关于缓存替换算法



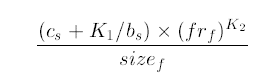


李英平

摘 要: 简述十种常见的缓存替换算法

关键词: 缓存替换算法

常见的缓存替换算法主要有以下十种：

1. Least-Recently-Used(LRU) - 最近最少使用  
   替换掉最近被请求最少的文档。这一传统策略在实际中应用最广。在CPU缓存淘汰和虚拟内存系统中效果很好。然而直接应用与代理缓存效果欠佳，因为Web访问的时间局部性常常变化很大。
2. Least-Frequently-Used(LFU) - 最不经常使用  
   替换掉访问次数最少的。这一策略意图保留最常用的、最流行的对象，替换掉很少使用的那些。然而，有的文档可能有很高的使用频率，但之后再也不会用到。传统 的LFU策略没有提供任何移除这类文件的机制，因此会导致“缓存污染(Cache Pollution)”，即一个先前流行的缓存对象会在缓存中驻留很长时间，这样，就阻碍了新进来可能会流行的对象对它的替代。
3. SIZE  
   替换size最大的对象。这一策略通过淘汰一个大对象而不是多个小对象来提高命中率。不过，可能有些进入缓存的小对象永远不会再被访问。SIZE策略没有提供淘汰这类对象的机制，也会导致“缓存污染”。
4. LRU-Threshold  
   不缓存超过某一size的对象，其它与LRU相同。
5. Log(Size) + LRU  
   替换size最大的对象，当size相同时，按LRU进行替换
6. Hyper-G  
   LFU的改进版，同时考虑上次访问时间和对象size
7. Pitkow/Recker  
   替换最近最少使用的对象，除非所有对象都是今天访问过的。如果是这样，则替换掉最大的对象。这一策略试图符合每日访问web网页的特定模式。这一策略也被建议在每天结束是运行，以释放被“旧的”，最近最少使用的对象占用的空间。
8. Lowest-Latency-First  
   替换下载时间最少的文档。显然它的目标是最小化平均延迟。
9. Hybrid  
   Hybrid有另外一个目标，减少平均延迟。对缓存中的每个文档都会计算一个保留效用（utility of retaining）。保留效用最低的对象会被替换掉。位于服务器s的文档f的效用函数定义如下：  
   Cs: 与服务器s的连接时间  
   bs: 服务器s的带宽  
   frf: f的使用频率  
   sizef: f的size，单位字节  
     
   K1和K2是常量，Cs和bs是根据最近从服务器s获取文档的时间进行估计的。
10. Lowest Relative Value(LRV)  
    LRV也是基于计算缓存中文档的保留效用。然后替换保留效用最低的文档。有点复杂，实际应用价值不大，就不详述了。

关于他们孰优孰劣：

* [ASAWF95, WASAF96]所做的性能研究表明在命中率方面，SIZE优于LFU， LRU-Threshold, Log(Size) + LRU, Hyper-G以及Pitkow/Recker。
* [WASAF96]还显示大多数情况下，SIZE要优于LRU。然而，[LRV97]的研究表明在字节命中率方面LRU要优于SIZE。
* 大多数情况下，LRU都要比LFU好[Cao-Irani97]。在极少情况下，LFU表现比LRU好。
* 在最小化延迟方面，[WA97]表明Hybrid表现优于Lowest-Latency-First。
* [LRV97]的性能研究表明，无论在命中率还是字节命中率方面，LRV表现都比LRU和SIZE好。
* 这缩小了我们的选择范围，似乎我们只需要考虑这些策略：LRU, SIZE, Hybrid, LRV。Hybrid和LRV涉及到大量的参数，理论价值大于实际应用。
* [Cao-Irani97]所做的性能研究表明，Greedy-Dual-Size算法是目前最好的，在不同的维度都优于LRU, SIZE, Hybrid和LRV。