Caching over the Entire User-to-Data Path in the Internet

Theo Härder

University of Kaiserslautern, Germany
haerder@informatik.uni-kl.de

Abstract. A Web client request traverses four types of Web caches, before the Web server as the origin of the requested document is reached. This client-to-server path is continued to the backend DB server if timely and transaction-consistent data is needed to generate the document. Web caching typically supports access to single Web objects kept ready somewhere in caches up to the server, whereas database caching, applied in the remaining path to the DB data, allows declarative query processing in the cache. Optimization issues in Web caches concern management of documents decomposed into templates and fragments to support dynamic Web documents with reduced network bandwidth usage and server interaction. When fragment-enabled caching of fine-grained objects can be performed in proxy caches close to the client, user-perceived delays may become minimal. On the other hand, database caching uses a full-fledged DBMS as cache manager to adaptively maintain sets of records from a remote database and to evaluate queries on them. Using so-called cache groups, we introduce the new concept of constraint-based database caching. These cache groups are constructed from parameterized cache constraints, and their use is based on the key concepts of value completeness and predicate completeness. We show how cache constraints affect the correctness of query evaluations in the cache and which optimizations they allow. Cache groups supporting practical applications must exhibit controllable load behavior for which we identify necessary conditions. Finally, we comment on future research problems.

1 Motivation

Internet-based information systems and e*-applications are growing with increasing pace and their users are placing tremendous workloads with critical response-time restrictions on the Internet and the Web servers. For these reasons, scalability, performance—in particular, minimization of user-perceived delays—, and availability are prime objectives for their system development. Most of all, various forms of caching in the Web have proven to be a valuable technique[1] towards these design goals. Three as-

T. Härder and W. Lehner (Eds.): Data Management (Wedekind Festschrift), LNCS 3551, pp. 67-89, 2005.

pects make caching attractive in the Web environment, because it effectively reduces network bandwidth usage, user-perceived latency, and workload on the origin server.

To improve response time and scalability of the applications as well as to minimize communication delays in wide-area networks, a broad spectrum of techniques has emerged in recent years to keep static Web objects (like HTML pages, XML fragments, or images) in caches in the client-to-server path. These techniques, often summarized as *Web caching*, typically support access by object identifiers and aim at locating and possibly assembling user-requested Web objects in caches near the Web client to unburden the Web traffic and to achieve minimal response times. In particular for static Web objects, it can provide various kinds of performance improvements for e*-applications [26]—a reason which amplified the setup of Web caches and the optimization of their usage by tailored replacement strategies [23] in recent years. Nowadays, however, more and more dynamically generated content is needed and offered making Web applications even more attractive and enabling new forms of business: contents' personalization, goal-oriented advertisement, interactive e-commerce, one-to-one marketing, and so on. Obviously, the way caching is performed has to respond to these new requirements. To effectively serve this trend, caches have to be aware of the internal structure of documents (Web pages) to enable selective reuse of static fragments (objects) and exchange of dynamic parts in order to assemble them to the actual document to be delivered in the most cost-effective way. Fragment-enabled caching techniques have to be developed which can distinguish and manage templates and fragments of Web documents separately.

2 The Client-to-Server Path

Conceptually, a Web request is processed as follows: A Web client (client throughout the paper) sends a query containing a URL via HTTP and the Internet to a Web server (origin server or server, for short) identified by the URL. The server processes the request, generates the answer (typically an HTML or XML document), and sends it back to the client. To solve the performance and availability problems sketched above, we add, again conceptually, a Web proxy server somewhere in the client-to-server path. Such a proxy can be used in a number of ways, including

- caching documents and parts thereof
- converting data to HTML/XML format so it is readable by a client browser
- providing Internet access for companies using private networks
- selectively controlling access to the Internet based on the submitted URL
- permitting and restricting client access to the Internet based on the client IP address

In this contribution, we concentrate on the caching functionality of proxies and discuss the client-to-server path how it evolved during the recent past. Caches, in general, store

1. "The three most important parts of any Internet application are caching, caching, and, of course, caching …"—Larry Ellison, Oracle Chairman & CEO.