Design and Analysis of Variable Bit Rate Caching Strategies for Continuous Media Data

Ligang Dong* and Bharadwaj Veeravalli**
*College of Information and Electronic Engineering
Hangzhou University of Commerce
149 Jiaogong Road, Hangzhou, Zhejiang, P.R. China, 310035
Email: donglg@mail.hzic.edu.cn
**Department of Electrical and Computer Engineering
The National University of Singapore,
10, Kent Ridge Crescent, Singapore 119260
Email: elebo@nus.edu.sg

Abstract: We designed and analyzed three variable bit rate caching strategies for continuous media data. Firstly, we use a just-in-time scheme to improve the utilization efficiency of the resource (the cache space and the cache I/O bandwidth). Secondly, the assumption of constant retrieval bandwidth in past caching algorithms is nearly impossible in a practical multimedia system. We propose the strategy used in the case of variable retrieval bandwidth as well as constant retrieval bandwidth. Thirdly, the phenomena of “switching” in past caching algorithms significantly impacts the system performance, therefore, we propose and satisfy a non-switch constraint, so that the probability of switching operation is reduced to a maximum extent. The simulation result confirms our analysis.

Keywords: Multimedia caching, disk caching, memory caching, resource management, non-switch constraint, variable bit rate

I. INTRODUCTION

The problem of designing efficient data caching algorithms has always been an interesting study. The research of caching lies in carefully exploiting the data characteristics, the frequency with which the objects/data are accessed, and resource management. Two categories of the caching are memory caching and disk caching [13]. Memory caching refers to caching objects in the main memory whereas, disk caching refers to caching objects in the hard disk. Bandwidth of a memory cache is rarely a bottleneck [14]. On the contrary, disk-caching policies have to consider constraints imposed by the disk bandwidth as well as the disk space. Typically, there are two kinds of multimedia documents [1]: continuous media (CM) data and non-continuous media (non-CM) data. Firstly, CM data (e.g., movies) can have an order of magnitude much larger than the non-CM data (e.g., text and image object). Hence, caching an entire CM document leads to a poor performance [14]. Secondly, we also need to guarantee the playback continuity of CM data.

In the past research, some strategies for caching CM data have been proposed, e.g., prefix-caching [8][11], work-ahead smoothing [12][10], prefetching [2]. Besides, there are two classes of caching strategies/algorithms for CM data. (1) Block-level algorithm, e.g., BASIC [9]. BASIC selects to replace a block that would not be accessed for the longest period of time. (2) Interval-level algorithm, e.g., DISTANCE [9], Interval Caching (IC) [3], Resource-Based Caching (RBC) [14], and Generalized Interval Caching (GIC) [4]. DISTANCE is observed to be similar to IC [4]. RBC is a disk-caching algorithm while DISTANCE, IC, and GIC are memory-caching algorithms. IC is suitable for caching long documents, and GIC extends IC so that both long and short CM document can be managed. In an interval-level algorithm, the basic caching entity is an interval, which is the amount of data that is in between two adjacent streams, so that the latter stream always can read the data cached by the former stream(s). In comparison with the interval-level caching algorithm, the block-level caching algorithm has much higher operational overhead. Also, it is difficult to guarantee a continuous playback at the client end for the block-level caching algorithm. Thus, we will consider designing efficient interval-level caching algorithms in this paper. Next, we analyze the drawbacks in past interval caching algorithms. Firstly, for any of the above algorithms, if a stream that is reading from the cache finds that there is no required data or sufficient retrieval bandwidth in cache before its next C/C interval, this stream has to retrieve the corresponding data from the original server (or disk storage in terms of memory caching). We refer to this simply as switching back to the server (either the disk storage in memory caching or the original server in disk caching). In order to avoid any discontinuity while retrieving the required data owing to shortage of resources (i.e., server I/O bandwidth and network bandwidth), the required amount of bandwidth should be reserved in advance. However, an estimation of the required amount of bandwidth is difficult for the server as the existence of caches and/or the access patterns are probably unknown by the server in advance. Furthermore, the switching operation not only causes additional bandwidth consumption, but also incurs additional processing overheads, and hence, it is undesirable. Thus, an efficient caching algorithm must also attempt to minimize the number of switches to the server, apart from improving the byte hit ratio. Thus, in this paper, one of our objectives is to design an interval-level caching algorithm that minimizes the number of switches to the server. Lee et al. [7] also notices the disadvantage of switching operations, and propose the
following strategy to eliminate the switching operation. However, the studies in [7] are only based on IC (i.e., interval caching) used in the memory caching. In fact, the shortage of bandwidth in disk caching may also result in the switching operation. Secondly, previous interval-level algorithms assumed that the streams that retrieve data of the same document have the identical retrieval bandwidth. However, this assumption is not always true. There are two reasons. (1) Different streams probably have different retrieval bandwidths. (2) The retrieval bandwidth of one stream probably varies with time. As a result, the cache space requirement of intervals and the cache I/O bandwidth requirement of streams will be variable. Past interval-level caching algorithms allocate the resource (including bandwidth and space) only once for every interval or every stream, and therefore past interval-level caching algorithms clearly cannot handle the case of variable retrieval bandwidth. Finally, in the case of disk caching algorithm RBC, the allocation of the cache I/O bandwidth for a stream is a reservation mode, which is less efficient. If we can allocate the bandwidth just-in-time instead of by a reservation, then the caching performance can be considerably improved.

