Database Caching – Towards a Cost Model for Populating Cache Groups

Theo Härder and Andreas Bühmann
Department of Computer Science, University of Kaiserslautern, P. O. Box 3049, D-67653 Kaiserslautern, Germany
{haerder, buehmann}@informatik.uni-kl.de

Abstract. Web caching keeps single Web objects ready somewhere in caches in the user-to-server path, whereas database caching uses full-fledged database management systems as caches 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 and domain 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. For such safe cache groups, the cost trade-off for record loading and predicate evaluation saving has to be observed during their design. Therefore, we analyze their load overhead and propose a population estimation algorithm to be used for a cache group advisor.

1 Introduction

Transactional Web applications (TWAs) in various domains (often called e*-applications) dramatically grow in number and complexity. At the same time, each application faces increasing demands regarding data volumes and workloads to be processed efficiently. In such situations, caching is a proven concept to improve response time and scalability of the applications as well as to minimize communication delays in wide-area networks. For this reason, 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 user-to-server path (client-side caches, proxies of various types, CDNs).

As the TWAs must deliver more and more dynamic and frequently updated content, this so-called Web caching [9,11] should be complemented by techniques that are aware of the consistency and completeness requirements of cached data (whose source is dynamically changed in backend databases) and that, at the same time, adaptively respond to changing workloads. Attempts targeting these objectives are called database caching, for which several different solutions have been proposed in recent years [2,3,4]. Currently many database vendors are developing prototype systems or are just extending their current products [8,10].
What is the technical challenge of all these approaches? When user requests require responses to be assembled from static and dynamic contents somewhere in a Web cache, the dynamic portion is generated by a remote application server, which in turn asks the backend DB server for up-to-date information, thus causing substantial latency. An obvious reaction to this performance problem is the migration of application servers to data centers closer to the users: Figure 1 illustrates that clients select one of the replicated Web servers “close” to them in order to minimize its response time. This optimization is amplified if the associated application servers can instantly provide the expected data – frequently indicated by geographical contexts. But the displacement of application servers to the edge of the Web alone is not sufficient; conversely it would dramatically degrade the efficiency of DB support because of the frequent round trips to the then remote backend DB server. As a consequence, primarily used data should be kept close to the application servers in so-called DB caches. A flexible solution should not only support database caching at mid-tier nodes of central enterprise infrastructures [10], but also at edge servers of content delivery networks or remote data centers.

Another important aspect of a practical solution is to achieve full cache transparency for the applications, i.e., modifications of the application programming interface are not tolerated. Such a property gives the cache manager the choice at run time to process a query locally or to send it to the backend DB server, e.g., to comply with strict consistency requirements. Cache transparency typically requires that each DB object is represented only once in a cache and that it exhibits the same properties (name, type, etc.) as in the backend.

The use of SQL implies another challenge because of its declarative and set-oriented nature. This means that, to be useful, the cache manager has to guarantee that queries can be processed in the DB cache, i.e., the sets of records (of various types) satisfying the corresponding predicates – denoted as predicate extensions – must be completely in the cache. This completeness condition, the so-called predicate completeness, ensures that the query evaluation semantics is equivalent to the one provided by the backend.

A federated query facility [2,8] allows cooperative predicate evaluation by multiple DB servers. This property is very important for cache use, because