A COMPARISON OF CONTINUOUS MEDIA STREAM PROXY CACHING POLICIES

Fujian Liu
Department of Computer Science
University of Saskatchewan
Saskatoon, SK S7N 5A9
Canada
Email: ful113@mail.usask.ca

Dwight Makaroff
Department of Computer Science
University of Saskatchewan
Saskatoon, SK S7N 5A9
Canada
Email: makaroff@cs.usask.ca

ABSTRACT
As the computing capacity and network bandwidth at clients increase rapidly, the bottleneck for distributing continuous media streams is pushed upstream to the network backbone and the media source server. Compared with traditional text or image files, streaming media files need very large storage space, large bandwidth to be transferred, and a time-constrained service to ensure the play back quality.

To reduce server workload and network bandwidth consumed by media streams, continuous media stream proxy was introduced. Different proxy caching policies were also proposed to maximize the utilization of the limited proxy space and the network bandwidth saving. In this paper, using bandwidth saving as a metric, we compare three media proxy caching policies: Smallest Caching Utility (SCU), Least Frequently Used (LFU), and Least Recently Used (LRU). Our comparisons investigate the impact of proxy size, time interval, and various workload characteristics on these three policies. Our simulation results show that time interval is a significant parameter for SCU, and that the policies behave differently under various workload parameters (e.g. shape parameter α of *Zipf-like* distribution popularity).

KEYWORDS
Continuous Media, Proxy Caching, LFU, SCU

1 Introduction

Continuous streaming media objects, particularly video files, have become more and more popular on the Internet. Unlike traditional text or image files, media files are very large, ranging from hundreds of megabytes to gigabytes. In addition, a time sensitive service is required for streaming media objects to avoid client jitter.

As the computing capacity of personal computer and network bandwidth increase rapidly, the bottleneck for distributing continuous media streams is pushed upstream towards the media source server [10]. Deploying a proxy to cache the popular objects in the context of distributing continuous streaming media is one technique to reduce network bandwidth consumption and server workload. The logic for using a proxy is that requests for objects that are cached at the proxy, will not require resources at the server and the network between the proxy and the server. Another advantage of deploying proxy is to decrease the initial latency of response.

Caching is only effective if locality of reference is present in the client request patterns. A recent study [5] has attempted to isolate two aspects of locality: correlation and popularity. Correlation considers that references to the same object occur close together in time, while popularity merelet refers to the fact that some objects have much more requests than others, though they could be evenly spread out over the length of the observation period. Our study chooses three algorithms which take divergent views on which aspect of locality is more significant. The Least Frequently Used (LFU) approach considers popularity only, while Smallest Caching Utility (SCU) and Least Recently Used (LRU) make some attempt at considering both aspects of locality. As the time interval considered by SCU increases, it puts more weight on the popularity component of locality. In this paper, we use bandwidth saving as a metric to compare these continuous media stream proxy caching policies under various workload conditions.

Our comparison investigates the impact of proxy size, time interval, and various workload characteristics on these policies. Our simulation results show that the time interval Δt is a significant parameter for SCU. The performance difference among LFU, SCU, and LRU decreases when the shape parameter α for *Zipf-like* distribution increases. When the media traffic is self-similar, the performance difference between LFU and SCU is larger than other traffic modes. The performance difference between LFU, SCU, and LRU

increases when the number of sessions goes up, and also when the number of objects decreases.

The paper is organized as follows. Section 2 presents the related work to our research. Section 3 presents our simulation background and methodology. Our simulation results are presented in Section 4; and we give our conclusions in Section 5.

2 Related Work

Compared with the extensive study of the characteristics of the requests to the traditional web objects, there are just a few published papers that study the characteristics of media workloads. Chesire et al. [4] analyzed a client-based streaming media workload generated by a large educational organization. Based on their trace-driven simulation, the authors argued that network bandwidth saving can benefit from proxy caching although less than a traditional web workload. Almeida et al. [2] studied the log files of two media servers at two major United States universities and characterized the workloads in terms of session arrival rate, session inter-arrival time distributions, and distribution of file access. Cherkasova et al. [3] analyzed the log files of two enterprise media servers. They found that most of the incomplete sessions were accessing the initial parts of media files; the distribution of clients accesses to media files can be approximated by *Zipf-like* distribution. They also proposed two new metrics, new file impact and life span, to characterize the dynamics and evolution of the workloads.