The above-mentioned issues considerably motivate us to propose novel strategies for caching CM data that demand a variable bit rate retrieval. This paper is organized as follows. Firstly, in Section II, we introduce the system architecture and the performance metric. Then we describe the proposed strategies in Section III. Furthermore, in Section IV, we show the result of rigorous simulation and provide a detailed discussion. Finally, in Section V, we highlight the key contributions.

II. System Model

A. System architecture

The disk-caching strategies discussed in this paper are applicable to an architecture formed by “the server - the disk cache - the clients” configuration whereas, the memory-caching strategies discussed in this paper are applicable to an architecture formed by “the server - the memory cache - the clients” configuration. In memory caching, “the server” actually means “the local disk storage”. The connection between the server and the cache, or between the cache and the client is via a network. In this paper, for both the disk caching and the memory caching schemes, we only consider a single cache case (system with a single cache storage) as attempted by other researchers [9][10][14][4].

B. Performance metrics

In this paper, the design of caching strategies fundamentally aims to reduce the traffic overload on the network, through which the requested objects are transferred, and the server. In other words, the saving on the network bandwidth or the server I/O bandwidth is treated as the main benefit of caching. Therefore, we evaluate the caching strategies in terms of their ability to improve byte hit ratio. Byte hit ratio equals the ratio of the total number of bytes accessed from the cache to the total number of bytes accessed.

III. Design of Strategies

A. Caching strategy for the variable retrieval bandwidth

In Section I, we explained on why the retrieval bandwidth is often a variable. In this section, we will present strategies to handle the variable retrieval bandwidth. When the retrieval bandwidth is constant for all the streams that retrieve data of the same object, once we allocate the cache bandwidth for a stream, or allocate the cache space for an interval, there will be no need to check the availability of the bandwidth or space until the stream (or the two streams in the interval) finishes its or their retrieval. When the stream (or the two streams in the interval) finishes its or their retrieval, the allocated bandwidth or space is reclaimed. However, in the case of variable retrieval bandwidth, the cache bandwidth requirement of one stream and the cache space requirement of one interval are time-varying. Therefore we need to check the availability of the bandwidth and space in the cache every service cycle D. D is tunable. During a service cycle, the availability of bandwidth and space in the cache is checked. If the total requirement for bandwidth (\(\sum R_s\)) is greater than the cache I/O bandwidth capacity BW, some amount of bandwidth will be reclaimed from some streams using a cache bandwidth reclaim strategy (refer to [6]). The performance metric byte hit ratio is directly affected by the amount of data read from the cache. In comparison, when data are written into the cache, thesecached data is probably swapped out of the cache before the data is read because of not enough cache space. Therefore, a stream that is reading data from the cache is more “important” than a writing stream. As a result, we do not reclaim the bandwidth from a stream that is reading from the cache unless there is no writing stream.

