Core caching components

Г.Р. Гарифуллина, Университет ИТМО, Санкт-Петербург
Научный руководитель -- старший преподаватель,
А.С.Сомко, Университет ИТМО, Санкт-Петербург

The continuous development of Information Technologies causes not only gradual improvement of human lives, but also increases complexity of the informational systems, which can affect the speed of application. The more calculations the system requires, the more time it will spend and the worse the user experience will be. Therefore complex systems use different optimization techniques to leverage response time. One of the most widely used techniques is caching.

Caching is the process of storing the result of long operations for later use. A cache is a memory area that is used by a client to store source data and is closer to the client than the source. Consequently, access to cached data is usually much faster. Caching speeds up data collection, calculation of complex operations and reduces the load on the source.

The theoretical significance of this work is the study of various existing approaches for caching. Among the main components, which affect the efficiency of the caching algorithm are:

1. admission policy,
2. page replacement policy (cache size management methods),
3. synchronization methods.

Admission policy

There are two basic cache filling strategies: proactive (active) and reactive (lazy). In the first case the system predicts and loads the necessary data in advance, so during the request the required values are read from the cache. In contrast, if reactive admission policy is used, the data is loaded into the cache when it is first accessed. Therefore, only the requested data is loaded, but the first access to data takes much more time than subsequent ones.

In real systems the combined method is often used: the most popular data is preloaded on the application start, and the remaining values are loaded lazily.

Page replacement policy

The cache size is limited, so in order to write new values, it's necessary to free up the cache. To solve this problem, different page replacement algorithms are used. For example, one of the most used and simple element replacement algorithms are First In Last Out (FILO), First In First Out (FIFO), Least Frequently Used (LFU), Least Recently Used (LRU) and etc. In addition to the above methods, there are many other more advanced replacement algorithms. For example, machine learning can be used to predict the next candidate for deletion. Advanced replacement policies require more memory and (or) time to perform, so the choice of replacement method and its complexity should correspond to its further use.

Synchronization methods

Any cached data is potentially stale, as it can be updated in the remote system. Data inconsistency can lead to incorrect behavior of the application or reduce its performance. Depending on the structure of the system, there are various ways to synchronize data: Write-through Caching, Time Based Expiry and Active Expiry.

A write-through cache is a cache that supports updates from the local system, but not from remote one. Time Based Expiry and Active Expiry methods are more flexible and support independent change of local and remote systems. First method deletes items from the cache if it exists for more than a predetermined time. Second method uses the remote system to inform the local system to update cache. Therefore, unnecessary deletions do not occur.

Conclusion

Caching increases delivery performance of applications. Efficiency of caching algorithms depend on the correct choice of admission policy, page replacement policy and synchronization methods.