Various continuous streaming media proxy caching policies have been proposed. Resource Based Caching (RBC) [9] is a web proxy caching algorithm that handles different data types, including continuous and non-continuous objects. By characterizing the resource requirement and caching gain of an object, the RBC scheme dynamically selects the object granularity to maximize the utilization of the limited cache resource, either bandwidth or space, and replaces the cached objects based on their updated cache resource usage and caching gain. Almeida et al. [1] proposed a hybrid caching strategy called LFU/IC (Interval Caching), for streaming media files. This strategy deploys IC caching either in memory or on disk and LFU for disk caching. Smallest Caching Utility scheme presented by Lim et al. [7], replaces the last segment of a cached media object with the smallest caching utility, which is the ratio of caching bandwidth benefit to caching space cost.

Our simulation comparison differs from the above research in terms of the research scope that it focuses on. In this paper, we investigate the impact of media proxy cache size, compared with origin server, and various parameterized workloads on the continuous media proxy caching policies LFU, SCU, and LRU. To our knowledge, there is no such detailed research work reported, although this topic is a big concern in the field of continuous stream media proxy [1] [6]. This paper tries to explore and identify the possible workload parameters that have a significant impact on these continuous stream media proxy caching schemes. As well we use these significant parameters to compare the performance of these schemes in terms of backbone network bandwidth saving.

3 Simulation Methodology

We use a synthetic workload for our experiments. This is because we want to study the impact of various workload parameters on the performance differences among SCU, LRU, and LFU. A real trace or log file is not generic because it is unable to provide a controllable, large range of workload characteristic parameters.

Generator of Internet Streaming Media Objects and workloads (GISMO) was proposed by S. Jin et al. [6]. GISMO can generate continuous streaming media workloads that are characterized to match properties of real workloads in terms of object popularity, temporal correlation of requests, seasonal access patterns, user session durations, user interactivity, and VBR long-range dependence and marginal distribution.

We use the output of GISMO [8] as the objects , stored at the origin media server and the workload requests generated by clients behind the proxy. Our simulator, running as the proxy, responds to the clients' requests by caching objects according to the replacement policy. LFU, LRU, and SCU algorithms are implemented in our simulator to decide which object should be cached, and which object or segment should be evicted if the cache is full and a new caching requirement arrives. The bytes corresponding to the portion of the file cached are accumulated as bandwidth saving if they are directly served from proxy. The percentage of bandwidth saving is the ratio of bandwidth saving to the initial bandwidth requirement without proxy. The simulator outputs the percentages of backbone network bandwidth saving for LFU, LRU, and SCU under the current workload.

GISMO [8] reads in two parameters: the number of objects to create and the number of sessions to create, and outputs the objects and access requests to these objects. In GISMO, the skewed popularity of streaming media objects is characterized by a *Zipf-like* distribution. The parameter α determines the level of popularity skewness for the objects. GISMO uses the *Pareto* distribution to model correlated inter-arrival time. The media object size is modeled by *Lognormal* distribution. We use CBR objects to model our continuous media objects, which are stored at the origin server side and will be partially or totally cached

at the proxy when clients request the objects. The default system parameters are shown in Table 1.

Table 1: Workload Parameters

Characteristic	Distribution	Default value
Object popularity	Zipf-like	$\alpha = 0.73$
Temporal correlation	Pareto	$\alpha = 1.0$
		$k = 0.001$
Partial access	Pareto	$\alpha = 1.0$
		$k = 0.001$
Object size	Lognomal	$\mu = 10$
		$\sigma = 0.5$

The network architecture for our simulation includes single media server and single media proxy, which located near the client population. The simulator is implemented in C.

