# Virtual Interval Caching Scheme for Interactive Multimedia Streaming Workload

Kyungwoon Cho[1], Yeonseung Ryu[2⋆], Youjip Won[3⋆⋆], and Kern Koh[1]

[1] School of Computer Science and Engineering, Seoul National University, Korea
[2] Department of Computer Software, Myongji University, Korea
[3] Division of Electrical and Computer Engineering, Hanyang University, Korea

**Abstract.** We carefully believe that server workload for interactive multimedia service exhibits different characteristics from legacy application, e.g. ftp server, web server, ASP server and etc., and that there are greater chance of improvement with better understanding of streaming workload. In this work, we present a novel buffer management algorithm for interactive multimedia streaming workload. We carefully examine the workload traces obtained from several streaming servers in service. The analysis results show that most users exhibit non-sequential access pattern using VCR-like operations such as jump backward and jump forward. We exploit the workload characteristics of the VCR-like operation and develop a buffer caching algorithm called *Virtual Interval Caching* scheme. Simulation based experiment shows that the proposed buffer management scheme yields better performance than the legacy schemes.

## 1 Introduction

Buffer cache management scheme for multimedia workload has been the subjects of numerous works during past several years. Most of these works assume that the streaming workload exhibits sequential access characteristics with bandwidth guarantee. There have been widely proposed interval-based buffer caching schemes for multimedia servers [6,2,4]. Interval-based buffer caching schemes take advantage of sequential access characteristics of continuous media data. They cache data blocks in the *intervals* of consecutive streams accessing the same file. In order to maximize the number of playbacks accessing data from buffer cache, interval based schemes sort intervals with increasing order and cache data blocks from the shortest intervals.

A few studies recently examine the user access logs obtained from streaming servers in service and show that there are non trivial amount of VCR-like operations, e.g. jump forward, jump backward, or etc [13,1,3,12,7]. These works also deliver insightful information on the usage of the multimedia server and the user behavior, e.g. access frequency distribution, file popularity, aging of file

⋆ Corresponding author

⋆⋆ Work of this author is in part funded by Hanyang Faculty Research Initiative Grant 2002.

A. Yazici and C. Şener (Eds.): ISCIS 2003, LNCS 2869, pp. 276–283, 2003.

access popularity, etc. From these works we can easily see that the data access pattern is more than sequential. For example, [1] showed that for short media files, interactive requests like jump backward are common.

In this work, we analyze the user access behavior on three different types of contents: (i) educational contents, (ii) game contents, and (iii) movie. Typical user accesses the content via broadband Internet access, e.g. ADSL, cable modem. User access patterns for these contents are extracted from the commercial streaming service site: *Hanmir Education Center*[8], *Ongamenet*[10] and *Myung Films*[9], respectively. There is no geographical limitation on accessing these contents since these are accessed via public network. Examining the access pattern on this site enables us to understand the access behavior in broadband Internet access. We find that non-sequential accesses constitutes significant fraction of multimedia file accesses. There are relatively large number of VCR-like operations, e.g. jump forward, jump backward, etc. The fraction of VCR-like operations varies dependent upon the type of contents.

We aim at developing the buffer cache management algorithm which is optimized for *real* streaming workload. In order to manages the buffer space effectively exploiting the streaming workload which is mixture of sequential and bidirectional skip operation, we develop buffer management scheme, *Virtual Interval Caching (VIC)*. The proposed VIC scheme maintains past intervals as virtual intervals and increases buffer cache hit ratio. Experimental results show that the proposed buffer management scheme yields better performance than the legacy schemes such as interval-based schemes, LRU, etc.

The remainder of this paper is organized as follows. In section 2, we show the access activity to media files by analyzing trace data obtained from three streaming servers in service. In section 3, we presents proposed buffer caching scheme called virtual interval caching. Section 4 presents performance results and finally section 5 summarizes our works.

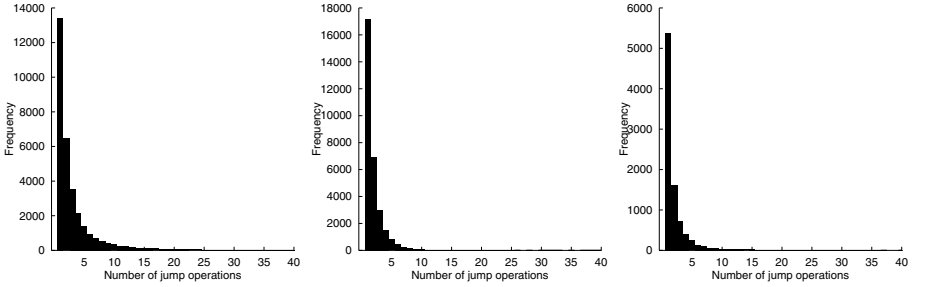
## 2 Analysis of Multimedia Streaming Workload

To characterize the user access pattern over video contents, we analyze the access logs[5] from three different streaming service site each of which services different types of contents: (i) education, (ii) game and (iii) movie. They are *Hanmir Education Center* [8], *Ongamenet* [10] and *Myung Films* [9]. Table 1 illustrates the summary of the analysis. It is worth noting that there are the significant number of VCR-like operations such as *jump forward* or *jump backward*. In case of educational contents in *www.hanmir.co.kr*, the number of jump operations per each session is 1.62 on the average. Fig. 1 illustrates the distribution of the number of jump operations for each site.

