6F-06

PSS-W: A New Efficient Caching Policy for the World Wide Web

Kai Cheng and Yahiko Kambayashi
Graduate School of Informatics, Kyoto University

1 Introduction

Web caching is a technique by storing popular objects near the clients to improve response times, save network bandwidth and distribute workload from server "hot spots". In general, three factors are most important to Web caching: *object size*, *time since last access* and *reference times*. However, due to the difficulty in incorporating all three factors in a cache policy, current policies are mostly based on subset of these factors.

In this paper, we propose a new cost-to-benefit model for Web caching that takes into account all the three factors. We then give a new implementation scheme, namely *PSS-W*, which is a good approximation to our model but very simple to implement. Trace-driven simulations demonstrate PSS-W outperforms its predecessor PSS very much in terms of byte hit rate meanwhile its hit rate is at least as good as PSS.

2 Design of PSS-W Caching Policy

2.1 SLRU/PSS

Due to the protocol restrictions, Web objects need to be treated as a whole. To cope with *variable sizes*, a Size-Adjusted LRU or SLRU has been proposed in a recent paper as an generalization to traditional LRU[1]. SLRU is based on the following *cost-to-benefit* model: an object being cached will incur a cost of occupying a cache space ($size$) for certain time ($atime$), meanwhile it will benefit from saving a latency (lat) to download the object from Web site. When cache space runs out, the cache will purge the object with the largest *cost-to-benefit*,

$$(atime * size)/lat$$

Since the latency of the same object usually varies significantly from one access to another, it is often factored out in Web caching and let $lat = 1$.

SLRU is unrealistic to implement in practice because of the difficulty in comparing the product of $size$ and $atime$ for every object in cache. A practical variant, Pyramidal Selection Scheme or PSS is designed to implement SLRU. The caching mechanism of PSS is:

1. Objects are classified into a limited number of groups based on their size levels, i.e. $\lfloor log_2(size) \rfloor$;
2. Each group is maintained using a LRU mechanism. A hit will make the hit object move to the MRU end of this LRU queue;
3. The SLRU policy is only applied to a limited set of LRUs from all nonempty groups to make final decision. The object with largest $(atime * size)$ will be purged from the cache.

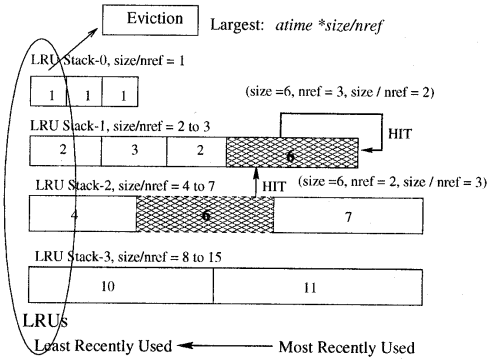

Figure 1: PSS-W : An Efficient Improvement to PSS

PSS biases towards the smaller objects, so that after a long time, the total number of objects in cache will grow but the average object size will decrease. This has a positive effect on hit rate but is negative to byte hit rate. In fact, there are quite a fraction of smaller objects only get few re-references, on the contrary, some bigger objects are really popular, however they have little chance to stay in cache for more time to gather subsequent references.

2.2 PSS-W

The limitation of SLRU/PSS roots in the SLRU policy which only takes into account $atime$ and $size$. This is unfair to those objects with more accesses because the repeated accesses are served by cache and these accesses should share the cost of caching the object. Numerous papers show that the reference frequency or *nref* is a more important factor to Web caching. Here we propose an **averaged cost** $(atime * size/nref)$ for each object in cache. The cache should purge the object with largest value of

$$(atime * size/nref)/lat$$

so as to gain more benefits. Here for the same reason as above, we set $lat = 1$

Similarly, we propose a practical scheme to implement the extended SLRU. The basic idea is to classify objects based on the value level of $(size/nref)$, instead of $size$. Since $nref$ changes with each hit, it is necessary to determine which group the object belongs. This new scheme is referred to as *PSS-W*. The outline of PSS-W is depicted in Figure 1.

