Popularity-aware interval caching for multimedia streaming servers

Taeseok Kim, Hyokyung Bahn and Kern Koh

An efficient buffer management scheme for multimedia streaming servers is presented. The proposed scheme exploits the reference popularity of multimedia objects as well as the time interval between two consecutive requests on an identical object. Through trace-driven simulations, it is shown that the proposed scheme improves the performance of multimedia servers significantly.

Introduction: Recently, multimedia streaming services such as Video on Demand (VOD) are rapidly becoming prevalent, and caching of multimedia objects in these environments is becoming increasingly important. Owing to the large volume of multimedia objects and the strictly sequential access pattern, traditional buffer cache management techniques will not work well for multimedia server systems. To address this problem, Dan proposed the *interval caching policy* that exploits the short-term temporal locality of accessing the same multimedia object consecutively [1]. By caching only the data in the interval between two successive streams on the same object, the following stream can be serviced directly from the buffer cache without I/O operations. Ozden *et al.* proposed the *distance caching policy* which is similar to interval caching [2]. However, these interval-based caching mechanisms exploit only the short-term temporal locality of two consecutive requests on an identical object and do not consider the popularity of an object. Consequently, when the size of a multimedia object is not sufficiently large or when the inter-arrival time of stream requests is too long, there is little opportunity to obtain the effectiveness of interval caching.

In this Letter, we propose the Popularity-aware Interval Caching (PIC) scheme for multimedia streaming servers. PIC resolves the aforementioned problems of interval caching by considering reference popularity as well as request intervals. We introduce the concept of virtual interval based on the past reference behaviour of multimedia objects to exploit the reference popularity. Trace-driven simulations with extensive VOD traces show that the PIC scheme performs better than the interval caching policy, the Least Recently Used (LRU) policy, and the Most Recently Used (MRU) policy in terms of cache miss ratio.

System model: Our multimedia server consists of an I/O manager, a buffer manager, and a network manager (Fig. 1). The buffer manager divides the memory buffer into the *cache* and the *read-ahead buffer*. The read-ahead buffer stores data to be sent immediately to clients while the cache stores data already sent to clients which can be reused when requests for the same object arrive. Note that data in the memory buffer do not actually move their physical positions (from the read-ahead buffer to the cache) but just a cache flag is used to indicate whether it is in the cache. For each stream request, when the requested block is not in the cache, the I/O manager acquires a free block, inserts it into the read-ahead buffer, and starts disk I/O. However, if the requested block is in the cache, the cached block is serviced directly without I/O operations. Finally, the network manager reads necessary data blocks from the memory buffer and sends them to the client through the network.

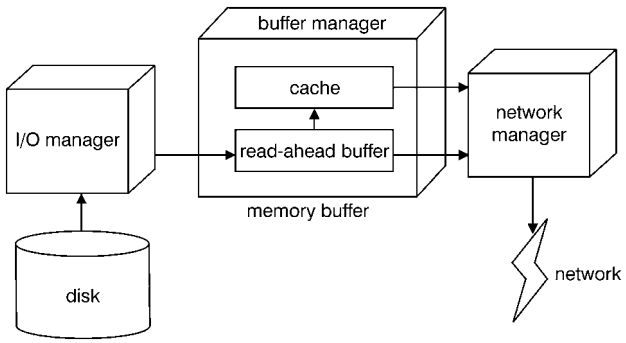

Fig. 1 *Multimedia streaming server architecture*

Popularity-aware interval caching scheme: For any two consecutive requests for the same object, the later stream can read the data brought into the memory buffer by the earlier stream if the data is retained in the cache until it is read by the later stream. Understanding such dependencies makes it possible to guarantee continuous delivery of the later stream with a small amount of cache space. Let an *interval* denote the distance of the offsets between two consecutive requests on an identical object. The interval caching policy aims at maximising the number of concurrent streams serviced from the memory buffer. Hence, with a given cache space, the interval caching policy orders the intervals in terms of space requirements and caches the shortest intervals (Fig. 2*a*).

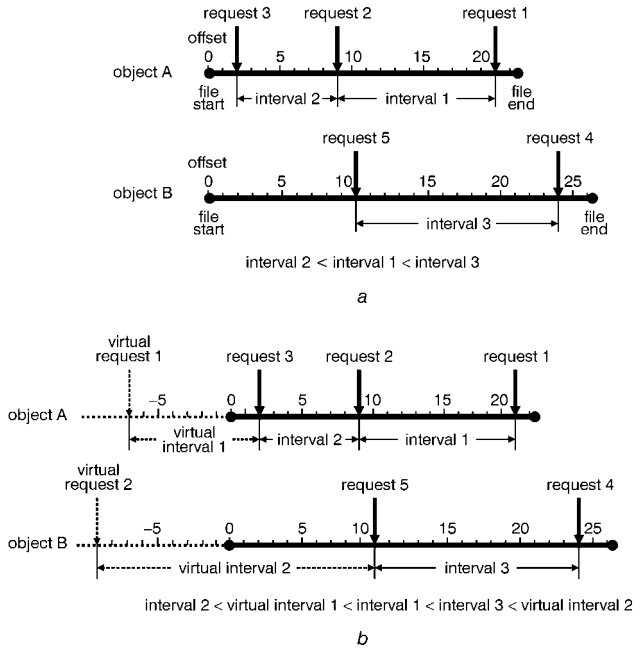

Fig. 2 *Interval caching and popularity-aware interval caching*
a When cache size is 20, interval caching selects intervals 2 and 1 for caching
b When cache size is 20, PIC selects interval 2 and virtual interval 1 for caching