Similarly, if the total requirement for space (\(\sum R_s\)) is greater than the cache space capacity A_s, some amount of space will be reclaimed from some intervals using the reclaim strategy for the cache space (refer to [6]).

In the case of variable retrieval bandwidth, the size of the retrieved data of a following stream probably exceeds a preceding stream. For example, when a preceding stream may be paused by a VCR operation, a following stream probably overtakes the preceding stream. In this case, to handle the repositioning of streams within an interval, we propose a strategy referred to as exchange strategy (refer to [6]).

In short, the cache bandwidth reclaim strategy, the cache space reclaim strategy, and the exchange strategy for repositioning the streams are special for the case of variable retrieval bandwidth. In other words, they are not needed in the case of constant retrieval bandwidth. With these strategies, the resource allocation in the case of variable retrieval bandwidth can be handled successfully.

B. Caching strategy under the non-switch constraint

In Section I, we explained the concept and reasons of switching. Because of the shortage of cached data or cache bandwidth, the stream has to switch back to the server and begin to read data from the server.

B.1 Influence of the switching operation on the performance

The switching operation affects the performance in two ways. One is the estimation of the bandwidth reser-
vation. The other is the overhead of switching operations. Firstly, since the streams that are reading data from the cache probably switch back to the server, for satisfying the continuity of these streams, enough I/O bandwidth should be reserved in the server. However, the estimation is very difficult because of the following reasons.

- In disk caching, the cache and the access pattern in the cache is unknown by the server. When the number of caches is very large, the server probably cannot accept all the switching streams even the server reserve all its bandwidth.
- In memory caching, the disk I/O bandwidth is much less than the memory bandwidth. The number of streams reading data from the memory is large, consequently, the number of switching streams is also unreasonably high for the disk with smaller I/O bandwidth. Secondly, it may be noted that the switching operation causes additional operational overhead. As a block-level caching algorithm, BASIC caches unrelated sets of blocks, and hence, BASIC cannot guarantee continuous playback. Therefore, BASIC is not applicable for the CM caching. In fact, the actual reason why BASIC cannot guarantee continuous playback is because of many switching operations that penalize the performance due to additional operational overheads that are prohibitively high.

B.2 Strategies for minimizing the switching operations

Here, a non-switch constraint is presented. If the non-switch constraint is satisfied in the caching algorithm, the switching operation will be eliminated. In the case of constant retrieval bandwidth, i.e., special retrieval bandwidth for the streams that retrieve the same object, the switching operation can be reduced to zero. In the case of variable retrieval bandwidth, the probability of a switching operation will be reduced to a maximum extent. The following two rules constitute the non-switch constraint.

- Allocated retrieval bandwidth for one stream that is reading from the cache will not be released until the current retrieval finishes. Thus, once a stream begins to read from the cache, the stream will never switch back to the server because of the shortage of bandwidth.
- When a stream is reading the data from the cache, there is at least one preceding stream, which is writing the data into the cache, or all the data of the requested object have been cached. Thus, one stream can always read available data from the cache, and the stream will not switch back to the server because of the shortage of data.

Through satisfying the non-switch constraint, the switching operation is eliminated or reduced, and therefore bandwidth of servers and networks is saved to a significant extent. Consequently, the acceptance ratio of requests in the entire system will be improved.

C. Allocation strategy of the cache I/O bandwidth

Before presenting better allocation strategy of bandwidth, the bandwidth requirement in the case of disk caching is analyzed. We analyze the bandwidth requirement for an interval (refer to Fig. 1) and two consecutive intervals as examples. For a simple description, the analyzed example adopts constant retrieval bandwidth. In RBC, when one interval is formed, the bandwidth is allocated for the former stream and the latter stream. However, Fig. 1 shows the allocated bandwidth for one stream is not actually utilized until there are available data in the cache for this stream. If we allocate the cache bandwidth when the stream is actually needed by the stream, then the bandwidth resource can be considerably saved. Our caching algorithm adopts this kind of so-called just-in-time method instead of reservation method employed in RBC. The saving of bandwidth in our algorithm, when compared to the bandwidth reservation method by RBC, is defined as $G = \int_0^{L/L_2} \delta bw dt$, where, $\delta bw$ is the amount of the saved bandwidth. Hence, the saving of bandwidth in Fig. 1 (c) and (d) are $L_1$. The bandwidth allocation algorithm of the disk cache is detailed in [6].

