Abstract

Traditional caches see a fixed-size page; a Web cache, on the other hand, sees complete objects (text files, images, or video clips), which vary considerably in size. In addition, a traditional cache deals with addresses, while a Web cache can potentially deduce more contextual information from its objects. Web objects are predominantly read-only, taking implementation of cache coherence easier. The response times of Web accesses are in the order of seconds (versus milliseconds for system access), which allows for more elaborate caching algorithms. Finally, a Web cache encounters more dimensions of dependence than are taken into account by traditional methods. Primary cache replacement algorithms consider arrival time as the only one factor as the basis of their functionality. They disregard parameters such as page size, fetching delay, reference rate, and invalidation cost and invalidation frequency of a web object. Considering these parameters produces better and more apt results with greater efficiency, thereby surpassing traditional and conventional algorithms such as LRU, LIFO and LFU in performance and accuracy. Most of them are favorable to objects with homogenous sizes. Also, many of these algorithms depend on manual interference to find quick
cures for symptoms instead of understanding the core issues. Because the cache space is limited and no technology can be as suitable to cater to each user’s request separately, we need caching algorithms that are intelligent and adapt to the available resources and utilize them optimally. Systems must evolve towards more scalable, adaptive, efficient and self-configuring web caching systems to effectively support the phenomenal growth in demand for web content on the internet. Adaptive caching views caching problems as a way of optimizing global data dissemination. Studies have shown that adaptive algorithms yield better results than conventional caching algorithms.

References

Index Terms

Computer Science
Information Sciences

Keywords

Web Caching, Caching Cache Algorithm.