targeted pricing and advertising for user groups) may become reusable and cacheable for a larger audience. Only fragments that are not cacheable need to be fetched from the back-end, thereby reducing server-side workload and improving performance.

Web pages should be fragmented to cache dynamic content effectively within the enterprise infrastructure and at the content distribution network. However, in some cases, even caching the most granular, final formatted fragment is not sufficient. Under such circumstances, caching at the raw data level is the next granular technique that can be used.

Caching a final formatted whole page (such as HTML or XML), a final formatted fragment, or a piece of unformatted raw data, each, in its own way, contributes to the ultimate benefit of caching dynamic content. The WebSphere solution for caching dynamic content offers features that enable dynamic content to be cached at various granularities, namely whole pages, fragments, and raw data. These important features are *Servlet/JSP Result Cache* and *Command Cache.*

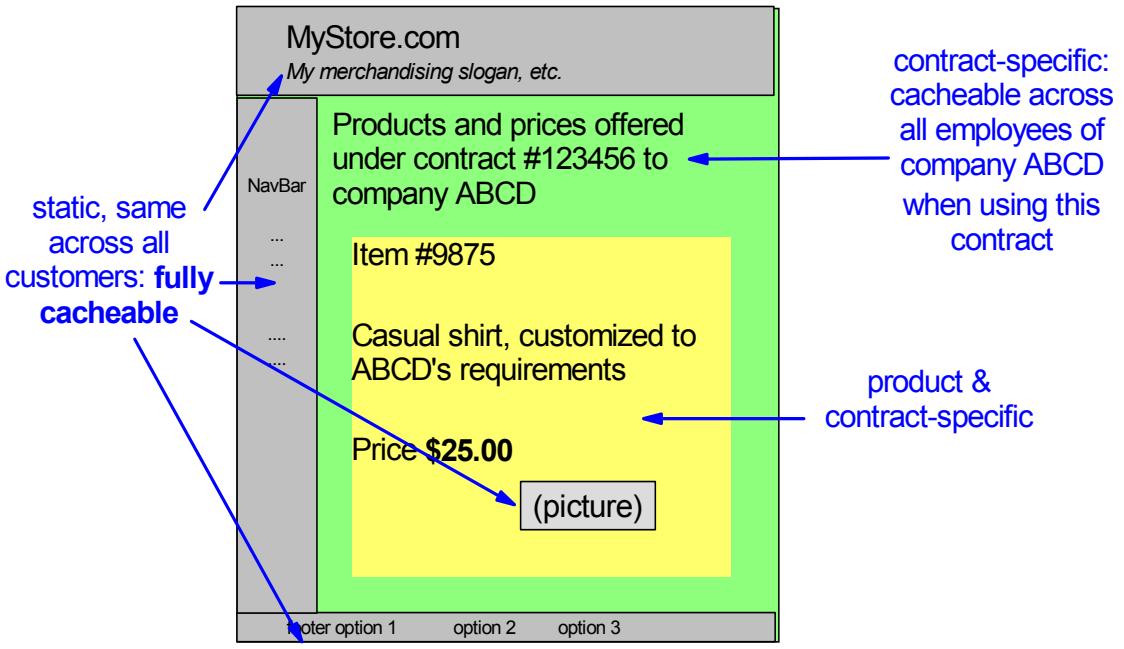

Figure 2. Example of the dynamic page fragmented for caching

Where should caching take place?

Theoretically, caching of dynamic content should take place as close to the user as possible. In reality, other factors such as security and user specific data may influence the choice for the best place to cache dynamic content.

Web page design also plays an important role in determining where dynamic data is cached. One example is personalized pages. Although personalized, these pages would contain user specific, nonuser-specific, locale sensitive, secure, nonsecurity sensitive dynamic data. To maximize the benefit of caching dynamic content, these types of data should be fragmented as finely as possible so they can be cached independently at different locations. For example, the nonuser-specific, nonsecurity sensitive fragments or components are generally useful to many users, and thus can be cached in a more public space and closer to users. The security sensitive data should be cached behind the enterprise firewall, yet as close to the edge of the enterprise as possible.

In a multi-tier e-business environment (Figure 3), WebSphere Application Server dynamic cache service can be activated at the business logic and/or presentation layer. It can also control external caches on servers, such as IBM WebSphere Edge Server and IBM HTTP Server. When external caching is enabled, the cache matches pages with their universal resource identifiers (URIs) and exports matching pages to the external cache. The contents can then be served from

the external cache instead of the application server, which saves resources and improves performance. Additionally, WebSphere Application Server dynamic cache service's *Replication and Invalidation Support* extends the cost effectiveness of caching dynamic content by enabling cache sharing and cache replication in an environment with multiple tiers and multiple servers. Finally, WebSphere Application Server's *Edge of Network Caching Support* expands the application caches into the network.