We define *Jump distance* as a distance between the playback positions before and after the jump operation. As is shown in Fig. 2, we find the this distance is short in most cases. However, there are small number of jump operations which skips most of the contents. This phenomenon can be clearly observed in Fig. 2(c). We carefully believe that intolerable nature of human behavior significantly contributes to this phenomena.

**Table 1.** Analyzed results of 3 Windows Media Servers

| | Hanmir | Ongamenet | Myung Films |
|---|---|---|---|
| **Log Duration** | 10 days | 3 days | 43 days |
| **Number of log entries** | 192457 | 98640 | 69330 |
| **Number of file** | 3432 | 953 | 468 |
| **File Length** | 1009 secs | 809 secs | 141 secs |
| **Number of sessions** | 73655 | 55423 | 42119 |
| **jump operations** | 1.62 | 0.78 | 0.64 |
| **backward distance** | 281.28 secs | 254.32 secs | 77.93 secs |
| **forward distance** | 311.68 secs | 153.63 secs | 26.31 secs |
| **Concurrent sessions** | 38.78 | 166.28 | 1.1 |




(a) *Hanmir* (b) *Ongamenet* (c) *Myung Films*

**Fig. 1.** The distribution of jump operations

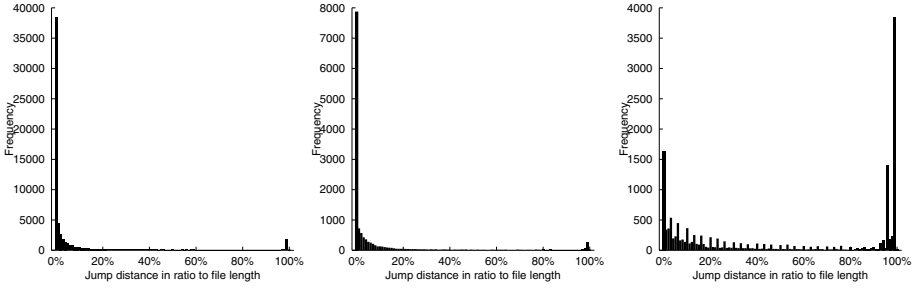
## 3 Virtual Interval Caching

### 3.1 Concept of Interval

We first introduce the concept of *interval* and briefly explain the existing interval-based buffer caching algorithms[6].

In Fig. 3, $S_i$ denotes the stream accessing the file $i$ and the small arrows marked by stream $S_i$ denote the current playback position. Interval is the distance between two session, e.g. $S_i$ and $S_j$. It is denoted by $I_{ij}$. For example, $I_{21}$ and $I_{32}$ are intervals formed by two consecutive streams. The two streams of an interval are referred to as the preceding and the following streams.

The basic idea of interval-based caching algorithms is that by caching the blocks brought in by the preceding stream, it is possible to serve the following stream from the cache. The cache requirement of an interval is defined to be the number of blocks needed to store the interval and is a function of the time-interval between the two streams and the compression method used. When the




(a) *Hanmir* (b) *Ongamenet* (c) *Myung Films*

**Fig. 2.** Jump distance distribution

server services multiple number of streams, it is possible that more than one stream accesses the same video contents and thus a number of interval is created. The objective of the interval caching scheme is to select the intervals to be cached so as to maximize the number of streams served from cache. If all the streaming session have the same playback rates and all accesses are sequential, caching the shortest interval first will yield the best performance.

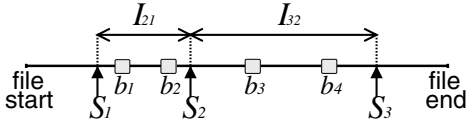



**Fig. 3.** Interval between two consecutive streams

When all streaming sessions are sequential playback only, the interval length does not change unless either preceding or following session finishes. Start of new streaming session creates new interval when there has been ongoing streaming session for the same file. When the successor stream finishes earlier than the predecessor stream, the respective interval is removed. Cache management module releases the space used by the interval and examines the next shortest interval for possible caching. If there exist VCR operations, intervals changes dynamically. The intervals can be merged, split, created and/or removed when VCR request is issued. The relative access position may be switched. For example, the successor can be predecessor and the predecessor can be successor as a result of jump backward operation.