We add the virtual interval concept to this. A *virtual interval* is defined as the distance of the offsets between the latest request on an object and the *virtual request* on that object (Fig. 2*b*). A virtual request is not a real request from a client but a predicted request that is expected to be generated at that time based on the past requests on an object. To predict forthcoming requests precisely we use an exponential average in calculating the virtual interval which puts more weight on the latest interval but also considers previous intervals. Let I be the latest interval and VI_{n-1} be the $(n-1)$th virtual interval. Then, the nth virtual interval VI_n is computed as

$$VI_n = \alpha \cdot I + (1-\alpha) \cdot VI_{n-1} \tag{1}$$

where α is a constant between zero and one, and determines how much weight is put on the latest interval. We set the default value of α as 0.6 through empirical analysis. Since the virtual interval is updated only when a new request on the object arrives, it may be overestimated when there are no requests for a long time. To resolve this phenomenon, we use an adjustment function. Let t_n be the time since the latest request arrived. Then, the adjusted value should be close to the original value when t_n is small, and it should be large enough when t_n becomes large. The following adjustment function satisfies these requirements.

$$f_n = e^{t_n/VI_n} \tag{2}$$

After this adjustment, (1) is replaced by

$$VI'_n = f_n \cdot VI_n \tag{3}$$

and the adjustment function is invoked periodically. PIC allocates the cache space to intervals (including virtual intervals) by increasing order of the interval size. Since the cache operations need to find only the largest interval in the cache when deciding whether a new interval will be cached or not, the intervals do not need to be completely sorted.

Hence, we use the heap data structure for an efficient implementation. By allocating the cache to as many intervals as possible, PIC could maximise the number of concurrent streams serviced from the cache. Furthermore, through the virtual interval concept, PIC caches the prefix of popular multimedia objects before they are actually requested. This could eventually reduce the start-up latency of popular streams perceived by users which was not possible to the interval caching policy.

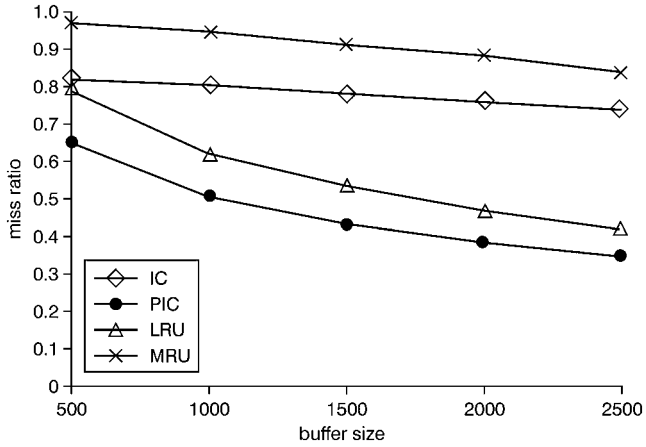

Fig. 3 *Performance comparison*

Performance evaluation: We gathered traces from several commercial VOD servers and performed extensive simulations to compare PIC with IC (interval caching), LRU and MRU. Among various results, we presented only one result. (Note that results from other traces are similar.) The trace shown in the result has 274 video files, the average playback time of which is 167 s with an average inter-arrival time of 44 s [3]. Fig. 3 shows the miss ratio of each scheme against the memory buffer size. The PIC scheme shows consistently better performance than the other three schemes. Specifically, PIC performs better than IC, LRU and LFU up to 40%, 14% and 49%, respectively. The performance of IC in our experiments is not so good as in [1]. This is because the object size is relatively small and the inter-arrival time is long in our traces. Note that we used real world traces while experiments in [1] were performed with synthetic traces.

Conclusion: We have presented the Popularity-aware Interval Caching (PIC) scheme for multimedia streaming servers. By considering reference popularity as well as the request interval of multimedia objects, the PIC scheme performs better than interval caching, LRU and MRU in terms of cache miss ratio for various real VOD traces we considered.

Acknowledgment: The authors thank Myung Films for making their traces available.

23 July 2003
Electronics Letters Online No: 20030965
DOI: 10.1049/el:20030965

Taeseok Kim and Kern Koh (*School of Computer Science and Engineering, Seoul National University, Korea*)

Hyokyung Bahn (*Department of Computer Science and Engineering, Ewha Womans University, Korea*)

References

1 DAN, A., and SITARAM, D.: 'Buffer management policy for an on-demand video server'. IBM Research Report, RC 19347, 1993
2 OZDEN, B., RASTOGI, R., and SILBERSCHATZ, A.: 'Buffer replacement algorithms for multimedia storage systems'. Proc. IEEE Int Conf. Multimedia Computing and Systems, Japan, 1996, pp. 172–180
3 Myung Films Co. Ltd, http://www.myungfilm.com.