Using the three strategies introduced before, we design a new caching algorithm—variable bit rate caching (VBRC) algorithm, which can be used in disk caching or memory caching. Because of the limited space in this paper, we put the detail and discussion of the VBRC algorithm in [6].

IV. PERFORMANCE EVALUATION

In this section, we conduct rigorous simulation experiments to testify all our strategies presented in Section III. The performance measure in our study is the byte hit ratio under several influencing factors such as, disk cache size $B$, disk bandwidth $BW$, arrival rate $\lambda$, and percentage of requests for large objects $P$, respectively. The detail of simulation test, which follows past literatures, can be found in [6]. We show only part results of the simulation because of limited space in this paper.

A. Effect on performance due to non-switch constraint

In this section, the non-switch constraint is imposed on GIC, so that the influence on the performance due to the non-switch constraint can be observed. We compare the performance of GIC with non-switch constraint and GIC without non-switch constraint. The result shows GIC without non-switch constraint exhibits a little better performance than GIC with non-switch constraint with respect to byte hit ratio. However, note that here,
Only the performance of cache is shown. The performance improvement of GIC without non-switch constraint is achieved at the cost of poor utilization of bandwidth resource in the server. GIC with the non-switch constraint has gotten rid of all the switching operation while GIC without the non-switch constraint has to reserve bandwidth for the streams reading from the cache.

On the one hand, if we reserve bandwidth of the server for every stream that is reading data from the cache, then the total amount of bytes read from the server is obviously reduced. Note that, the amount of bytes read from the cache is equivalent to a resource wastage in the server. To understand this, assume $A_1$ is the amount of data read from cache, which is also the amount of resource wasted in the server, by reserving the server bandwidth for stream reading from cache. On the other hand, when we adopt a non-switch constraint to eliminate the switching operation, the byte hit ratio decreases. We denote $A_2$ as the difference in the number of bytes that are read from the cache without non-switch constraint and with non-switch constraint. Thus, we compare these two cases by observing the ratio $A_2/A_1$ and we found the average value of $A_2/A_1$ is less than 0.1 (refer to [6]). It means that to satisfy the non-switching constraint can bring benefits for the entire system (including the server and the cache).

B. Performance Comparison of RBC and VBRC

In the case of memory caching, it should be clear at this stage that VBRC is actually GIC with non-switch constraint. Hence we do not need to compare the performance of VBRC and GIC. Thus, in this section, we compare the performance of RBC and VBRC in the case of constant retrieval bandwidth (since RBC cannot be used in the case of variable retrieval bandwidth).

Because of limited space, we only use Fig. 2 as an example. In comparison with RBC, our VBRC algorithm adopts the “just-in-time” allocation method of bandwidth, and the allocation of space is among intervals and not the runs. We observe that VBRC has much better performance than RBC. Besides, there are no switching operations in VBRC (as we use non-switch constraint), while the switching operation exists in RBC.

V. Conclusions

In this paper, we designed and analyzed three strategies for caching CM data in the disk and the main-memory storage media. Firstly, we proposed the strategy to handle the case of variable retrieval bandwidth (or variable bit rate), which is beyond the ability of existing caching algorithms. Variable retrieval bandwidth demand is most commonly encountered in multimedia applications. Secondly, we reduced switching operations to a maximum extent, as these switching operations result in low utilization of bandwidths in servers. Finally, we proposed that the bandwidth resource is allocated in just-in-time manner instead of carrying out any advanced reservation. The performance of our strategies is conclusively testified to perform better than the popular disk caching algorithm RBC [14].

REFERENCES


Fig. 2. Performance comparison between RBC and VBRC ($B = 5000$ MB, $\lambda = 0.25/s$, $P = 80\%$)