### 3.2 Virtual Interval Caching

We define a *virtual interval* as an interval which is associated with only one stream or is not associated with any stream. A *real interval* is defined as an interval which has both a preceding stream and a following stream. We also define a *head interval* as an interval whose immediate following stream accesses the latest portion of the file. The head interval can have no preceding stream. Likewise, an interval whose immediate preceding stream is at the earliest position in the file is a *tail interval*.

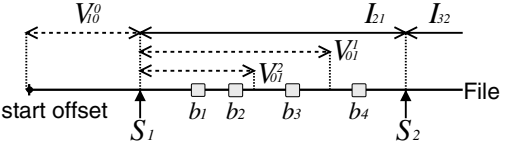
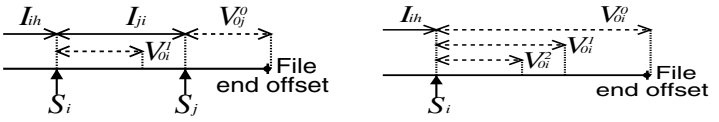



**Fig. 4.** Virtual Interval

Fig. 4 shows relationship between real interval and virtual interval. $I_{21}$ represents a real interval which is generated by two streams $S_2$ and $S_1$. In contrary, $V_{01}^1$, $V_{01}^2$ and $V_{10}^0$ are virtual intervals which have only a following stream or neither of streams. We can see that $I_{21}$, $V_{01}^1$, $V_{01}^2$ have the same following stream but only $I_{21}$ has a preceding stream.

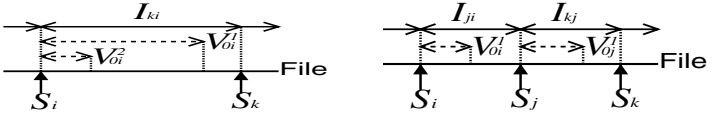
An interval becomes *virtual* when a stream disappears due to the change in playback position, e.g. *stop* or *jump*. Virtual interval can also be split into two smaller ones or can be reduced to a smaller one when a new stream arrives or an existing stream resumes playing from a new position. Fig. 5 shows that new virtual interval is created. In this figure, as the preceding stream of $I_{ji}$ disappears, the respective interval becomes virtual and we denote the respective interval as $V_{0i}^1$. Its interval length does not change. However, the interval size of $V_{0j}^0$ becomes longer because its immediate following stream becomes $I_{ih}$. Fig. 6 illustrates that a virtual interval becomes smaller due to the advent of new stream. As a result of the advent of the stream $S_j$, $I_{ki}$ is split into $I_{ji}$ and $I_{kj}$. At the same time $V_{0i}^1$ in Fig. 6(a) also get split but is regarded as being reduced into $V_{0j}^1$ because its immediate following stream becomes $S_j$.

The objective of buffer cache management policy is to minimize buffer cache miss rate. Our buffer cache management algorithm compute the access probability of individual intervals and caches the interval with largest *normalized* access probability. As an approximation to access probability of each interval, *virtual interval caching* conducts replacement policy based upon access probabilities of each interval which is linearly proportional to interval length. In the case of a tail interval, interval size should be estimated from a average number of streams within a file as $max\{interval\ size\ , \frac{file\ length}{average\ number\ of\ streams}\}$. By treating the virtual intervals as same as regular interval, access probabilities of intervals can




(a) Before $S_j$ disappears (b) After $S_j$ disappears

**Fig. 5.** A real interval is converted into a virtual interval due to the disappearance of an existing stream




(a) Before $S_j$ appears (b) After $S_j$ appears

**Fig. 6.** A virtual interval is reduced to a smaller one due to the advent of a new stream

be computed using the uniform method. As such, *VIC* selects a buffer in the interval with largest virtual size as a victim at replacement time.

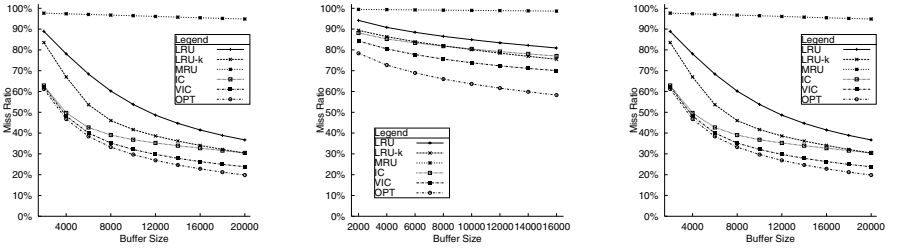
## 4 Simulation

In this section, we examine the performance of VIC based replacement algorithm via simulation based experiment. We compare the performance of VIC based buffer replacement algorithm with legacy algorithms: Interval Caching(IC), LRU, LRU-k[11][1], MRU and Optimal algorithm(OPT). The performance metric we use is cache miss ratio.

