Identifying and Caching Dynamic Web Applications: A Flexible Approach to Solving Performance Issues

Overview

Deploying dynamic web applications in the enterprise offers significant benefits, but users may well experience unpredictable application performance. One cause of poor performance is server overload, which results from greater application usage and content complexity.

The F5 WebAccelerator solves this key issue with its Dynamic Caching technology. Dynamic Caching is a unique capability that helps deliver highly dynamic web-enabled applications to users five to ten times faster while greatly reducing web infrastructure costs. This feature is particularly relevant for applications reaching scalability limits and needing to support more users, as well as for web applications that are slow because of legacy design and other problems.

Challenge

The key challenges revolve around increasing end user performance and achieving server offload for dynamic web applications and content. Generally, static caching can only be used for 30% of HTTP requests—a percentage that does not typically include high-value dynamic data (query responses, XML data, and so on).

Solution

Dynamic Caching

Dynamic Caching completely changes the caching model, making it possible to cache a much broader variety of content including highly dynamic web pages, query responses, and XML objects. This patented technology is completely unique to F5 and unavailable from any other vendor.

Dynamic Caching focuses on application logic and behavior, not just individual web objects. By understanding an application's high-level logic (what can and cannot be cached, what events cause invalidation, etc.), the WebAccelerator eliminates repeated processing of complex web requests. Dynamic Caching enables the WebAccelerator system to decide when to invalidate objects and how to identify reusable pieces of content. This is made possible by predefined application acceleration policies, an intuitive user interface, a powerful XML-based API (ESI), and an HTTP request-based triggering facility that together provide comprehensive controls for validating and invalidating content.

Without WebAccelerator and Dynamic Caching, the existing caching solution has only the object expiration date as a guide. Dynamic Caching enables the cache to look at anything in an HTTP request—from URLs to cookies, query parameters and other headers—and produce "smart" invalidations and cache keys. Additionally, it enables the WebAccelerator system to decide when to invalidate objects and how to identify single pieces of content.

By leveraging Dynamic Caching, WebAccelerator can respond directly to up to 80% of the most computationally expensive user requests without involving the rest of the site infrastructure. In addition, WebAccelerator will not be confused by application semantics, and never sends invalid items from the cache.

Static Caching

As an extension of its Dynamic Caching capability, WebAccelerator also provides static caching. Static caching simply serves objects—typically images, javascript, stylesheets—as long as they haven't passed their expiration date. Even though static caching is likely to already exist in an application's computing infrastructure, enabling WebAccelerator to serve static objects removes another overhead operation from the origin application.

Identifying Content

To cache applications, it is necessary to precisely identify individual pieces of content. The first time WebAccelerator receives a request for a particular piece of content, it proxies the request to the origin servers and retrieves that content for the user. When WebAccelerator receives the page—before sending the response to the client—it transforms a copy of the page into a compiled internal representation. WebAccelerator uses these compiled responses to recreate a page in response to an HTTP request.

Compiled responses consist of an internal representation of the page as it was received from the origin site, as well as instructions describing how to recreate the page from the internal representation, including information needed to update the page with any random or rotating content.

WebAccelerator also assigns a Unique Content Identifier (UCI) to the compiled response, based on elements present in the request such as the URI, query parameters, etc. This information is stored in cache as a compiled response under the UCI, which is used for both the request and the compiled response that is created to service the request. When a future request is received that contains the same elements and generates the same UCI, this cached response is found and used to respond to the request.

Using these elements as part of the identifier makes it easy to match future requests to the correct content in cache. HTTP request elements that do not affect page content are ignored and are not used in the UCI. As a result, the values set for those elements are not used to identify unique instances of cached content.

But not all elements in a request indicate a unique response. For example, two requests with the same URI, same method, and same query parameters but different cookies might still generate an identical response from your origin servers. By default, WebAccelerator assumes that certain elements cause the content to vary, and certain other elements do not. This information is also used to create a UCI, and can be configured using the WebAccelerator's variation policies.