3.1 Metrics

There are at least two possible measures of performance that are of interest: latency and bandwidth usage. If only a prefix is delivered from the cache, while the remainder is serviced directly from the server, there is no startup latency introduced, though the client needs additional buffer and bandwidth to be able to receive the media object. If the remainder portion of an object must be retrieved into the proxy cache, it may be the case that the proxy does not have the bandwidth to both receive from the server and stream to its clients. We do not consider this aspect of performance. We concentrate on the savings in bandwidth that result from caching entire video object, or at least some prefix of the object. Bandwidth savings is defined as:

Requested bytes from cache/total request size.

For every set of the experimental results we used in this paper, we ran the simulator based on 15 different randomly generated workloads and used average values. The different workloads generated very similar results.

3.2 Caching Algorithms and Parameter Settings

The LRU policy supposes that the most recent requested object will be most likely accessed again in the future. LRU will replace the object that has not been requested for the longest time when a cache replacement is necessary.

The LFU scheme keeps records of the access frequencies for every cached media object, and updates the records when a new session arrives. An object with the least access frequency will be kicked out when a replacement is necessary.

To maximize the LRU and LFU cache utilization in our simulation, when the size of target cached object is larger than that of as needed, we evict part of the target cached object, not the entire cached object with the least access frequency.

The related parameters mentioned for SCU by Lim et al. [7] are:

- s: segment size
- L_i: length of stream i, which is requested by client
- C_i: the size of the cached part of stream i
- P_i: the total amount of data played back by clients for stream i during the past recent time interval Δt
- $CU_i(C_i)$: caching utility of stream i for cached data C_i

For every cached media object, SCU keeps records of C_i and the start & end timestamps of every live stream during the recent past time interval Δt for C_i. When a segment replacement is necessary, SCU calculates the CU_i:

$$CU_i = \frac{\sum_{j=1}^{k} P_{i,j}}{C_i} \quad (1)$$

where k is the number of total live streams for cached object i during time interval Δt, $P_{i,j}$ is the amount of data played back by client j for object i during time interval Δt for every cached media object. The simplified abstract SCU algorithm is:

```
while (a client requests "current" object)
{
 using time interval delta_t, update P[i] for every cached object;
 if (one segment s of current object requires cache space) {
   if (free space is not enough) {
     calculate CU[i] for every cached object;
     victim = the object with the least CU of all updated CU[i];
     if (victim object == the object that s belongs to)
     /*the victim object has the same object id with current object*/
         return;
     if (CU[victim](C[victim] -s) > CU[current](C[current] + s))
     /*replacing s of victim object can't get better caching utility*/
         return;
     replace the last s of the victim object with the current s;
     update C[current] and C[victim];
   }
   else allocate free space for this segment s;
 }
}
```

While implementing SCU for our simulator, we found that the authors of [7] omitted one significant system parameter for SCU policy interval time Δt, which could have a very close relationship with the performance of SCU, since it determines the value of P_i for a cached object. The experimental results in section 4 confirm our guess.

During our first stage of experiments, we set the SCU parameter s=10MB to be the same as the original setting presented by Lim et al. [7].

We deployed prefix caching to cache the initial part of the object. We assume that all the media objects have the same encoding bit rate, and that the media objects are transferred to clients and played back at a constant speed, proportional to the encoding speed of media files.

For our simulation, we change various parameters to examine their impact on the performance of LFU and SCU, and on the performance difference between them. We test the impact of one parameter while leaving the other parameters unchanged. For our experiments, we generated a workload with the duration of one day.

4 Experimental Results

4.1 SCU Time Interval

First, to investigate the impact of time interval on SCU performance saving, we use GISMO default parameters to generate 200 objects and 20,000 sessions as the workload. The total size of these 200 objects, the origin server size, is *210,016* megabytes, and the initial total backbone network bandwidth cost for this workload is *3,756,236* megabytes. For our later experiments, we do not provide the information about origin server size and initial bandwidth cost again, since we use the ratio of size (*proxy_size/server_size*) and the ratio of bandwidth saving as our metric. The value of popularity parameter α is *0.73*, which is the default value of α in GISMO [8] for testing the impact of time interval.