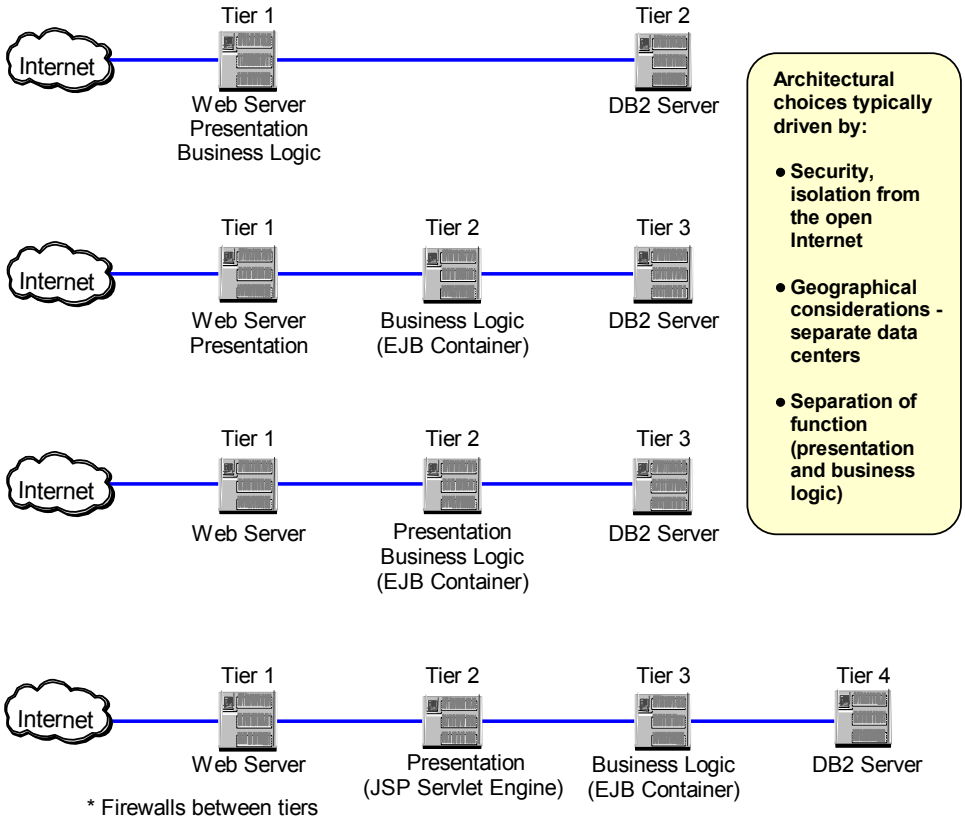

Figure 3. Examples of two-, three-, and four-tiered infrastructures

How is cache invalidated?

The biggest challenge when caching dynamic content is to guarantee the freshness, consistency, and accuracy of the content. This requires efficient and comprehensive mechanisms for identifying and updating pages/fragments/data that are no longer valid, a process called *invalidation*.

WebSphere Application Server dynamic cache service provides invalidation techniques that are rule-based, time-based, group-based, and programmatic. It can also invalidate the remote caches that were configured as its external caches. Additionally, the WebSphere Application Server dynamic cache service uses a built-in standard-based Java Message Service (JMS) messaging system that is lightweight and high performance to support the efficient and effective propagation of the invalidation requests among servers.

WebSphere Application Server dynamic cache service

WebSphere Application Server dynamic cache service is an in-memory cache system that has disk offload capability. Its caching behavior is described in the form of XML cache policy files. Cache policy is configured when building the XML configuration files, or with graphical user interface (GUI) tools such as the WebSphere Application Server's Administrative Console or the Application Assembly Tool.

Unique ID strings distinguish unique entries in WebSphere Application Server's dynamic content caching solution. These ID strings can be defined declaratively with the XML configuration file or programmatically by the user's Java program.

WebSphere Application Server dynamic cache service can control external caches. Different groups of external caches can be defined, each with its own set of member caches. The interface between WebSphere Application Server and external cache is the External Cache Adapter provided by WebSphere Application Server.

WebSphere Application Server dynamic cache service includes an alternative feature named disk overflow, which stores the overflow cache entries on disk for potential future access.

