International Journal of Internet, Broadcasting and Communication Vol.14 No.1 31-36 (2022)
http://dx.doi.org/10.7236/IJIBC.2022.14.1.31

IJIBC 22-1-5

Optimality of Interval Caching Policies in Multimedia Streaming Systems

Kyungwoon Cho and Hyokyung Bahn

Department of Computer Engineering, Ewha University, Professor, Korea
bahn@ewha.ac.kr

Abstract

Interval caching is one of the representative caching strategies used in multimedia streaming systems. However, there has been no theoretical analysis on interval caching. In this paper, we present an optimality proof of the interval caching policy. Specifically, we propose a caching performance model for multimedia streaming systems and show the optimality of the interval caching policy based on this model.

Keywords: *Cache replacement, interval caching, multimedia, streaming, cache performance*

1. Introduction

Caching has been widely studied to alleviate the performance gap in various computing systems including embedded systems [1], mobile systems [2], web servers [3], and smartphones [4]. Multimedia data caching has also great impact on the performance of streaming server systems. Over the last decades, a lot of cache replacement algorithms have been proposed in the literature and lots of them are based on interval caching [5, 6, 7]. Interval caching fairly well reflects the characteristics of streaming media, which is very large in size and is likely to be accessed sequentially. It is known that interval caching empirically performs well [5], but there is no theoretical background to investigate the effectiveness of interval caching. In this paper, we show the optimality of interval caching in terms of the cache miss ratio. To do so, we design a caching performance model for multimedia streaming systems and analyze the optimality of the interval caching policy based on this model.

The remainder of this paper is organized as follows. Section 2 describes the caching performance model of multimedia streaming systems for interval caching. Section 3 describes the revised interval caching policy based on the caching performance model. Section 4 will show the optimality of revised interval caching with respect to the cache miss count in streaming environments. Finally, Section 5 concludes this paper.

2. Caching Performance Model

Our caching performance model is developed to evaluate the performance of caching algorithms in streaming systems. We assume that a single media cache block is streamed periodically in a fixed duration

Manuscript Received: November. 5, 2021 / Revised: November. 10, 2021 / Accepted: November. 12, 2021
Corresponding Author: bahn@ewha.ac.kr
Tel: +82-2-3277-2368, Fax: +82-2-3277-2306
Professor, Department of Computer Engineering, Ewha University, Korea

round and every cache block loaded from a file block per each round has the same size. Two neighboring streams on the same file form an *interval*, which denotes the distance of the file blocks referenced by the streams. Two streams of an interval can be named as a preceding stream and a following stream according to their file block positions. The following stream of an interval tries to access a cache block in an interval and if the interval has no matching cache block, cache miss occurs. For example, two consecutive streams S_1 and S_2 in Figure 1 form *interval* I_{21}. In this example, S_2 is the preceding stream of I_{21} and at the same time the following stream of I_{32}. As we see, S_2 will incur a cache miss in the next round whereas S_1 will be serviced from the cache block in I_{21}. Such an interval concept has been the main idea of many caching algorithms in multimedia systems [8, 9, 10]. For example, with a given cache space, the original interval caching [5] sorts the intervals by the increasing order of space requirements and caches from the shortest interval.

In our study, interval approach is used in order to design an analytic performance model, where the sum of cache block misses incurred per each round is used as the performance metric. Specifically, the stochastic miss count of a stream is calculated using the interval concept. Cache blocks can be grouped according to which interval their corresponding file blocks belong to. An interval can be regarded as a container of cache blocks and cache miss count can be estimated based on the ratio of the maximal cache blocks an interval can hold to the number of actually allocated blocks. The former is defined as *interval size* and the latter is termed as *interval allocation*. It should be noted that *interval size* is invariant with respect to a round whereas *interval allocation* at each round may change.

There may exist a number of intervals in the system and they can be represented as an ordered list which is decreasingly sorted by their *interval size*. For an ordered interval list ($I_1 I_2 \dots I_n$), *size list* can be defined as ($\psi_1 \psi_2 \dots \psi_n$) where ψ_x is *interval size* of I_x ($1 \le x \le n$). Also we can define *allocation list* which consists of *interval allocation* of each interval at round t and it will be denoted as ($\phi_1^t\ \phi_2^t\ \cdots\ \phi_n^t$). From these two lists we can derive stochastic cache miss counts incurred during a round period.

