Buffer Sharing for Proxy Caching of Streaming Sessions *

Songqing Chen
Dept. of Computer Science
College of William and Mary
Williamsburg, VA 23185
sqchen@cs.wm.edu

Bo Shen
Mobile and Media System Lab
Hewlett-Packard Laboratory
Palo Alto, CA 94304
boshen@hpl.hp.com

Yong Yan
Mobile and Media System Lab
Hewlett-Packard Laboratory
Palo Alto, CA 94304
yongyan@hpl.hp.com

Xiaodong Zhang
Dept. of Computer Science
College of William and Mary
Williamsburg, VA 23185
zhang@cs.wm.edu

ABSTRACT

With the falling price of the memory, an increasing number of multimedia servers and proxies are now equipped with a large memory space. Caching media objects in the memory of a proxy helps to reduce the network traffic, the disk I/O bandwidth requirement, and the data delivery latency. The *running buffer* approach and its alternatives are representative techniques to cache streaming data in the memory. There are two limits in the existing techniques. First, although multiple running buffers for the same media object co-exist in a given processing period, data sharing among the multiple buffers is not considered. Second, user access patterns are not insightfully considered in the buffer management. In this study, we propose two techniques based on *shared running buffers (SRB)* in the proxy to address these limits. Considering user access patterns and characteristics of the requested media objects, our techniques adaptively allocate memory buffers to fully utilize the currently buffered data of streaming sessions to serve existing requests concurrently, with the aim to reduce both the server load and the network traffic. Experimentally comparing with several existing techniques, we have shown that the proposed techniques have achieved significant performance improvement by effectively using the sharing running buffers.

Key Words: Shared Running Buffer (SRB), Proxy Caching, Patching, Streaming Media, VOD.

1. INTRODUCTION

The delivery of the streaming media content presents several challenges: (1) The size of a streaming media object is usually several orders of magnitudes larger than that of the traditional text-based Web content. (2) The demand of the streaming media object on the continuous and timely delivery is more rigorous than that of the text-based Web pages.

To address these challenges, researchers have proposed different methods, such as partial caching, to cache streaming media objects. More recently, researchers have paid attention to the temporal caching of media objects in the proxy memory. The fixed-size running buffer caching [1] and the interval caching [2] are two major running (memory) buffer based caching techniques. The basic working principle of the running buffer based caching technique is as follows: when a request arrives, a fixed-size buffer of a predetermined length is allocated to cache the media data fetched by the proxy, hoping that closely followed requests could reuse the data in the memory instead of obtaining it from other sources (the disk, the origin server or other caches). In contrast, the interval caching technique uses a different approach. It considers two requests immediately followed as a request pair, and incrementally orders the arrival intervals of all request pairs (the arrival interval of a request pair is defined as the difference in their arrival times). In the memory allocation, the interval caching gives preference to smaller intervals, expecting more requests can be served for a given amount of memory.

However, the running buffer caching does not take consideration of user access patterns, resulting in the inefficient usage of the memory resource. The interval caching approach does consider the user access pattern in the buffer allocation and it shares another limit with the running buffer caching: data sharing among different buffers has not been considered.

2. SHARED RUNNING BUFFERS (SRB)

It has been shown that the running buffer caching and the interval caching do not take effective use of the limited memory resource. The SRB algorithm is designed to overcome the above listed problems. In SRB, an initial memory buffer is allocated once a request comes for the object with the *scale* of this object length. When the buffer is full, it will decide whether to expand or shrink the buffer according to the user access pattern to that object, so that there will be no any waste of the memory. The SRB is thus adaptive to the object length and the user access behaviors. More importantly, the data cached in different running buffers for a same object are fully shared by any later requests concurrently, even if the buffer is not allocated for these requests. This is different from any of the previous work. When requests are prematurely terminated, the SRB algorithm efficiently reclaims the idle memory space and does near-optimal buffer replacement at runtime.

Motivated by the patching approaches that were heavily studied in the VOD environment, we further propose another enhanced media delivering algorithm: Patching SRB (PSRB), which further

*The work was originated and largely completed while Songqing Chen worked at HP Labs during the summer 2002.

WWW2003, May 20–24, 2003, Budapest, Hungary.
ACM 1-58113-680-3/03/0005.