Servlet/JSP Result Cache

Servlet/JSP Result Cache intercepts calls to a servlet's service method, and checks whether the invocation can be served from a cache. If the servlet cannot be served from cache, the servlet is invoked to generate the output that will be cached. The resulting cache entry contains the output; the side effects of the invocation, for example, calls to other servlets or JSP files; and the meta data about the entry including timeout and entry priority information (see Figure 4).

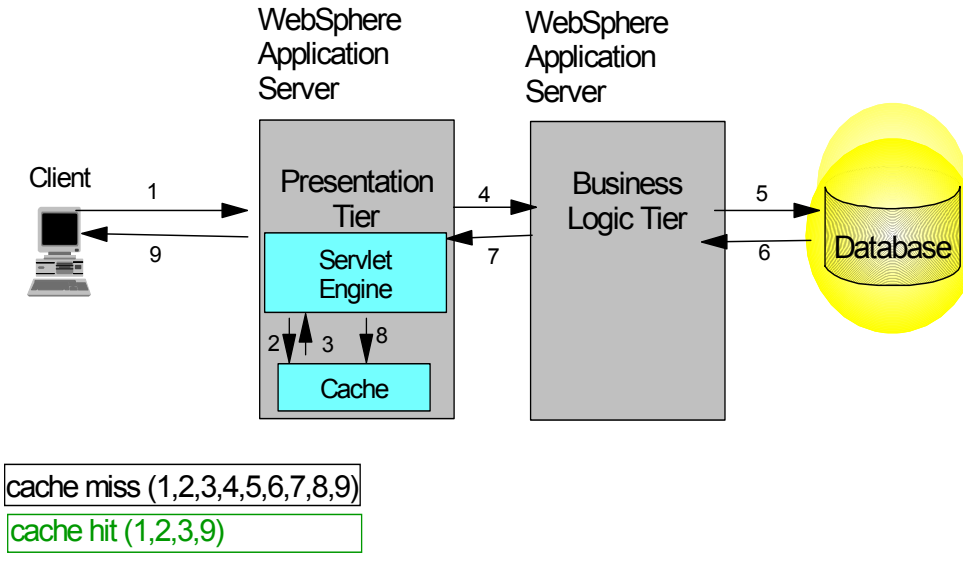

Figure 4. WebSphere Application Server Servlet/JSP Result Cache

Servlet/JSP Result Cache caching can be based on:

- Request parameters and attributes
- The URI used to invoke the servlet or JSP
- Session information

- Other options, including cookies

Figure 5 shows that the Servlet/JSP Result Cache can be used to cache whole pages or fragments at the WebSphere Application Server. A page, A.jsp, is made up of three fragments: B.jsp, C.jsp and D.jsp. All three fragments are independently cached at the application server.

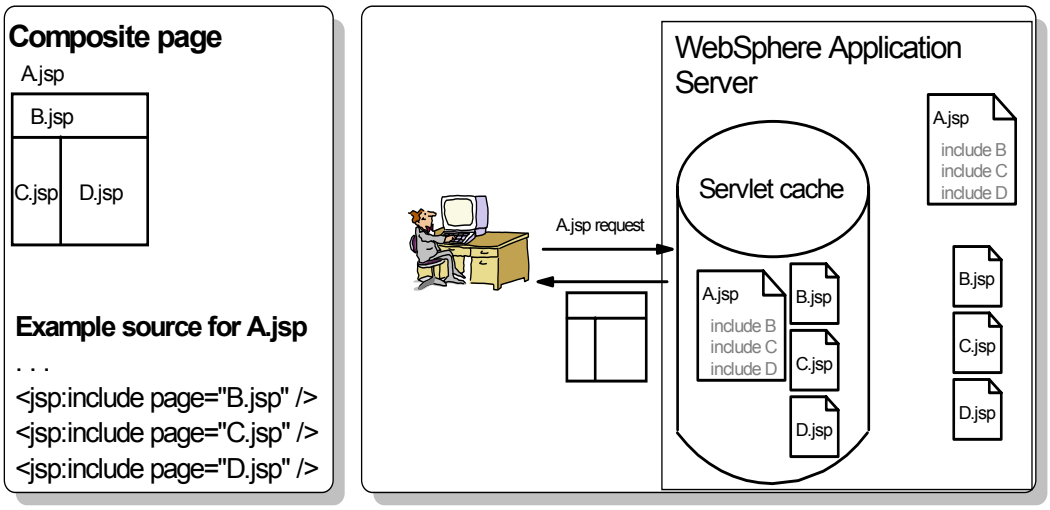

Figure 5. Caching fragments with the Servlet/JSP Result Cache at application server

The cache entries generated by Servlet/JSP Result Cache can be cached at the presentation layer. Alternatively, they can be pushed to external caches, such as the cache component of WebSphere Edge Server or IBM HTTP Server. See the section "Edge of Network Caching Support" for more information about external cache support.