Suppose that *size list* is Ψ and *allocation list* is Φ at round t. Then cache miss counts generated by caching algorithm $\mathbb{A}$ from round $t + j$ to round $t + k$ can be denoted by $C_j^k(\mathbb{A},\Psi,\Phi)$. It implies that cache miss counts can be regarded as being affected by the states of *size list* and *allocation list* at each round.

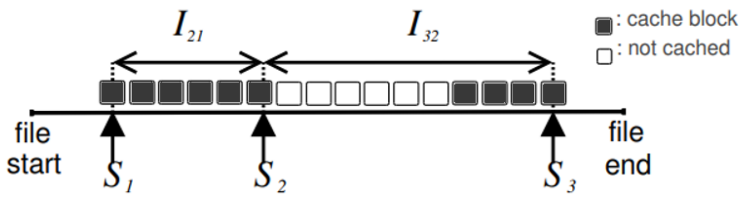

Figure 1. Interval between two consecutive streams

3. Revised Interval Caching

Revised Interval Caching (RIC) is reinvented from the original interval caching [5] to be optimized for the caching performance model. Note that RIC behaves similar to the original interval caching, so their empirical performance may be similar, but we devise it as the optimality can be proven with our revised interval caching. To illustrate RIC formally, let us suppose that *size list* is ($\psi_1 \psi_2 \dots \psi_n$) and *allocation list* which RIC has at round *k* is $(\phi_1^k\ \phi_2^k\ \cdots\ \phi_n^k)$. Then RIC satisfies the following conditions where

$$\Delta_m^k = \sum_{x=1}^{m-1} \phi_x^k - \sum_{x=m+1}^{n} \left(1 - \left\lceil \frac{\phi_x^k}{\psi_x} \right\rceil \right) , \quad 1 \le m \le n \, .$$

$$\phi_m^{k+1} = \begin{cases} \phi_m^k + 1 & \Delta_m^k > 0 \text{ and } \phi_m^k < \psi_m \\ \phi_m^k + \Delta_m^k & \Delta_m^k < 0 \text{ and } \phi_m^k + \Delta_m^k \ge 0 \\ 0 & \Delta_m^k < 0 \text{ and } \phi_m^k + \Delta_m^k < 0 \\ \phi_m^k & \text{otherwise} \end{cases}$$

Note that Δ^k_m is the difference between the number of cache blocks allocated to intervals whose size is longer than ψ_m and the number of uncached blocks for those intervals whose size is shorter than ψ_m. RIC tries to allocate more cache blocks to intervals that have a shorter *interval size*. If there exists a cache block in a longer interval while a shorter interval is not fully allocated, that cache block will be allocated to the shorter interval. Accordingly, all cache blocks will be replaced into shorter intervals sequentially according to the ordered interval list. As time progresses, a new interval may form or an existing interval may disappear, and thus the allocation of cache blocks to each interval needs to be adjusted. For each round, RIC allocates blocks to shorter intervals preferentially, and if a free cache block is needed, a block allocated to the longest interval is removed first. Thus, some intervals may be granted to maximal blocks in a certain rounds, while it is not the case in other rounds depending on how many short intervals exist in that round.

As the service round goes on, intervals managed by RIC will be partitioned into two groups: a shorter interval group with cache blocks fully allocated and a longer interval group with no cache block. Specifically, intervals with a smaller index (a larger *interval size*) than I_α have no cache block where α is an *allocation index* which satisfies $\sum_{x=\alpha+1}^{n} \psi_x < N \le \sum_{x=\alpha}^{n} \psi_x$ and N is the number of cache blocks. *Allocation list* in such a situation is said to be a Shorter-Interval-First-Allocated (SIFA) list.

4. Optimality Proof

In this section, we will show the optimality of RIC with respect to the cache miss count in streaming environments.

Lemma 1. *Suppose that Φ^s is a SIFA list for interval size list Ψ. Then for any allocation list Φ,*

$$C_0^1(\mathbb{A}, \Psi, \Phi) - C_0^1(\mathbb{RIC}, \Psi, \Phi^s) \ge 0$$

Proof. For $(I_1 I_2 \ldots I_n)$, let $\Psi = (\psi_1 \psi_2 \ldots \psi_n)$, $\Phi^s = (\phi_1^s\ \phi_2^s\ \cdots\ \phi_n^s)$, and $\Phi = (\phi_1 \phi_2\ \ldots\ \phi_n)$. From the fact that expected cache miss count per each interval I_x is $1 - \varphi_x/\psi$, we have

$$C_0^1(\mathbb{A}, \Psi, \Phi) - C_0^1(\mathbb{RIC}, \Psi, \Phi^s) = \sum_{x=1}^{n} \left(1 - \frac{\phi_x}{\psi_x}\right) - \sum_{x=1}^{n} \left(1 - \frac{\phi_x^s}{\psi_x}\right)$$
$$= \sum_{x=1}^{n} \left(\frac{\phi_x^s - \phi_x}{\psi_x}\right) \qquad (1)$$