Fig. 7 illustrates buffer cache miss ratio under different cache management policy. The system is fed with two different workloads: (i) access trace from online game site(Fig. 7(a)) and (ii) access trace from movie-on-demand site(Fig. 7(c)). The size of the buffer cache varies from 2000 to 16000 (in terms of the number of buffer blocks). According to our simulation study, VIC exhibits superior buffer cache miss ratio to other algorithms. VIC is average 6% better than IC or LRU-k.

Figure 7(a) shows results using game trace(*Ongamenet*). In this server, there are relatively larger number of concurrent streams. This implies that there exist relatively larger number of intervals. This leads to good performance of interval caching and virtual interval caching, which is very close to OPT. With small buffer size (i.e., less than 8000), difference of miss ratio between IC and VIC

[1] Throughout experiments LRU-k takes into account knowledge of the last two references.




(a) *Ongamenet* trace (b) *Hamir* trace (c) *Myung Films* trace

**Fig. 7.** Miss ratio comparison

is not significant. However, VIC still exhibits better miss ratio as buffer size increases. The miss ratio of the VIC is 6.7% lower than IC or LRU-k and 12.9% lower than LRU with buffer size 20000.

Results with Movie-On-Demand trace(*Myung Films*) is shown in Fig. 7(c). When buffer size is 500, VIC is 2.8% and 8.6% better than LRU-k and IC, respectively. However, we can see that VIC does not provide a lower miss ratio any more as the buffer size increases. This is because there are too few concurrent streams and thus the number of intervals is not so sufficient that VIC runs effectively.

It is worth noting that there are significant amount VCR operations, e.g. *backward jump* and *forward jump* in real world trace data. In media servers with such workload, proposed VIC scheme can deliver better performance than legacy schemes.

## 5 Conclusion

In this work, we show that there exist a significant number of non-sequential access in real multimedia workload through analysis of real-world media server's log. Most of the non-sequential file accesses are jump forward or backward operations, which are triggered by VCR control capable client player. Moreover, the distance of skip(or jump) operations are in most cases very short. We need new buffer caching algorithms that exploit various file access workload in multimedia streaming servers.

Our buffer replacement algorithm, *Virtual Interval Caching (VIC)*, handles buffer cache effectively under such a real workload that a non-sequential access pattern exists. Unlike the existing interval caching scheme, VIC keeps intervals that were removed by VCR operations as virtual intervals. By keeping data blocks in the buffer cache that frequently accessed by VCR operations, VIC significantly improves the cache hit probability. We showed through trace-driven simulations that VIC algorithm can perform over 5% better than well-known legacy algorithms.

**Acknowledgement.** Authors would like to thank Sun-Nyoung Kim for collecting access traces from various streaming service sites.

## References

1. Jussara M. Aimeida, Jeffrey Krueger, Derek L. Eager, and Mary K. Vernon. Analysis of educational media server workloads. In *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video*, June 2001.
2. Matthew Andrews and Kameshwar Munagala. Online algorithms for caching multimedia streams. In *European Symposium on Algorithms*, pages 64–75, 2000.
3. M. Chesire, A. Wolman, G. Voelker, and H. Levy. Measurement and analysis of a streaming media workload. In *Proceedings of $3^{rd}$ USENIX Symp. on Internet Technologies and Systems*, San Francisco, CA, USA, March 2001.
4. K. Cho, Y. Ryu, Y. Won, and K. Koh. A hybrid buffer cache management scheme for vod server. In *Proceeding of IEEE International Conference on Multimedia and Expo*, August 2002.
5. Microsoft Corporation. Windows media service sdk, version 4.1.
6. A. Dan, Y. Heights, and D. Sitaram. Generalized interval caching policy for mixed interactive and long video workloads. In Proc. of SPIE's Conf. on Multimedia Computing and Networking, 1996.
7. N. Harel, V. Vellanki, A. Chervenak, G. Abowd, and U. Ramachandran. Workload of a media-enhanced classroom server. In *Proceedings of IEEE Workshop on Workload Characterization*, Oct. 1999.
8. Hanmir education center. `http://edu.hanmir.com`.
9. Myung films. `http://www.myungfilm.co.kr`.
10. Ongamenet. `http://www.ongamenet.com`.
11. E. O'Neil, P. O'Neil, and G. Weikum. Page replacement algorithm for database disk buffering. SIGMOD Conf., 1993.
12. J. Padhye and J. Kurose. An empirical study of client interactions with a continuous-media courseware server. In *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video*, July 1998.
13. Lawrence A. Rowe, Diane Harley, and Peter Pletcher. Bibs: A lecture webcasting system. Technical report, Berkeley Multimedia Research Center, UC Berkeley, June 2001.