Journal of Theoretical and Applied Information Technology
10th January 2013. Vol. 47 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

THE CACHE REPLACEMENT POLICY AND ITS SIMULATION RESULTS

[1]ZHU QIANG, [2]SUN YUQIANG
[1]Zhejiang University of Media and Communications, Hangzhou 310018, P.R. China
[2]Changzhou University, Changzhou 213022, P.R. China
E-mail: [1]zq.hz@163.com, [2]sunyuqiang@126.com

ABSTRACT

To optimize the multimedia proxy server effectively, this paper proposes a gain-based cache replacement policy by using the multimedia proxy server gain model. The performance of the proposed gain-based policy is evaluated by simulation experiments.

Keywords: *Multimedia Proxy Server, Media Data Object, Cache Replacement*

1. INTRODUCTION

Cache replacement is a key problem in multimedia proxy server cache management. The conventional cache replacement policy aims to increase the proxy server cache-hit rate. However, cache-hit rate is just one of the proxy server performance indexes and cannot fulfill proxy design objective alone [1]-[4]. Due to the resource limit, different performance indexes may conflict with each other. To achieve a balance on the resource consumed by different performance indexes, a gain model of cache resource against performance index is desirable. This paper constructs such a gain model, and proposes a multi-proxy cache replacement policy based on it. To evaluate this policy's performance, simulation experiments are conducted. We compare it with some typical cache replacement policies in terms of the improvement on video quality, start-up delay, network resource cost.

Let's study how does proxy server cache management policy have an influence on these performance indexes.

2. GAIN-BASED CACHE REPLACEMENT POLICY

2.1 Multimedia Proxy Server Gain Model

The multimedia proxy server manages the cached media data in the way of fine granularity. A multimedia stream can be divided into slices in the time axis. For the stream encoded with extensible video encoding technology, it can be further divided into layers with each layer corresponding to certain bit rate and media quality [5]. The data on a layer in a time slice of a stream is the basic media data unit, called as media data object. The channel encoding brings in the redundant data for each media data object, and it is called media redundant object. They together compose the media object, which is the basic cache management unit on the multimedia proxy server. To evaluate the benefit of caching a media object, we quantify its caching gain. The caching gain of a media object consists of three sub-caching gains, which are media quality sub gain, start-up delay sub gain and network cost sub gain.

Media quality sub gain: It represents a media object, say gs(v)'s contribution to the media quality offered to users. Media quality sub gain is defined as

$$Rev_Q(g_s(v)) = PSNR(g_s(v)) \times f(g_s(v))$$

where PSNR(gs(v)) is the peak value of gs(v)'s signal-to-noise ratio, and f(gs(v)) is the gs(v)'s accessing rate. Similarly for the media redundant object gc(v), its media quality sub gain is defined as

$$Rev_Q(g_c(v)) = PSNR(g_c(v)) \times f(g_c(v)).$$

Start-up delay sub gain: caching the prefix of a media stream can decrease its start-up delay. We calculate the start-up delay sub gain for a media object based on whether it is part of the prefix. Start-up delay sub gain is defined as

$$Rev_L(g_s(v)) = \begin{cases} Delay(RTT(v)) \times f(g_s(v)), & \text{if } g_s \in Prefix \\ 0, & \text{otherwise} \end{cases}$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

where Delay(RTT(v)) is the delay of the media object from the video server to the proxy server. If a media object is part of the prefix, caching it can reduce the start-up delay; otherwise, its start-up delay sub gain is 0.

Network cost sub gain: A media data object's network cost sub gain has two parts. First, a media data object has the transmission cost, which can be evaluated by the distance between the Internet video server and the multimedia proxy server. This is called as network transmission sub gain, and defined as

$$Rev_T(g_s(v)) = Size(g_s(v)) \times Dist(RTT(v)) \times f(g_s(v))$$

where Size(gs(v)) is the size of the media object, Dist(RTT(v)) is the distance between the video server and the proxy server, which can be measured by using RTT, hops and so on. Second, the conditions of the links from video servers to the proxy server are different. We use link usage rate μ(j,t) to evaluate the link load. For the media object gs(v), its network usage sub gain is

$$Rev_U(g_s(v)) = f(g_s(v)) \times Size(g_s(v)) \times \mu(j,t).$$

Thus a media object's network cost sub gain is

$$Rev_N(g_s(v)) = Rev_T(g_s(v)) + Rev_U(g_s(v)).$$

2.2 Maximizing Caching Gain

By quantifying the media quality sub gain, start-up delay sub gain and network cost sub gain for a media object, the media object's overall caching gain is defined as follows:

$$Rev(g_s(v)) = p_Q \times Rev_Q(g_s(v)) + p_L \times Rev_L(g_s(v)) + p_N \times Rev_N(g_s(v))$$

where pQ、pL、pN are the weighs of media quality sub gain, and start-up delay sub gain and network cost sub gain respectively. Their values will be determined in the real life application [6].