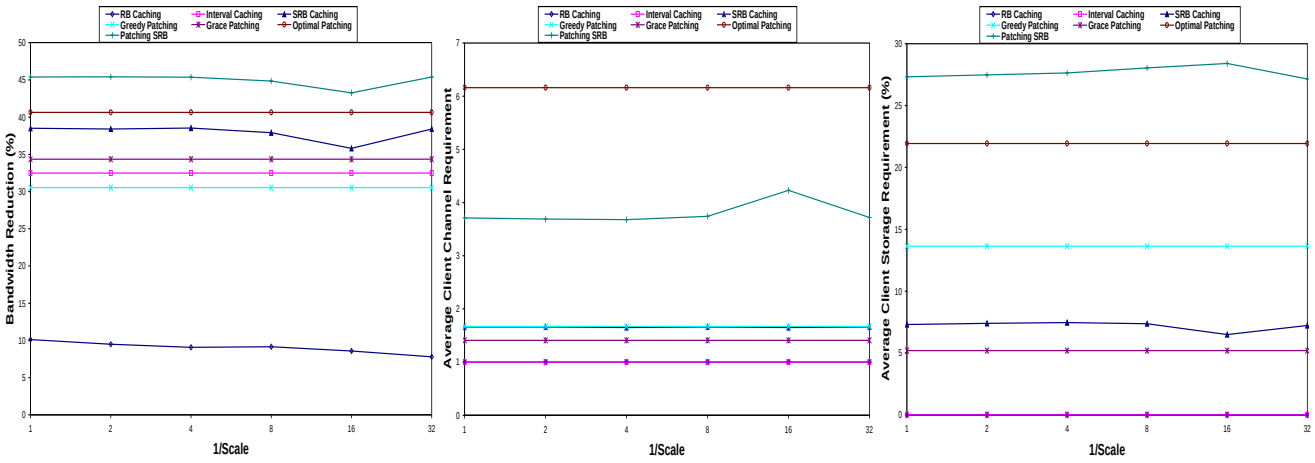

Figure 1: WEB: (a) Server Traffic Reduction, (b) Average Client Channel and (c) Storage Requirement(%) with 1GB Memory

improves the performance of the media data delivery by making use of the idea of patching with additional storage requirement on the client side.

3. PERFORMANCE EVALUATION

Synthetic and real workloads are used to evaluate the performance of the alogirhtms. WEB simulates accesses to media objects in the Web environment, while REAL comes from logs of HP Corporate Media Solutions, covering the period from April 1 through April 10, 2001. The effectiveness of the algorithms is studied by simulating different *scale* factors for the allocation of the initial buffer size. The *server traffic reduction* (shown as "bandwidth reduction" in the figures) is used to evaluate the performance of the proposed caching algorithms. We measure the traffic between the proxy and the client in terms of the *average client channel requirement*. The *average client storage requirement* in percentage of the full size of the media object is used to indicate the storage requirement on the client side.

For each simulation, we compare a set of seven algorithms in three groups. The first group contains buffering schemes which include the running buffer caching and the interval caching. The second group contains patching algorithms, specifically the greedy patching, the grace patching and the optimal patching. The third group contains the two shared running buffer algorithms proposed in this study.

Figure 1(a) shows the server traffic reduction achieved by each algorithm on WEB. Notice that PSRB achieves the best reduction and SRB achieves the next best reduction after the optimal patching. RB caching achieves the least amount of reduction. As expected, the performance of the three patching algorithms does not depend on the *scale* factor for allocating the initial buffer. Neither does that of the interval caching since the interval caching allocates buffers based on access intervals.

For the running buffer scheme, we observe some variations in the performance with respect to the *scale* factor. In general, the variations are limited. To a certain extent, the performance gain of the *bandwidth reduction* is a trade-off between the number of running buffers and the sizes of running buffers. A larger buffer implies that more requests can be served from the proxy buffer. However, a larger buffer also indicates that less memory space is left for other requests. This in turn leads to a larger number of server accesses since there is no available memory. On the other hand, a smaller buffer may serve a smaller number of requests but it leaves more memory space for the system to allocate for other requests.

Figure 1 (b) and (c) show the average channel and storage requirement on the client. Notice that the optimal patching achieves a better server traffic reduction by paying the penalty of imposing the biggest number of client channels required. Comparatively, PSRB and SRB require 30% to 60% fewer client channels while achieving a better or closer server traffic reduction ratio.

PSRB allows the session sharing even when memory space is not available. It is therefore expected that PSRB achieves the highest rate of server traffic rate reduction. In the mean time, it also requires the largest client side storage. On the other hand, SRB achieves about 6% less traffic reduction, but the requirement on client channel and storage is significantly lower. The performance results on REAL are omitted.

4. SUMMARY

In this study, we proposed the Shared Running Buffers (SRB) based algorithms for efficiently delivering streaming media objects. These algorithms dynamically cache media objects in the proxy memory during the delivery, adaptively allocate the memory buffer by exploiting the user access pattern, and enable the data being fully shared among different buffers. The simulation results show the efficiency of the proposed algorithms. Both algorithms require that the client be capable of listening to multiple channels at the same time. Comparing with previous solutions that also require multiple client channels, the proposed algorithms achieve a better server traffic reduction rate with less or similar load on the link between the proxy and the client.

Acknowledgment: Thanks to Sujoy Basu for providing proxy traces extracted from media server logs. Thanks to Susie Wee for the helpful discussions on the algorithm representation.

5. REFERENCES

[1] E. Bommaiah, K. Guo, M. Hofmann, and S. Paul. Design and implementation of a caching system for streaming media over the internet. In *Proceedings of IEEE Real Time Technology and Applications Symposium*, May 2000.

[2] A. Dan and D. Sitaram. Buffer management policy for an on-demand video server. In *IBM Research Report 19347*, Yorktown Heights, NY, 1993.