For J2EE applications that have high read-write ratios, Servlet/JSP Result Cache creates an opportunity for significant gains in server response time, throughput, and scalability. Moreover, since Servlet/JSP Result Cache is nonintrusive and is enabled declaratively, currently deployed and running Servlets/JSPs can be configured to take advantage of dynamic content caching without changing any code.

Command Cache

Command Cache introduces the next level of granularity to dynamic content caching. Its primary goal is to serve the object's content from cache and thus minimize the execution of remote messages, such as back-end database JDBC calls, or calls to access data at remote nondatabase servers.

Generally, Command Cache is used to cache dynamic data that requires back-end data retrieval or additional computation or manipulation later. Command Cache forms a good synergy with Servlet/JSP Result Cache because, in some cases, even caching the most granular, final formatted fragment is not sufficient. For example, a personalized page may contain a stock list fragment (Figure 6). This fragment consists of two sets of information: the stock list that is highly personalized, and the corresponding stock symbol pricing information that is generalized information and usable by many users. Suppose the stock list is the customer's stock portfolio, which is highly sensitive and is stored at the back-end server. In this case, it is not effective to cache the final formatted fragment. Since the stock quotes are highly volatile, this fragment will

be regenerated repeatedly. The net is every time this fragment is reconstructed, it is necessary to retrieve the stock list from the source. A better approach is to use Command Cache to cache the stock list and avoid fetching the list continually from the back-end database.

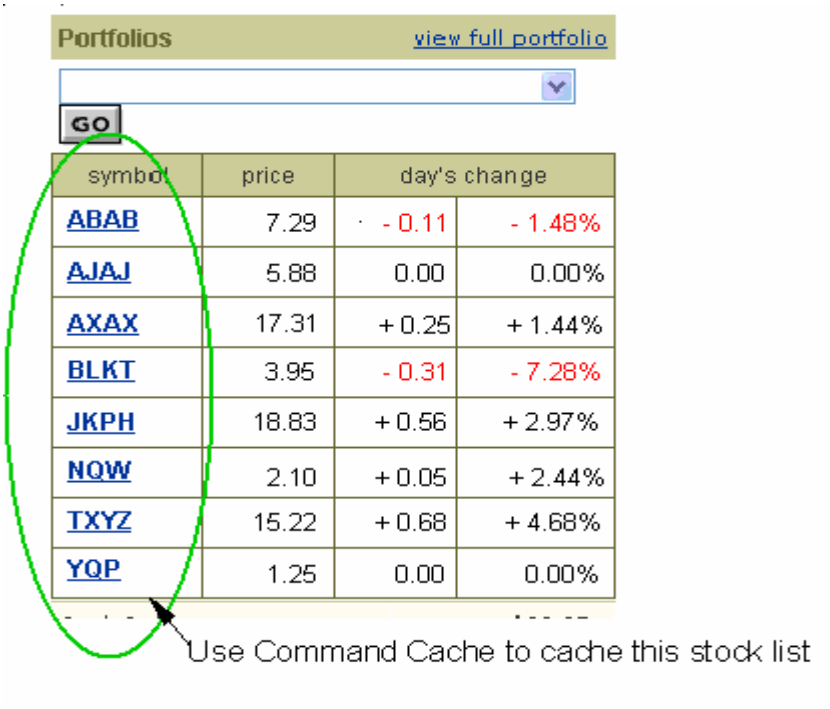

Figure 6. A stock list fragment

Command Cache is caching at the Java application level. To use Command Cache, user applications need to be written to the WebSphere Command Framework API. WebSphere Command Framework is based on the Command Design Pattern widely used in Java programming paradigms. Typically, these applications use "setters" to set the command's input states, one or more "execute" methods, and "getters" to retrieve the results. The results can be either formatted or raw data.

Command Cache intercepts the "execute" method call of a command written for Command Cache and checks whether the command object can be served from the cache. If the command object does not exist in the cache, the logic in the "execute" method is performed and the resulting object is cached.