By letting $\phi_x^{diff} = \phi_x^s - \phi_x$ and α be the *allocation index*, we get

$$\sum_{x=1}^{n} \frac{\phi_x^{diff}}{\psi_x} = \sum_{x=1}^{\alpha-1} \frac{\phi_x^{diff}}{\psi_x} + \frac{\phi_\alpha^{diff}}{\psi_\alpha} + \sum_{x=\alpha+1}^{n} \frac{\phi_x^{diff}}{\psi_x} \tag{2}$$

In (2), ϕ_x^{diff} where $x < \alpha$ is negative or 0 and ϕ_x^{diff} where $x > \alpha$ is positive or 0. So

$$\sum_{x=1}^{\alpha-1} \frac{\phi_x^{diff}}{\psi_x} \geq \frac{\sum_{x=1}^{\alpha-1} \phi_x^{diff}}{\psi_\alpha}\ ,\ \sum_{x=\alpha+1}^{n} \frac{\phi_x^{diff}}{\psi_x} \geq \frac{\sum_{x=\alpha+1}^{n} \phi_x^{diff}}{\psi_{\alpha+1}}$$

are derived and we can rewrite (2) as

$$\sum_{x=1}^{n} \frac{\phi_x^{diff}}{\psi_x} \geq \frac{\sum_{x=1}^{\alpha-1} \phi_x^{diff}}{\psi_\alpha} + \frac{\phi_\alpha^{diff}}{\psi_\alpha} + \frac{\sum_{x=\alpha+1}^{n} \phi_x^{diff}}{\psi_{\alpha+1}} \tag{3}$$

To apply

$$\sum_{x=1}^{n} \phi_x^{diff} = 0$$

to (3), we have

$$\begin{aligned}
\sum_{x=1}^{n} \frac{\phi_x^{diff}}{\psi_x} &\geq \frac{\sum_{x=1}^{\alpha-1} \phi_x^{diff}}{\psi_\alpha} + \frac{\phi_\alpha^{diff}}{\psi_\alpha} + \frac{\sum_{x=\alpha+1}^{n} \phi_x^{diff}}{\psi_{\alpha+1}} \\
&= \frac{-\sum_{x=\alpha}^{n} \phi_x^{diff}}{\psi_\alpha} + \frac{\phi_\alpha^{diff}}{\psi_\alpha} + \frac{\sum_{x=\alpha+1}^{n} \phi_x^{diff}}{\psi_{\alpha+1}} \\
&= \frac{-\sum_{x=\alpha+1}^{n} \phi_x^{diff}}{\psi_\alpha} + \frac{\sum_{x=\alpha+1}^{n} \phi_x^{diff}}{\psi_{\alpha+1}} \\
&= \frac{\sum_{x=\alpha+1}^{n} \phi_x^{diff} \cdot (\psi_\alpha - \psi_{\alpha+1})}{\psi_\alpha \cdot \psi_{\alpha+1}} \geq 0
\end{aligned}$$

□

Theorem 1. *For any buffer management algorithm* $\mathbb{A}$ *there exists a constant c such that*

$$C_j^k(\mathbb{A}, \Psi, \Phi) - C_j^k(\mathbb{RIC}, \Psi, \Phi) \geq 0$$

where $c \leq j$.

Proof. Suppose that $\Phi^{\mathbb{A}}$ and $\Phi^{\mathbb{RIC}}$ are allocation lists after $\mathbb{A}$ and $\mathbb{RIC}$ manage buffers during j rounds and let $\Phi_x^{\mathbb{A}}$ and $\Phi_x^{\mathbb{RIC}}$ be allocation lists at x round after j rounds are serviced. We can choose such c that Φ^{RIC} can be a SIFA list. As a result, all $\Phi_x^{\mathbb{RIC}}$ are also SIFA. Thus,