Examples

Consider a help-desk application that manages a constantly changing set of trouble tickets. While each ticket may be referenced in many ways, each represents a single object to be cached. To identify individual tickets, the application relies on unique IDs generated by its database. Different users can view the same tickets. Requests might look like the following:

GET /helpdeskapp/viewticket.asp?ticketid=121& parentname=Login HTTP/1.1 Host: helpdesk.mycompany.com Cookie: userid=323; sessionid=3xx3s

Here, the WebAccelerator would be configured to identify and cache the ticket using this key:

[/helpdeskapp/viewticket.asp, ticketid=121].

Now, contrast this to another application, such as an email client, that fundamentally relies on user identity to generate content. When caching the inbox, requests look like the following statement:

GET /mymail/inbox.asp?page=1& parentname=Login HTTP/1.1 Host: mymail.mycompany.com Cookie: userid=323; sessionid=3xx3s

The WebAccelerator can then be configured to identify this object using the following as the key:

[/mymail/inbox.asp, page=1, userid=323].

To effectively cache dynamic content, it is necessary to maintain flexibility in identifying content. To maintain functionality, the system must always cache different pieces of content as separate objects, and there must be one-to-one relationships between true objects and cache keys.

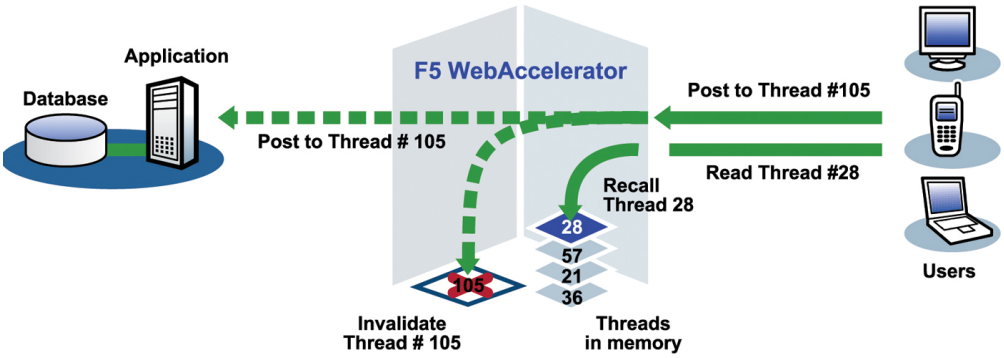
Maintaining Application Fidelity

Content generated by a web application changes constantly. The events that drive changes fall into three broad classes: time, user events, and application events. In each, the WebAccelerator has mechanisms within Dynamic Caching that can be used to describe when changes occur and which cached objects are affected. The WebAccelerator supports cache invalidation based on:

- **Expiration times,** which are useful for caching static, or nearly static, content. An example is an application generating weekly financial reports. Past reports can be cached indefinitely, and the report titled "current" is good for one week. The WebAccelerator invalidates the cached object as soon as it "expires."

- **User events,** which are caused by user interaction with the application. The WebAccelerator watches for HTTP requests that match certain criteria known to change the state of the application. When WebAccelerator finds such a request, the WebAccelerator executes a rule to invalidate the relevant section of the cache. Message boards provide an example. Individual threads are read far more often than they are written to, so there is great value in caching. To implement Dynamic Caching, the WebAccelerator is configured to recognize when a user is posting to a particular thread (See Figure 1 below). It caches the text of the thread—the first time pulling it directly from the application and, on succeeding occasions, simply from the cache. When someone posts to a thread, the WebAccelerator recognizes that and invalidates the cached objects associated with that thread only. When the next user asks for the thread, the WebAccelerator retrieves it from the application, re-seeding the cache.

Figure 1



User events invalidate individual objects in the F5 WebAccelerator cache

- **Application events,** which originate from outside the user-application interaction. The WebAccelerator handles application events by accepting standards-based XML invalidation messages (ESI) from the application when an application event occurs.

