Efficient Client Caching for Distributed Object Storage Systems

M. Castro, A. Adya, B. Liskov, U. Maheshwari, A.C. Myers, and L.Shrira
Laboratory for Computer Science,
Massachusetts Institute of Technology,
Cambridge, MA 02139

There are two approaches to client cache management in distributed object storage systems: caching pages and caching objects. Pages are typically 4 KB or 8 KB, and most objects are much smaller (e.g. the average object size in our implementation of the OO7 benchmark is 40 bytes). We have designed and implemented a new client cache management scheme that outperforms object and page caching by combining the best properties of both schemes.

Caching pages is simpler than caching objects because pages are fixed-size units. However, spatial locality in a page can be low, because it is difficult to cluster objects in pages to suit all possible uses. Therefore, it is common for a client to fetch a large page and use only a small fraction of it. This wastes network bandwidth, client and server CPU time, and client cache space (since most cache replacement schemes retain recently fetched pages for a certain period, e.g. LRU-K). Another related consequence of poor clustering is low cache utilization; clients cache full pages even when few objects in those pages are being used by the application.

Object caching solves the low cache utilization problem, because it allows clients to cache useful objects in a page and discard useless ones, but it introduces a set of new problems. Objects are variable-sized units, which makes cache management more difficult: it introduces fragmentation, and complicates allocation and eviction. Fragmentation reduces effective cache utilization and complex allocation slows down fetches. Also, since there are many more objects than pages, maintaining per-object usage statistics can introduce high space and time overheads.

We have designed and implemented a new cache management scheme that solves the problems with object and page caching by combining the best properties of both. The client cache is managed in terms of fixed-size 1 KB units called *subpages*, which are the unit of fetching from the server. Objects that are smaller than a subpage are entirely contained within a subpage; big objects (e.g. large arrays) are organized as trees. Even 1 KB subpages are large enough that space lost due to internal fragmentation in our implementation of OO7 is about 2%.

We use a simple adaptive fetching mechanism that fetches a small number of subpages when clustering is poor and a large number when clustering is good. This mechanism increases or decreases the number of subpages fetched depending on whether the fraction of the recently-fetched subpages that were used is high or low. Thus, this mechanism retains the good performance of page fetching when clustering is good and avoids the waste of resources otherwise.

Most importantly, our scheme achieves better cache utilization than either page or object caching by using a hybrid approach. We maintain per-object usage statistics, using a novel scheme with low space and time overheads. The usage value of a subpage is the sum over all its objects of the product of object size and object usage. Subpage usage is computed lazily; the system cycles through the cache and, at every fetch, computes the usage of a set of subpages while waiting for the fetch reply. Subpages whose usage was computed recently are maintained in a priority queue.

Eviction proceeds as follows: the subpages with lowest usage are removed from the priority queue; the objects in those subpages whose usage is above a certain threshold are compacted; and those whose usage is below the threshold are discarded. This process splits the set of subpages removed from the priority queue in two subsets: one whose subpages are full of hot objects, and another whose subpages are empty and can be used to receive the new subpages in the next fetch.

Eviction and computing subpage usage have essentially zero cost because they are performed while waiting for fetch replies. Furthermore, we have solved the problems with object caching: fragmentation is eliminated, and cache allocation is simple. Therefore, we obtain better cache utilization than both object and page caching and retain the efficient fetch path of page caching. Our preliminary experiments confirm this intuition; they indicate that the new scheme is very efficient and significantly outperforms other object storage systems.