1. Objects are classified into a limited number of groups according to $\lfloor log_2(size/nref) \rfloor$, instead of $\lfloor log_2(size) \rfloor$.

2. Each group is managed using a LRU policy. A hit may make the hit object move to new LRU queue according to its new value of $\lfloor log_2(size/nref) \rfloor$.

3. The extended SLRU policy is applied to the eviction candidates from all nonempty groups, purging the object with largest $(atime \cdot size/nref)$.

One important concern is the *temporary locality*. In LRU-based model, even average occurrence is same, temporary locality will influence the result. For example, if requests occur periodically (Figure 2(a)), the object will stay in the cache. If accesses are locally concentrated, the object will not be kept in the cache (Figure 2(b)) even if average occurrence is same.

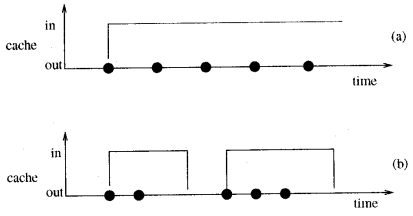

Figure 2: Influence of locality for LRU-based cache

We will use accumulation of occurrences so that the effect of localities will be removed. Figure 3(a) and Figure 3(b) show the cases. If sum is greater than 0, the 0bject will stay in the cache.

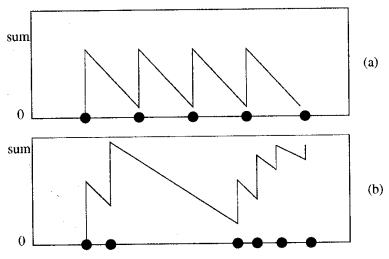

Figure 3: Removal of the effect of locality

One additional problem is how many groups should be set in a PSS-W or PSS scheme. Our answer is not more than 24, because objects larger than 16MB(2^{24}B) are very rare.

3 Performance Evaluation

Based on trace-driven simulations, we compare PSS with PSS-W in terms of hit rate and byte hit rate. We use a one-week proxy trace to drive our simulator. It contains 1,848,319 client requests with a maximum hit rate 0.228 and byte hit rate 0.245

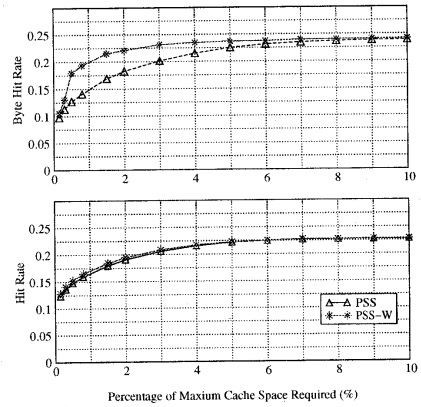

Figure 4: Simulation Results (Upper:BHR,Lower:HR)

PSS-W has been proven an efficient improvement to PSS. From Figure 4, we can see PSS-W outperforms PSS very much in byte hit rate meanwhile the hit rate keeps at lest the same as PSS. It demonstrates that objects with more accesses are more popular and worth to stay in cache for long time. In fact, we have compare PSS-W to various other algorithms, demonstrating that PSS-W is the best algorithm in all cases.

4 Conclusion

In this paper, we propose a new efficient cache replacement policy based on the consideration of reference frequency. The new caching policy factors in three key parameters to Web access so that it can achieve a high cache performance. As future work, it is necessary to explore the mathematical/statistical explanation to this approach.

References

[1] Charu Aggarwal, Joel L. Wolf, and Philip S. Yu. Caching on the World Wide Web. *IEEE transactions on knowledge and data engineering*, 11(1), 1999.

[2] Yahiko Kambayashi. *Fundamental Theory of Information Science*. SHOKODO CO., LTD., May 1997.