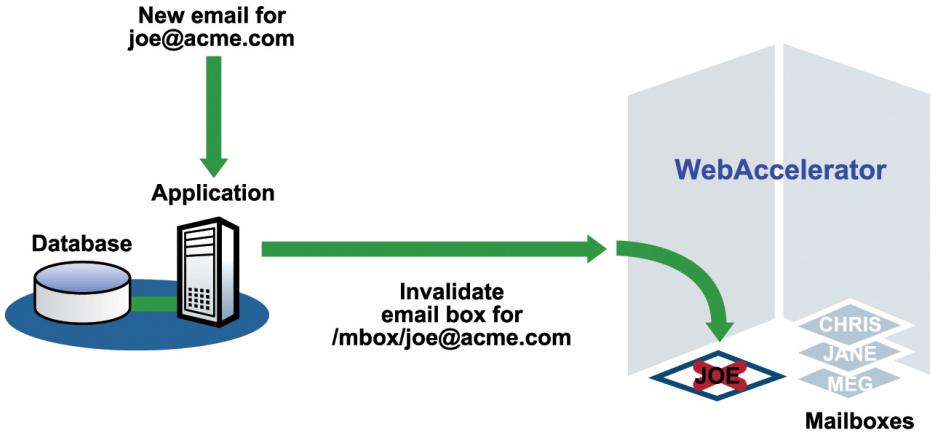
XML Cache Invalidation Message Example

The following XML cache invalidation message invalidates all objects generated by /myapp/somepage.jsp that were created using the cookies plan=gold and group=ADMINISTRATOR.

```
<?xml version="1.0"?>
                <!DOCTYPE INVALIDATION SYSTEM "invalidation.dtd">
<INVALIDATION VERSION="WCS-1.0">
   <OBJECT>
      <ADVANCEDSELECTOR URIPREFIX="/myapp/somepage.jsp">
           <COOKIE NAME="plan" VALUE="gold"/>
           <COOKIE NAME="group" VALUE="ADMINSTRATOR"/>
      </ADVANCEDSELECTOR>
</OBJECT>
</INVALIDATION>
```

Receipt of a new message is an example of an external event for an email application (See Figure 2 below). When there is no new email, the WebAccelerator serves the response directly from cache. When a new message arrives, notification goes from the email application to the WebAccelerator indicating which user's inbox has changed. Then, when the user accesses their inbox again, the WebAccelerator forwards the request to the email application, and the inbox listing is updated.

Figure 2



Application events invalidate individual objects in the WebAccelerator cache

Another example is a product price change on an e-commerce site. Price changes occur often, and perhaps one percent of the pages on an e-commerce site change daily. When a price change occurs, the WebAccelerator deletes the previously cached page. This would not happen with static caching from other vendors.

Conclusion

WebAccelerator has solved the persistent problem of caching dynamic content by implementing two key capabilities:

- A sophisticated matching algorithm that links fully qualified user queries to cached content
- A cache invalidation mechanism triggered by application and user events

Dynamic Caching reduces server load and latency up to 80 percent, improves user performance, reduces infrastructure costs, delivers stand-in capability during unanticipated or planned site outages, and provides pinpoint control of content accuracy without changes to site architecture or code. In addition, it seamlessly integrates with applications, and supports a wide variety of software and site types not dependent on particular infrastructure elements.

About F5

F5 Networks is the global leader in Application Delivery Networking. F5 provides solutions that make applications secure, fast, and available for everyone, helping organizations get the most out of their investment. By adding intelligence and manageability into the network to offload applications, F5 optimizes applications and allows them to work faster and consume fewer resources. F5's extensible architecture intelligently integrates application optimization, protects the application and the network, and delivers application reliability—all on one universal platform. Over 10,000 organizations and service providers worldwide trust F5 to keep their applications running. The company is headquartered in Seattle, Washington with offices worldwide. For more information, go to www.f5.com.