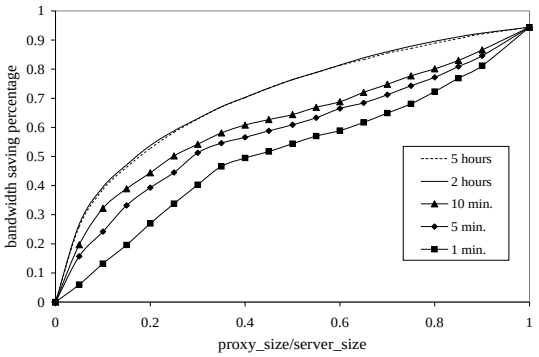

Figure 1: Impact of SCU Time Interval

Figure 1 shows the impact of the parameter time interval on SCU performance. The *x-axis* is the ratio of proxy size to origin server size, and the *y-axis* is the percentage of SCU bandwidth saving.

Figure 1 presents the SCU performance for the time interval, Δt, of 1 minute, 5 minutes, 10 minutes, 2 hours, and 5 hours. When proxy size is zero, there is no bandwidth saving for SCU proxy scheme, since SCU cannot cache any objects. The bandwidth saving percentage can reach 94% when the proxy size is equal to server size, which means SCU proxy scheme can be very effective at reducing the backbone network media traffic if the proxy size is large enough. It cannot reach 100% because the first request for every object cannot be served by the proxy that is initially empty. Figure 1 shows that the SCU performance increases gradually when the time interval becomes longer until an interval of 2 hours. This is because by increasing P_i, the longer time interval helps SCU to make a wiser decision regarding segment replacement. For all the comparisons in our simulation, we use the performance of SCU with the optimal time interval, not a constant value, to compare with that of LRU and LFU.

4.2 Pouplarity Parameter α

Our experiments show that the shape parameter α for *Zipf-like* distribution (i.e., the popularity parameter), is a quite influential parameter for the bandwidth saving ratio. Figure 2 shows the impact of popularity parameter α, when α=0.5, 0.73, and 1.5. We chose these values basing on the workload characterization reported in literature. The parameter α ranges from 0.2 to 2.0, reported by Almeida et al. [2] [3]. The optimal time intervals of SCU policy for α=0.5, α=0.73, α=1.5 are 5 hours, 2 hours, and 5 hours respectively.

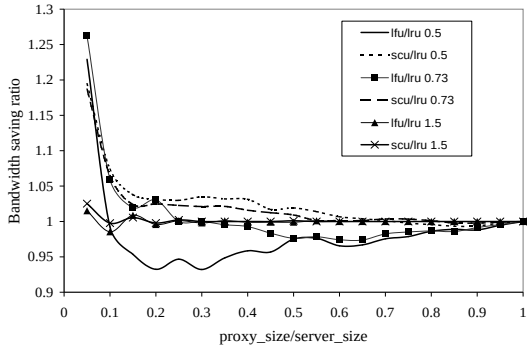

Figure 2: Impact of Popularity Parameter α, Δt=2h

The *y-axis* on Figure 1 shows the difference in bandwidth savings as a ratio. The standard LRU cache replacement algorithm is used as the denomiator for this ratio. When the value is greater thatn 1, the comparative algorithm has a higher bandwidth saving than LRU.

From Figure 2 we can see that when α is small, LRU outperforms LFU for all cache sizes. As α increases, the difference between LRU and LFU decreases to the point where there is no difference for α=1.5. In general, SCU reduces the bandwidth more for all values of α, but the performance becomes the same for large cache size and high value of α. As the

popularity is skewed, the influence of corelation decreases and LFU becomes a more useful algorthm to consider. With small cache and low skew, LFU is a bad choice. LFU benefits much from popularity skewness since its replacement decision is just based on access frequencies.

4.3 Diurnal Parameter

In our experiments, we adjusted GISMO to generate three typical byte transfer modes to test the performance difference of the caching algorithms. Figure 3 shows the impact of diurnal parameter.