The caching behavior of Command Cache is defined declaratively with the XML cache policy file, which describes whether and how a command should be cached. Command Cache can be used at the presentation and/or business logic layer in multi-tier environments.

Command Cache is easy-to-use. For existing applications that follow a Model, View, and Control (MVC) design pattern, Command Cache can be implemented with minimal impact to existing code.

Replication support

Replication Support further extends the value of caching in an e-business application. With this support, caches can be shared (central cache) or replicated (local cache) among servers. Replication can be enabled and configured declaratively with XML cache policy files. Cache policy also defines the cache entries or groups that should be shared or replicated.

Replication Support uses a built-in high performance standard-based JMS messaging system as its underlying engine for data replication.

Figure 7 shows an example of three servers configured to have their local caches replicated with each other. The messaging broker can be the built-in JMS broker belonging to one of the three servers. In this case, the broker is the built-in JMS broker at server one. Request for data (1) hits server one (or any one of the three servers). If the requested data is not in the cache, the data is fetched from the back-end server (2). Resulting data from the back-end server (3) is cached at server one and then returned to requester (4). When the cache of server one detects the cache update request, it publishes a message (a) regarding the change to the messaging broker. Whatever change occurred at server one is automatically replicated to server two (b) and server three (c).

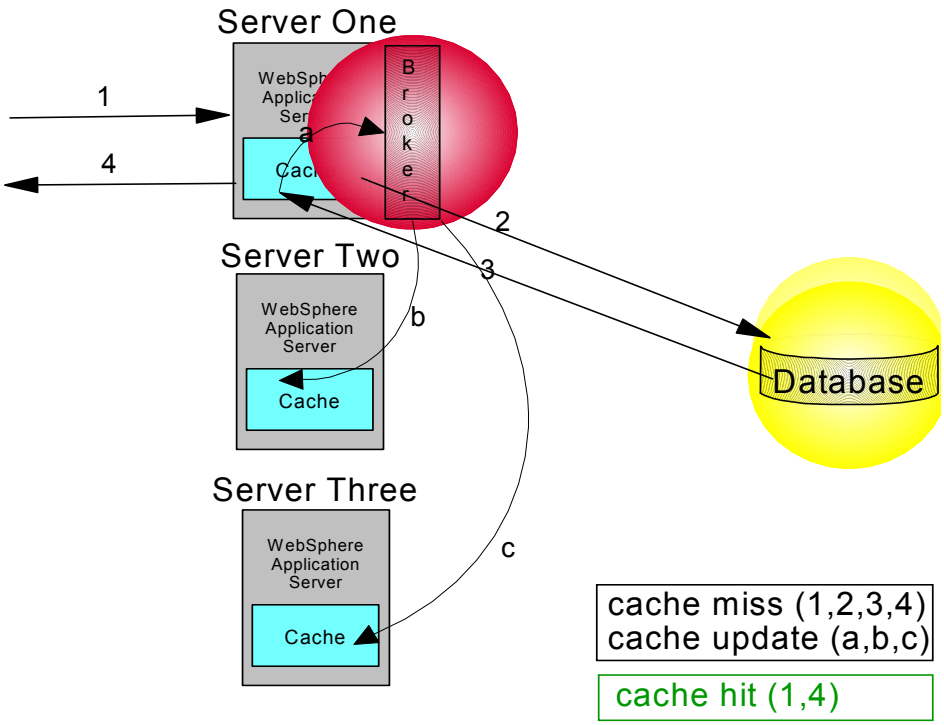

Figure 7. An example of cache replication

Invalidation support

The difference between caching static and dynamic content is the requirement for proactive and effective invalidation mechanisms, such as event-based invalidation, to ensure the freshness of content. The time-based invalidation alone is no longer adequate for dynamic cache invalidation.

WebSphere Application Server dynamic cache service provides rule-based, time-based, and group-based invalidation techniques. The WebSphere Application Server Enterprise Edition, Version 5 offers access to a programmatic cache and invalidation techniques. Invalidation policies can be defined with XML cache policy files. Invalidation policies allow triggering events to invalidate cache entries without the need to write explicit code. More complex invalidation scenarios may require code, which invokes the invalidation API.

The responsibility for synchronizing the dynamic cache of external caches and the WebSphere Application Server is shared by both systems. For example, a public Web page dynamically created and cached at the application server using Servlet/JSP Result Cache can be exported by Application Server and cached by the WebSphere Edge Server's Caching Proxy. Caching Proxy can then serve the application's execution results repeatedly to many different users until notified that the page is invalid. Content in the Caching Proxy's servlet response cache is valid until the proxy server removes an entry because the cache is congested, the default timeout set by the Caching Proxy's configuration file expires, or the Caching Proxy receives an Invalidate message directing it to purge the content from its cache. Invalidate messages originate at the WebSphere Application Server that owns the content and are propagated to each configured Caching Proxy.