$$\begin{aligned} C_j^k(\mathbb{A},\Psi,\Phi) - C_j^k(\mathbb{RIC},\Psi,\Phi) &= C_0^{k-j}\left(\mathbb{A},\Psi,\Phi^{\mathbb{A}}\right) - C_0^{k-j}\left(\mathbb{RIC},\Psi,\Phi^{\mathbb{RIC}}\right) \\ &= \sum_{x=1}^{k-j} C_0^1\left(\mathbb{A},\Psi,\Phi_x^{\mathbb{A}}\right) - \sum_{x=1}^{k-j} C_0^1\left(\mathbb{RIC},\Psi,\Phi_x^{\mathbb{RIC}}\right) \\ &= \sum_{x=1}^{k-j}\left(C_0^1\left(\mathbb{A},\Psi,\Phi_x^{\mathbb{A}}\right) - C_0^1\left(\mathbb{RIC},\Psi,\Phi_x^{\mathbb{RIC}}\right)\right) \end{aligned} \tag{4}$$

Because each term in Equation (4) is non-positive by Lemma 1, $C_j^k(\mathbb{A}, \Psi, \Phi) - C_j^k(\mathbb{RIC}, \Psi, \Phi)$ is also non-positive. □

4. Conclusion

A lot of caching algorithms for stream server environments have been proposed in the research community but even today, the LRU (least recently used) algorithm is still used commonly in the industry implementation. This is mainly because most developers think buffer caching in file-systems would suffice. However, we believe that a replacement technique based on interval caching can improve the performance of streaming services even more. In this paper, we have shown that our revised interval caching achieves an optimal performance based on our proposed caching performance model, which provides the theoretical foundation for evaluating the performance of a cache replacement algorithm in streaming environments.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C1009275) and also by the ICT R&D program of MSIT/IITP (2020-0-00121, development of data improvement and dataset correction technology based on data quality assessment).

References

[1] T. Kim and H Bahn, Implementation of the storage manager for an IPTV set-top box, IEEE Transactions on Consumer Electronics, vol. 54, no. 4, pp. 1770-1775, 2008.
DOI: https://doi.org/10.1109/TCE.2008.4711233

[2] J. Park, H. Lee, S. Hyun, K. Koh, H. Bahn, "A cost-aware page replacement algorithm for NAND flash based mobile embedded systems," in Proc. the seventh ACM international conference on Embedded software (EMSOFT), 2009.
DOI: https://doi.org/10.1145/1629335.1629377

[3] H. Bahn, H. Lee, S.H. Noh, S.L. Min, K. Koh, "Replica-aware caching for web proxies," Computer Communications, vol. 25, no. 3, pp. 183-188, 2002.
DOI: https://doi.org/10.1016/S0140-3664(01)00365-6

[4] D. Kim, E. Lee, S. Ahn, H. Bahn, "Improving the storage performance of smartphones through journaling in non-volatile memory," IEEE Trans. Consumer Electronics, vol. 59, no. 3, pp. 556-561, 2013.
DOI: https://doi.org/10.1109/TCE.2013.6626238

[5] A. Dan and D. Sitaram, "Buffer Management Policy for an On-Demand Video Server," IBM Research Report, RC 19347, Yorktown Heights, NY, 1993.

[6] A. Dan, Y. Heights, and D. Sitram, "Generalized interval caching policy for mixed interactive and long video workloads," In Proc. SPIE Conf. Multimedia Computing and Networking, 1996.
DOI: https://doi.org/10.1117/12.235887

[7] T. Kim, H. Bahn, and K. Koh, “Popularity-aware interval caching for multimedia streaming servers,” Electronics Letters, vol. 39, no. 21, pp. 1555-1557, 2003.
DOI: https://doi.org/10.1049/el:20030965

[8] O. Kwon, H. Bahn, and K. Koh, “Popularity and prefix aware interval caching for multimedia streaming servers,” in Proc. 8th IEEE International Conference on Computer and Information Technology, pp. 555-560, 2008.
DOI: https://doi.org/10.1109/CIT.2008.4594735

[9] L. Dong and B. Veeravalli, “Design and analysis of a variable bit rate caching algorithm for continuous media data,” Multimedia Tools and Applications, vol. 38, no. 1, pp. 91-117, 2008.
DOI: https://doi.org/10.1007/s11042-007-0151-6

[10] L. Wujuan, L.S. Yong and Y.K. Leong, “A client-assisted interval caching strategy for video-on-demand systems,” Computer Communications, vol. 29, no. 18, pp. 3780-3788, 2006.
DOI: https://doi.org/10.1016/j.comcom.2006.06.014