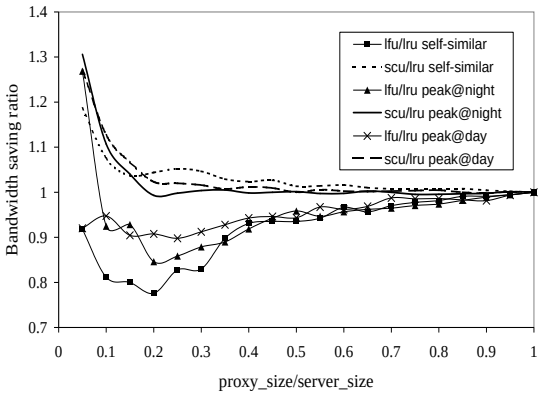

Figure 3: Impact of Diurnal Parameters

We use self-similar, peak@night, and peak@day to model the typical media network traffic processes. The optimal time intervals of SCU scheme for these three modes self-similar, peak@night, and peak@day are 5 hours, 5 hours, and 2 hours. Figure 3 shows that SCU scheme gets a fairly good bandwidth saving from self-similar traffic when the cache size is small. This is because SCU can take good advantage of the reasonable time period of traffic similarity to accumulate the P_is for the cached objects to keep the objects in cache for later similar accesses, not being evicted. Our test experiments confirmed this – SCU got the best performance when time interval Δt is 5 hours, which is a little longer than the time period of minimal traffic self-similarity, about 4 hours. In Figure 3 SCU gets a good performance for peak@day, $proxy\_size/server\_size$<0.15, since SCU can serve a long time of peak traffic well based on its ability of balancing cache space for all objects, which is the advantage of dividing P_i by C_i to calculate CU.

4.4 Session Number and Object Number

Figure 4 shows the impact of session number. The optimal time intervals of SCU scheme for $session\_number$=$10K$ and $session\_number$=$100K$

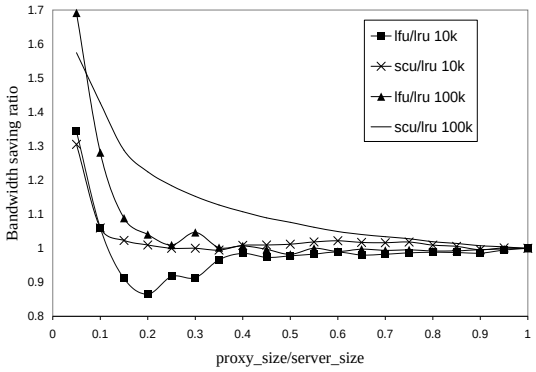

Figure 4: Impact of Session Number

are the same: 2 hours. Compared with the performance of SCU with $session\_number$=$10K$, SCU with $session\_number$=$100K$ gets a higher performance. More sessions means more traffic during time interval Δt, and that SCU can get more bandwidth saving information P_i during time interval Δt. Again, LFU performes worse than both SCU and LRU in all cases.

The impact of object number is shown in Figure 5. The optimal time intervals of SCU policy for $object\_number$=50, and $object\_number$=100 are 2 hours, and 5 hours respectively. In Figure 5, the difference between SCU and LFU are greater with smaller number of objects. Decreasing object number will result in more requests in a unit time period, while the object popularity skewness does not change. Thus SCU can get more past traffic information about the cached objects in terms of bandwidth saving contribution, while LFU cannot benefit due to the unchanging order of popularity.

In Figure 3, Figure 4, and Figure 5 we can see that SCU/LFU fluctuates dramatically when $proxy\_size/server\_size$ is less than 0.15. We can not explain this exactly. A possible reason is that the performances of LFU and SCU are not stable and that their performance change inversely the when the proxy size is too small compared with the server size.

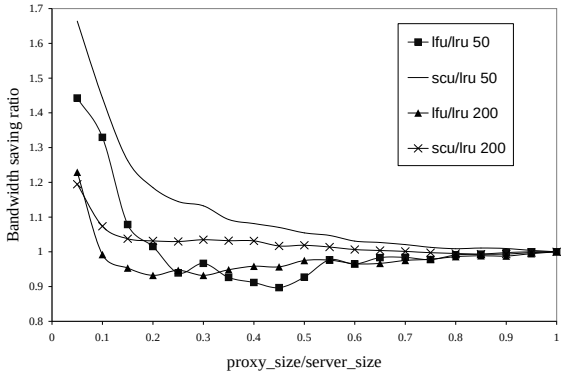

Figure 5: Impact of Object Number

4.5 Remaining Parameters

Due to some users partial access characteristic, i.e., behavior of just accessing initial part of media objects, all the algorithms can cache the initial parts of all the objects when proxy is big enough. At that time, the algorithms get similar bandwidth saving. Figure 2, Figure 3, Figure 4, and Figure 5 show that the performance difference between SCU and LFU is small when $proxy\_size/server\_size$ is greater than 0.5.