Edge of Network Caching support

WebSphere Application Server dynamic cache service can control external caches.

External Cache

WebSphere Application Server dynamic cache service can use IBM HTTP Server's high-speed cache, referred to as the Fast Response Cache Accelerator (FRCA), as its external cache, to cache whole pages and fragments.

The Edge Components of WebSphere Application Server Network Deployment V5.0 can also be configured as WebSphere Application Server's external cache for whole page caching. In such case, the WebSphere Application Server dynamic cache service can be enabled to match pages with their universal resource identifiers (URIs) and export matching pages to the external cache (Figure 8). The contents can then be served from the external cache instead of the application server to significantly save resources and improve performance.

Figure 8 shows an example of exporting a dynamic cache page from the WebSphere Application Server to the cache component of WebSphere Edge Server. More information about caching dynamic content at the edge of the network can be found in a white paper titled *WebSphere Edge Services Architecture Guide to Edge Applications* of WebSphere Edge Server Library [11].

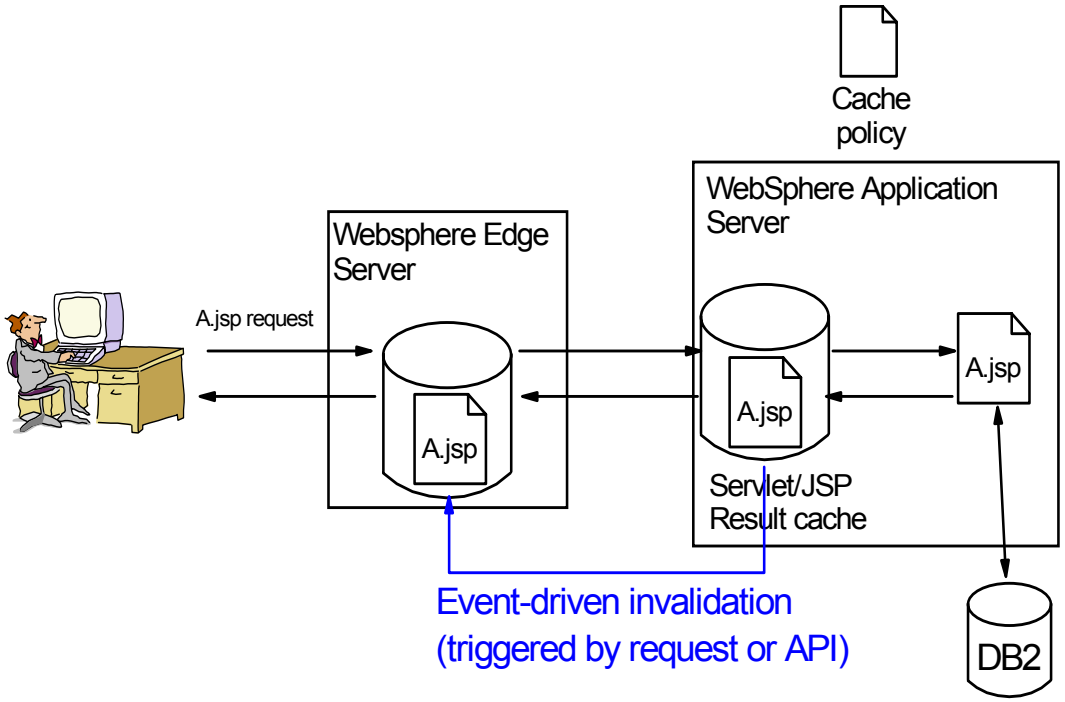

Figure 8. An example of exporting content to the external caches – WebSphere Edge Server

Distributed Fragment Caching and Assembly Support

WebSphere Application Server leverages the Edge Side Includes (ESI) [9] specification to enable caching and assembly of distributed fragments.

Edge Side Includes (ESI) [9] is a simple mark-up language used to define Web page fragments for dynamic assembly of a Web page at the edge of network. ESI is an open standard specification supported by Akamai [12] and leaders in the Application Server, Content Management Systems, and Content Delivery Networks markets.

With the Distributed Fragment Caching and Assembly Support, WebSphere Application Server customers can defer page assembly to any ESI compliant surrogate server, such as Akamai Akamai EdgeSuite service. This may result in significant performance advantage if fragments can be cached and reused at the surrogate.