Based on the media object's caching gain, we define the optimization goal of the caching on the multimedia proxy server to be the maximal multimedia proxy's overall caching gain. Let I be the set of media object cached in the multimedia proxy server, and S be the size of the multimedia proxy's cache. Then the cache replacement policy's optimization goal is

$$\max \sum_{g(v)\in I} Rev(g(v))$$
$$s.t. \sum_{g(v)\in I} Size(g(v)) < S.$$

This optimization problem cannot be solved in polynomial time. But if we assume that a media object is very smaller than the total cache size in the Proxy ($Size(g(v)) << S$), then the above problem can be simplified to

$$\sum_{g(v)\in I} Size(g(v)) = S$$

and we can use the offline greedy algorithm to achieve the feasible solution. Each media object is assigned a caching gain density, which is

$$Gain(g(v)) = \frac{Rev(g(v))}{Size(g(v))}.$$

The media objects are sorted in a queue in the descendant order of the caching gain density. One media object is taken from the head of the queue at a time, and put it into the cache storage, until there is no media object left in the queue or there is no more free space in the cache storage. In this way, a feasible solution can be achieved [7]-[8].

2.3 Gain-Based Cache Replacement Policy

The above mentioned greedy algorithm works offline, but it is not applicable in the online real time situation, in which cache replacement policy must process the current media object in a real time way without the complete information about the media stream. When the cache replacement policy uses the media object to fill the proxy cache, it cannot compute and optimize the caching gain on a go. In the system there are time-varying parameters leading to a time varying caching gain.

