Data Caching Patterns

Tony C. Shan
IBM, USA

Winnie W. Hua
CTS Inc., USA

INTRODUCTION

As information technology (IT) has become part of business in today's globalized economy, increasingly higher performance of information systems is demanded by the business models to support various business operations and help the business compete and succeed. IT must strive to be nimble and adaptive to provide a higher level of services and, at the same time, reduce the total cost of ownership (TCO). In most situations, the current enterprise infrastructure must be extended to get the most out of the existing investments. Creating innovative solutions is an effective approach to achieve this goal, and scalable data management is one of the most valuable innovations.

BACKGROUND

Definition

In general, a data cache is defined as a data block that contains frequently accessed contents in a text or binary format. A persistent data block is saved in storage at either the client or the server side. Alternatively, a transient data block may be stored in a memory buffer for the lifetime of an application, a user session, or a single-client request. Caching is a widely used technique to boost the performance of data access. When a program needs to access a particular data element, the process first checks the data cache to verify whether the element has been previously retrieved and stored. If a match is found, the application will use the data directly from the cache, rather than accessing to the data source again. As a result, a drastic performance gain is realized, since the data access in RAM is significantly faster than that to a disk or external resource over the network. In addition, a cached data element is usually in a ready-to-use form, so that little or no transformation and initialization is needed, resulting in higher performance in the processing.

Value Proposition

Generally speaking, many performance challenges may be resolved via the horizontal or vertical scaling. Extra servers are added in the horizontal scaling. The vertical scaling approach upgrades the existing machines with more and faster processors, additional memory, larger hard disks, and/or higher-speed network connection. In today's competitive environment, however, the ultimate challenge is a balance of the overall project cost and on-demand scalability to meet the capacity needs. Without looking into the end-to-end performance chain, particularly at the application software level, simply investing more on hardware alone usually does not fix the root cause despite the fact that it may temporarily alleviate the symptom. In other words, a more holistic approach should be taken to improve the overall architectural design. The best solution to systematically address the performance issue is usually an aggressive use of data caching technology.

DATA CACHING PATTERNS

A wide range of caching patterns can be applied individually or in combination as a means to increase application performance. Each pattern was designed with its own specific merits and addresses a certain type of data access issues. Classifying data caching patterns is a challenging task, resulting in a different outcome scheme based on the criteria applied. For example, they may be categorized as creational, structural, and behavioral. Alternatively, they may be grouped at the levels of the method, class, component, application, platform, and system.

A vast majority of today's distributed applications are developed in either Java or .NET on an n-tier architecture, which consists of a series of logical/physical layers: client, Web, application, integration, and data and enterprise resource. Accordingly, a taxonomic scheme is defined in Figure 1 to sort various caching patterns into appropriate layers. Furthermore, those patterns that can be used in multiple layers are grouped in the cross-layer category.

Client Layer

AJAX

Asynchronous JavaScript and XML (AJAX) is a Web development style in which small chunks of data in XML format are transferred from the Web server in an asynchronous manner behind the scene. As a result, a responsive interaction on the Web page is realized because there is no need to reload the entire Web page every time a user clicks a link or button on the page. In the AJAX model, the browser caches a great deal of data, and other blocks of data can be asynchronously retrieved from the servers in an on-demand or proactive fashion.

Cookie

Client-side cookies may serve as a data cache, which acts just like the hidden input fields in HTML. Persistent cookies can save the data permanently on the local hard drive of the client machine. Session cookies store the data elements temporarily in the memory, but the data contents will need to be set just once, unlike the hidden fields that have to be set by the server-side in each response. In the scenario where multiple systems collaborate to provide a service by transferring controls between each other, a cookie is a viable approach to storing the session data. However, the cookie data are sent to the server from a browser in every http request, which still generates a tremendous amount of network traffic. In case the originator sets the domain attribute, the cookie contents are invisible to other Web sites. Under the circumstance where a user opts to switch off the cookie option in the browser settings, this technique becomes worthless.

It is recommended in RFC 2109 (Kristol et al., 1997) that at least 4096 bytes per cookie are implemented, which is the size limit placed by most browsers in the market. The future browser versions tend to support 8192-byte size cookies.

DNS Resolver

The DNS lookup results may be cached to minimize query calls to the DNS server. Popular Web browsers implement the client DNS resolvers with cache. The Mozilla browser caches the DNS host entries for 15 minutes, whereas the DNS cache expiry time is set to 30 minutes in Internet Explorer (Microsoft Support, 2004).

Hidden HTML Frame

Data elements may be cached in a hidden frame in an HTML page on the client browser, which eliminates the roundtrips to the server—an inherent attribute in the hidden field pattern. Client-side scripting can be leveraged to access the data stored in a hidden frame. This method only requires the client-side resources because the data elements in the hidden frame are stored and retrieved locally from the page on a browser.

HTML Form Field

Session-related data may be stored in the hidden input fields in an HTML form. This pattern is server technology agnostic, and the hidden-field data elements may be submitted to other Web sites in addition to the original Web site that set the contents. On the other hand, the exact same chunk of data elements must be transferred back and forth between the browser and server in every hit, adding extra network bandwidth consumption in the communications. Special encoding/decoding must be implemented for binary data, via encoding schemes like *Base 64*. Moreover, the names and values of the hidden input fields may have to be scrambled or encrypted to restrain the possible hacking attempts.

Proxy Server

A proxy server is an appliance made up of software or hardware that enables indirect connections from applications to other network resources. For a dynamically constructed Web page, only changed blocks of data are sent, so that the network bandwidth consumption is significantly reduced.