WebSphere Application Server provides distributed fragment caching and assembly support through the WebSphere HTTP plug-in. WebSphere Application Server uses the WebSphere HTTP plug-in to communicate with the HTTP Server. This plug-in has the ability to cache whole pages or fragments. Additionally, it can dynamically assemble web pages containing ESI <esi:include> tags. Any server with WebSphere HTTP plug-in support can gain the benefits provided by the WebSphere Application Server dynamic cache service.

With WebSphere Application Server dynamic cache service's external cache control, distributed fragment caching and assembly support, dynamic content can be exported, cached, and assembled at the most optimal location, closer to the end user. More important, WebSphere Application Server can maintain control of the external cache through its Invalidation Support to ensure the freshness of cached content. As a result, WebSphere Application Server customers are equipped

to create and serve highly dynamic Web pages without jeopardizing page performance and user experiences.

In the table below, you can see which caching technique and cache distribution technique are most useful for each type of Web content.

Type of Web content	Caching technique	Cache distribution technique
Published / static (*.html, *.gif, …)	WebSphere Edge Server Caching Proxy	
Dynamic fragment/page (dynamically generated and reusable)	WebSphere Application Server Servlet/JSP Result Cache WebSphere Edge Server cache component and/or IBM HTTP Server as external cache WebSphere Application Server's HTTP plug-in distributed fragment caching and assembly support	WebSphere Application Server Servlet/JSP Result Cache configured with WebSphere Edge Server and/or IBM HTTP Server as external cache WebSphere Application Server's Cache Replication Support
Dynamic data (data object)	WebSphere Application Server Command Cache	WebSphere Application Server Cache Replication Support

Tools

WebSphere Application Server provides a variety of graphical user interface (GUI) tools to help configure and monitor the dynamic content cache.

Application Assembly Tool

Application Assembly Tool (AAT) is used to package application code components into the needed modules, which comply with the J2EE specification, for deployment onto the application server. WebSphere Application Server customers can also use AAT to configure the cache policy associated with the application for caching (Figure 9).

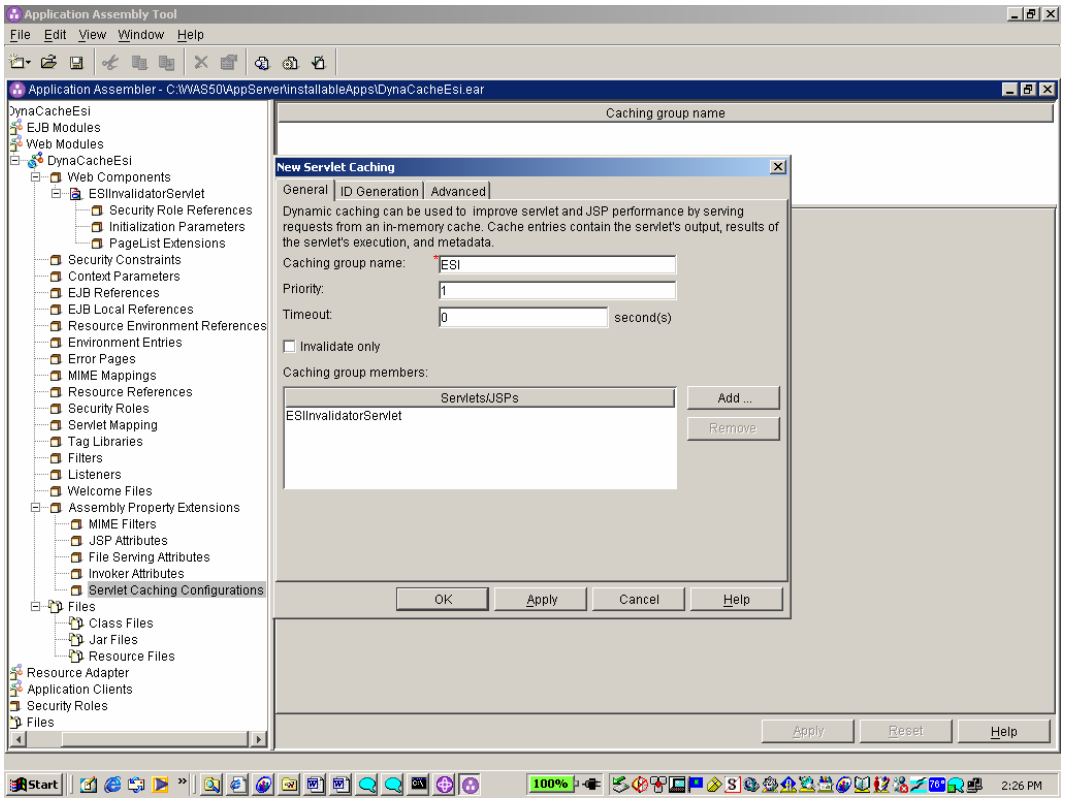

Figure 9. A screen shot of the Application Assembly Tool -- cache policy configuration