In order to optimize the caching in the proxy server effectively in the real time situation, we propose the gain-based cache replacement policy, which is described as below:

```
Gain-based Replacement Policy (for streaming Si)
While (not finished)
  {
//In the stream, look for the media object, which
//has the maximal caching-gain density and is
//independent of the object other than the cache.
gs=FindMaxGainDataObject(Si);
//look for the cacheable media redundant object
//with the maximal gain density.
gc=FindMaxGainRedundantObject(Cache);
//take the one with bigger gain as the replacing
//candidate
gin=MaxGainObject(gs, gc);
while(EmptySizeInCache<Size(gin))
//Prepare caching space for the candidate
  {
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

g_{out} = FindMaxGainObject(Cache);
//look for the independent media object in the //cache with minimal caching gain /density.
if (Gain(g_{out})<Gain(g_{in})) Delete(g_{out}); //delete the //cached object with minimal gain density
else return;
}
Insert(g_{in}); //put in the new cache object
}

For the media object flowing through the proxy server, the system first computes its caching gain density, and caches those with bigger gain density at a higher priority. If the cache storage space is full, the system will compare the gain density of the new media object with the smallest gain density in the cache storage. If the new media object has a larger density, the system will delete the media objects in the cache with smallest gain density until enough free space is available for the new media object. In order to ensure the integrity of the media object, the cache replacement policy must take into account the dependence among media objects. Examples are that the media redundant object is dependent of the media data object, and that higher layer media objects are dependent of lower layer ones.

2.4 The Cache Replacement Policy In The Case Of Multiple Servers

In the case of multiple servers, the copying of multimedia program makes the above cache replacement policy not applicable any more. This section revises the policy to cope with the new situation. In the case of a single server, the network usage sub gain, $Rev_U(v)$, is linear with the link bandwidth usage. While in the case of multiple servers, since the multimedia program is copied to multiple servers, the network usage sub gain is related with multiple links, and the network usage sub gain is

$$Rev_U(v) = BW(v) \times \varphi(v,t)$$

where $\varphi(v,t)$ is the overall network usage rate of program v at time of t. It is computed from the network usage rate and bandwidth condition of each link.F

$$\phi(v,t) = \frac{\sum_{if\ v\ in\ S_j} [\overline{BW_A(j,t)} \times \mu(j,t)]}{\sum_{if\ v\ in\ S_j} \overline{BW_A(j,t)}}$$

where $\overline{(BW_A(j,t)}$ *is the average valid bandwidth of the link connecting video server* S_j *at the time of t, and* $\mu(j,t)$ *is the network usage rate of this link.*

If a multimedia program does not have many copies, its network usage rate sub gain will be high, leading to a higher overall caching gain, and subsequently a high probability to be cached. For a multimedia program with many copies, the probability that it is cached in a proxy server is low since the stream is available on many servers. Of course, in the case that the usage on the links from these servers to the proxy server are all high, the overall caching gain of this program will be high, giving rise to a high chance of caching.

3. SIMULATION EXPERIMENT

In order to evaluate the service of multimedia proxy server, we designed an event-driven simulation platform. In this platform, 2000 multimedia programs are stored in 20 video servers. All the multimedia programs are PFGS MPEG-4 encoded.

A MEPG-4 stream has the basic layer and the extension layer. An extension layer can be broken anytime to be adaptive to the varying network bandwidth. To simplify the cache management, the extension layer is divided into three sub layers, and thus a four-layer encoding model is used. To improve the caching flexibility, we take the fine granularity media object as the basic cache management unit. The media object is 10s long, and each media object belongs to one of the four layers.

With the consideration of network isomerism, we limit the valid bandwidth on the link form the video server to the proxy server between 1.6Mbps and 64Mbps, and RTT between 5ms and 10ms. We also let the valid bandwidth vary between 50% and 150% of the average. A period of 24 hours is used to simulate the real life bandwidth's varying in daytime and night.

We assume that there are two types of Internet accessing methods, wireless access and cable access, and the Internet user of each type takes 50% share. Let the average wireless connection bandwidth be 800Kbps, its average bit error rate is 0.7%, and the average bandwidth on the connection of the Internet user to the proxy server be 2Mbps. Other parameters are listed in Table 1 below. The above parameters can be adjusted based on the experiment condition.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The simulation system is driven by user's requests. We assume that the user request comes as a poisson process with λ=0.4/s, and that the program request comes as a Zipf process with slope of 0.271. Let n be the number of programs, and vi be the ith most popular program. Then the demanding rate of this program is

$$\lambda_i = (1/i^{1-\theta})\left[\sum_{j=1}^{n}(1/j^{1-\theta})\right]$$

where θ is the slope of Zipf distribution.

The simulation system first runs for 72 hours for start up purpose, then it runs for another 72 hours, during which the various service performance indexes will be recorded.

Table 1 Parameters In Simulation

Parameter	Default	Bound
Number of media	2000	N/A
Media length	1800	N/A
Media rate/Kbps	1734	N/A
