

## 12.10 LRU Replacement

What replacement policy should be used? There are two issues. First, to increase the hit ratio, the replacement policy should retain those items that will be referenced most frequently. Second, the replacement policy should be inexpensive to implement.

One replacement policy is extremely popular. Known as *Least Recently Used (LRU)*, the policy specifies replacing the item that was referenced the longest time in the past<sup>†</sup>.

LRU is extremely easy to implement. The cache mechanism keeps a list of data items that are currently in the cache. When an item is referenced, the item moves to the front of the list; when replacement is needed, the item at the back of the list is removed.

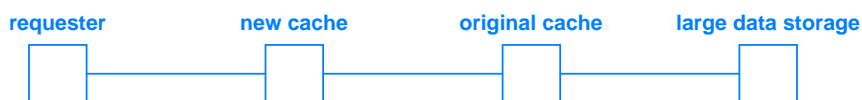
LRU works well in many situations. In cases where the set of requests has a high locality of reference (i.e., the cases where a cache can improve performance), a few items will be referenced again and again. LRU tends to keep those items in the cache, which means the cost of access is kept low.

We can summarize:

*When its storage is full and a new item arrives, a cache must choose whether to retain the current set of items or replace one of the current items with the new item. The Least Recently Used policy is a popular choice for replacement because it is trivial to implement and tends to keep items that will be requested again.*

## 12.11 Multi-level Cache Hierarchy

One of the most unexpected and astonishing aspects of caching arises from an optimization that uses caching to improve the performance of a cache! To understand how such an optimization is possible, recall that the insertion of a cache lowers the cost of retrieving items by placing some of the items closer to the requester. Now imagine an additional cache placed between the requester and the existing cache as Figure 12.3 illustrates.



**Figure 12.3** The organization of a system with an additional cache inserted.

Can a second cache improve performance? Yes, provided the cost to access the new cache is lower than the cost to access the original cache (e.g., the new cache is closer to the requester). In essence, the cost equation becomes:

$$Cost = r_1 C_{h1} + r_2 C_{h2} + (1 - r_1 - r_2) C_m$$

<sup>†</sup>Note that “least recently” refers to how long ago the item was last referenced, not to the number of accesses.