Cache Monitor Servlet

The Cache Monitor Servlet is provided as an installable Enterprise Archive file (EAR) that can be installed and used for monitoring cache statistics.

Conclusion

As more e-business sites seek to retain customers by serving personalized content, they face potential server-side bottlenecks, slow user response time, and increasing infrastructure costs. The WebSphere Application Server dynamic cache service can solve these critical challenges. Caching dynamic content that needs back-end requests or CPU-intensive computations can reduce server-side bottlenecks and maximize system resources, thus boosting performance and reducing infrastructure cost.

WebSphere Application Server dynamic cache service is easy-to-use and readily available. Customers can benefit from using the available comprehensive functions for caching their dynamic content. The *Servlet/JSP Result Cache* and *Command Cache* make possible the caching of dynamic content at various levels of granularity for the highest possible cache hits. The *Replication and Invalidation Support* facilitates caches to be shared, replicated, and synchronized in multi-tier or multiserver environments. The *Edge of Network Caching Support*, with its external caches and fragment support, generates a virtual extension of application server caches into the network.

Using WebSphere Application Server dynamic cache service can improve throughput and performance. As expected, improvements vary depending on how dynamic the data is, how well the pages are designed (fragmented) for maximum cache hits, and how costly it is to fetch and construct the content. Nevertheless, considerable benefits can be realized. Furthermore, the success of the IBM sport sites [1,2,3,4] confirmed the scalability and reliability of these underlying technologies in a truly high-volume environment.

The WebSphere Application Server dynamic cache service combined with an appropriate external cache, such as WebSphere Edge Server [10] or IBM HTTP Server, can power the high-volume sites with intensive dynamic content to achieve the highest level of scalability and performance.

References

[1] *High-Performance Web Site Design Techniques*, Arun Iyengar, Jim Challenger, Daniel Dias, and Paul Dantzig. IEEE Internet Computing, March/April 2000

[2] *A Publishing System for Efficiently Creating Dynamic Web Data*, Jim Challenger, Cameron Ferstat, Arun Iyengar, Paul Reed, Karen Witting IEEE IFOCOM, March 2000

[3] *Edge Services Architecture*, WebSphere Edge Server Library, January 2002

[4] *A Scalable System for Consistency Caching Dynamic Web Data*, Jim Challenger, Paul Dantzig, and Arun Iyengar IEEE INFOCOM, March 1999

[5] *Improving Web Server Performance by Caching Dynamic Data*, Jim Challenger, and Arun Iyengar. Proceedings on the Usenix Symposium on Internet Technologies and Systems, December 1997

[6] See all the IBM High-Volume Web Site white papers at www.ibm.com/websphere/developer/zones/hvws

[7] See the WebSphere Application Server Version 5 InfoCenter at www.ibm.com/software/webservers/appserv/infocenter.html

[8] Caching Technologies for Web Applications, C. Mohan, IBM at http://www.almaden.ibm.com/u/mohan/Caching_VLDB2001.pdf

[9] Learn about Edge Side Includes at www.esi.org

[10] Learn about the IBM WebSphere Edge Server at www.ibm.com/software/webservers/edgeserver/library.html

[11] *WebSphere Edge Services Architecture Guide to Edge Applications*, WebSphere Edge Server Library, May 2001

[12] Learn about Akamai Technologies, Inc. at www.akamai.com/

The WebSphere Application Server Performance Web site provides a centralized access to many helpful performance reports, tools and downloads. See www.ibm.com/software/webservers/appserv/performance.html

IBM Redbooks have additional WebSphere performance information at www.ibm.com/redbooks.nsf/portals/Websphere

Acknowledgements

The major contributors to this article are: Joe Anthony, John Connor, Jerry Cuomo, Catherine Diep, Edward Flickinger, Susan Holic, Larry Hsiung, Steve Ims, Brian Martin, C. Mohan, Gabe Montero, Steve Pogue, and Dan Shupp

Notices