Our experiments also show that object size, partial access characteristic, temporal correlation, and segment size do not have an obvious impact on the performance difference between LFU and SCU. The experimental figures for the impact of these parameters are not shown due to space limitation.

5 Conclusion and Future Work

Our work is to investigate the performance of three continuous media proxy caching policies, SCU, LFU, and LRU, in various workload scenarios. Our conclusions include:

- Time interval Δt is a significant parameter for SCU. SCU with a self-tuning mechanism to adjust time interval Δt is a possible successor for current SCU scheme.
- The performance difference among LFU, SCU, and LRU is reduced when shape parameter α for *Zipf-like* distribution increases.
- When the media traffic is self-similar, the performance difference between LFU and SCU is larger than other traffic modes.
- The performance difference between LFU and SCU increases when the session number goes up, while as the object number decreases, the performance difference between LFU and SCU increases.

Our future work is to extend our proxy to a broad network environment, which includes server/proxy side multicast, patching/merging techniques, VBR traffic, and workload with more user interactivity.

6 Acknowledgments

The authors would like to thank Shudong Jin for providing the workload generator and Yanping Zhao for her helpful suggestions.

References

[1] J. M. Almeida, D. L. Eager, and M. K. Vernon. A Hybrid Caching Strategy for Streaming Media Files. In *Proc. of IS&T/SPIE Conf. on Multimedia Computing and Networking (MMCN 2001)*, pages 200–212, San Jose, California, January 2001.

[2] J. M. Almeida, J. Krueger D. L. Eager, and M. K. Vernon. Analysis of Educational Media Server Workloads. In *Proc. of 11th Int'l. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2001)*, pages 21–30, Port Jefferson, NY, June 2001.

[3] L. Cherkasova and M. Gupta. Characterizing Locality, Evolution, and Life Span of Accesses in Enterprise Media Server Workloads. In *Proc. of 12th Int'l. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2002)*, pages 33–42, Miami, Florida, USA, May 2002.

[4] M. Chesire, A. Wolman, G. M. Voelker, and H. M. Levy. Measurement and Analysis of a Streaming Media Workload. In *Proc. of the USENIX Symposium on Internet Technologies and Systems (USITS'01)*, San Francisco, CA, USA, March 2001.

[5] R. Fonseca, V. Almeida, M. Crovella, and B. Abrahao. On the Intrinsic Locality Properties of Web Reference Streams. In *IEEE INFOCOM 2003*, San Francisco, CA, USA, March 2003.

[6] S. Jin and A. Bestavros. GISMO: A Generator of Internet Streaming Media Objects and workloads. *ACM SIGMETRICS Performance Evaluation Review*, 29(3):2–10, December 2001.

[7] E. Lim, S. Park, H. Hong, and K. Chung. A Proxy Caching Scheme for Continuous Media Streams on the Internet. In *Proc. Of 15th International Conference on Information Networking*, pages 720–725, Beppu City, Oita, Japan, January 2001.

[8] Shudong Jin. GISMO: Generator of Internet Streaming Media Objects and workloads. http://csr.bu.edu/gismo/. Web site.

[9] R. Tewari, H. M. Vin, A. Dan, and D. Sitaram. Resource-based Caching for Web Servers. In *Proc. Of IS&T/SPIE Conference on Multimedia Computing and Networking (MMCN'98)*, San Jose, California, January 1998.

[10] C. Venkatramani, O. Verscheure, P. Frossard, and K. W. Lee. Optimal Proxy Management for Multimedia Streaming in Content Distribution Networks. In *Proc. of 12th Int'l. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2002)*, pages 147–152, Miami, Florida, USA, May 2002.