Server-ProxyBandwidth	N/A	1.5～64
Server-ProxyRTT/ms	N/A	5～10
RequestFrequency/(1/s)	0.4	N/A
Skew factor	0.271	N/A
WirelessChannel	800	N/A
Internet client bandwidth	1750	N/A
Wireless BER/%	0.7	N/A
Wireless proportion	50	30～80
Proxy cache size/Gb	400	150～750
Prefix length/s	30	N/A

4. EXPERIMENT RESULTS

We tested the performance of the gain-based cache replacement policy, and compare it with LFU, LRU, LRU-2 in the following aspects.

Figure 1 below compares these policies in terms of byte hit rate. As shown in Figure 1, the byte hit rate under each cache replacement policy increases as the cache storage grows. This is because larger cache space contains more programs. The gain-based cache replacement policy outperforms other policies with at least 30% higher byte hit rate. The gain-based policy manages fine granularity media object and uses more material performance indexes. This improves the cache's adaptability to system gain and leads to a higher byte hit rate.

Figure 2 below compares the media quality provided to cable access Internet users under different cache replacement policies. As shown in Figure 2, the media quality increases as the cache storage grows regardless of the employed cache replacement policy, however, the gain-based policy gives the best performance. The gain-based policy takes into account the dependency among media date layers and introduces the network usage rate sub gain. Thus the proxy server can work effectively even with limited network resource.

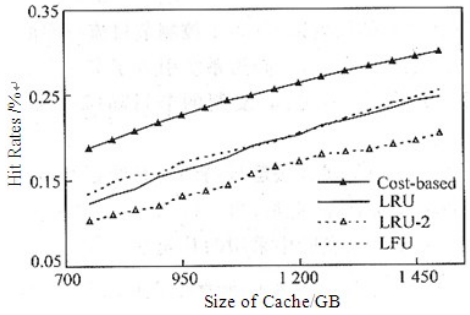

Figure 1 Hit Rates Of Cache Replacement Policies

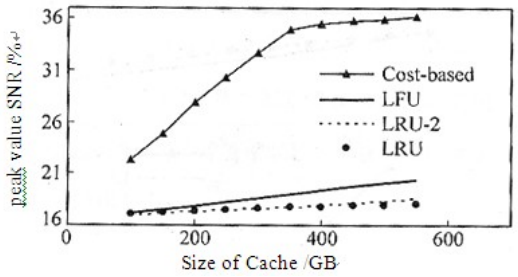

Figure 2 Media Quality Experienced By Cable Internet Users

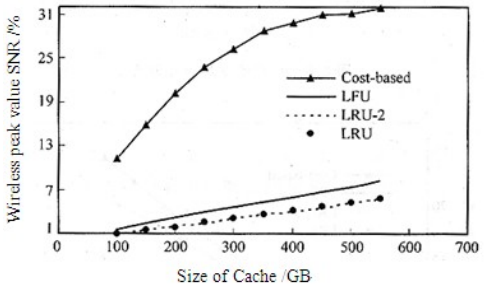

Figure 3 Media Quality Experienced By Wireless Internet Users

Figure 3 below compares the media quality provided to wireless access Internet users under different cache replacement policies. Due to the lower bandwidth (<800Kbps) and higher bit error rate (>0.7%), the wireless channel provides worse media quality than the cable channel. As shown in Figure 3, when the cache storage is 500GB, the media quality under the gain-based policy is about 30db, while the other policies provide only about

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

10db media quality. This shows that the computation of cache gain helps the unreliable wireless channel to provide the complete media stream.

Figure 4 below compares the start-up delay under different cache replacement policies. Since the gain-based policy introduces the start-up delay sub gain, more program prefixes are cached, and thus the average start-up delay is reduced remarkably. As shown in Figure 4, the average start-up delay under the gain-based policy is at least 50% lower than those under other policies.

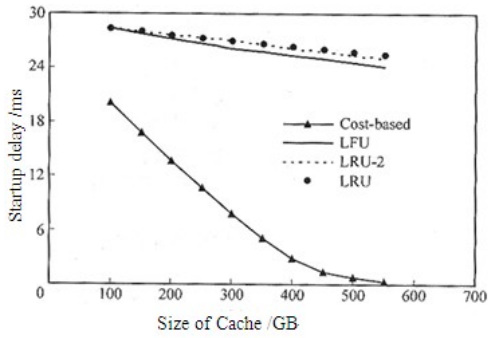

Figure 4 The Average Of Start Up Delay

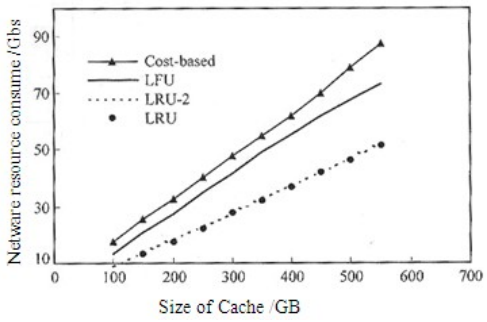

Figure 5 The Average Of Network Cost

Figure 5 below compares the reduction of the network cost under different cache replacement policies. As shown in Figure 5, all the policies can reduce the network cost effectively, and the reduction is nearly linear with the cache storage size. Since the gain- based policy takes the network cost explicitly as an optimization object, it outperforms others. For example, when the cache storage is about 500GB, the gain-based policy saves at least 25% more network resource than other policies.

5. CONCLUSION

Based on the research above, we can safely reach the conclusion, firstly, proxy server cache can be applied to improve the quality of receiving media stream by users. It's data generates media stream directly, which can avoid the influence on QoS caused by network congestion and shake. Secondly, Proxy server cache can reduce start delay of media stream effectively. Proxy Server is on the verge of the network, therefore, the delay from proxy server to user's side is insignificant. Last but not least, proxy server cache remarkably decreases transmission medium's consumption on network. Generating media stream directly by utilizing cache saves the network transmission consumption from video server to proxy server. It must be pointed out that cache replacement policy must consider the dependency relationship of media objects in order to guarantee their integrity and effectiveness in cache.

ACKNOWLEDGEMENTS:

This work was supported by a grant from the National Natural Science Foundation of Zhejiang (No. Y1100314).

REFRENCES:

[1] Lixin Gao, Zhi-Li Zhang, Don Towsley. Catching and selective catching: Efficient latency techniques for delivering continuous multimedia streams. *Proc ACM Multimedia'99*, 1999

[2] Wooster R, Abrams M. Proxy caching the estimates page load delays. *6th Int'l World Wide Web Conference*, April 7~11, 1997, Santa Clara, CA. http://www6.nttlabs.com/HyperNews/get/PAPER250.html

[3] Li Fan, Pei Cao, Jussara Almeida, Andreri Z Broder. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Trans Networking*, 2000,8(3)

[4] Tewari R, Vin H M, Dan A, Sitaram D. Resource-based caching for web servers. *ACM Multimedia Systems Journal*, 2000

[5] Yu F, Zhang Q, Zhu W, Zhang Y Q. QoS-adaptive proxy caching for multimedia streaming over the Internet. *Proc First IEEE Pacific-Rim Conf Multimedia*, Sydney, Australia, Dec 13~15,2000

[6] Zhi-Li Zhang, Yuewei Wang, David H C Du, Dongli Su. Video staging: A proxy-server-based approach to end-to-end video delivery over wide-area network. *IEEE/ACM Trans on Networking*, 2000

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[7] Lorenzetti P, Rizzo Vicisano L. Replacement Policies for a Proxy Cache. *http://www.iet.Unipi. It/luigi/research.html*

[8] Lance Spitzner. Honeypots: Definitions and Value of Honeypots. *http://www.trackinghackers.com/papers/ honeypots.htrni*.2003,5