Proceedings of Multimedia Computing and Networking (MMCN'2004), January 21-22, Santa Clara, CA, 2004

Investigating Performance Insights of Segment-based Proxy Caching of Streaming Media Strategies

Songqing Chen[a], Bo Shen[b], Susie Wee[b] and Xiaodong Zhang[a]

[a] Dept. of Computer Science, The College of William and Mary, Williamsburg, Virginia, USA;
[b] Mobile and Media System Lab, Hewlett-Packard Laboratories, Palo Alto, USA

ABSTRACT

In general, existing segment-based caching strategies target one of the following two performance objectives: (1) reducing client startup delay by giving a high priority to cache the beginning segments of media objects, or (2) reducing server traffic by caching popular segments of media objects. Our previous study has shown that the approach targeting the second objective has several advantages over the first one. However, we have also observed that the effort of improving server traffic reduction can increase client startup delay, which may potentially offset the overall performance gain. Little work so far has considered these two objectives in concert. In this paper, we first build an analytical model for these two types of typical segment-based caching approaches. The analysis on the model reveals the nature of the trade-off between two performance objectives and the bounds of each are given under certain circumstances. To provide a feasible way to evaluate different strategies, we propose a new comprehensive performance metric based on the analysis. To understand this performance trade-off, we restructure the adaptive-lazy segmentation strategy with a heuristic replacement policy to improve overall performance. The evaluation results confirm our analysis and show the effectiveness of our proposed new performance metric.

Keywords: Proxy Caching, Media Streaming, Startup Latency, Byte Hit Ratio

1. INTRODUCTION

Proxies have been widely used to cache Web pages on the Internet so that subsequent requests to the same object can be served directly from the proxy without contacting the origin server. Recently, proliferation of multimedia contents makes proxy caching more difficult. Due to the large sizes of multimedia objects, a full-object caching strategy can quickly exhaust the cache space. To solve this problem, a number of segment-based proxy caching strategies[1–6] have been proposed. These strategies cache segments of a media object instead of the entire object to reduce network traffic as well as the disk bandwidth requirement of the media server. The client startup delay is also reduced if beginning portions of media objects are cached. Based on their performance objectives, these caching methods can be classified into two types.

The first type of the segment-based caching methods focuses on reducing client startup delay by always giving a high priority to cache the beginning segments of media objects. If no beginning segments of a requested media object are cached, the client may experience a startup delay. In general, we use the delayed start request ratio to reflect this performance objective. The ratio is defined as the percentage of requests that has to be delayed because no beginning segment of the requested object is cached in the proxy. Among this type of caching methods, prefix caching[1,7] has been proposed to segment the media object as a prefix segment and a suffix segment. The proxy caches the prefix segments only and tries to cache as many prefix segments as possible. Prefix caching is effective when clients mostly access initial portions of media objects as noted in Ref. 8, 9. More recently, the uniform and exponential segmentation strategies have been developed. The uniform segmentation strategy segments all media objects into fixed-length segments. For example, Rejaie et al.[2] consider the caching of fixed-sized segments of layer-encoded video objects. The exponential segmentation strategy segments media objects

Emails for further correspondence: sqchen@cs.wm.edu, boshen@hpl.hp.com, swee@hpl.hp.com, and zhang@cs.wm.edu.

with an exponentially increased segment length.[4] The intuition of this strategy is based on the assumption that later segments of media objects are less likely to be accessed. Thus, the beginning segments are given a high priority for being cached. A combined use of these strategies can be found in Ref. 5, in which simple constant segment length and exponentially increased segment length are both considered.

The second type of segment-based caching methods tries to improve the server traffic reduction. In general, we use proxy byte hit ratio to reflect this performance objective. Byte hit ratio is defined as the percentage of requested bytes that can be served from proxy caches. An example of this type of caching methods is found in the adaptive-lazy segmentation work[6] proposed by us most recently. Based on the observation made in previous research[8,9] that client accesses to media objects always represent a skewed pattern with a lot of accesses to a few popular objects that are likely to be fully or largely accessed, the adaptive-lazy segmentation strategy favors the caching of popular segments without giving a high priority to beginning segments. Our previous study[6] shows that this caching strategy can achieve a higher byte hit ratio. However, we have also observed that the effort of improving the byte hit ratio has also increased the delayed start request ratio, which may potentially offset the overall performance gain.

To find out whether these two objectives can be considered together and to obtain insights into optimizing both performance objectives, we first build an analytical model for two types of segment-based proxy caching strategies. Specifically, we choose to build the model based on typical representatives of each type of caching strategy discussed above, namely, exponential segmentation and adaptive-lazy segmentation strategies. The analysis of the model reveals the nature of the trade-off between the byte hit ratio and delayed start request ratio. The analytical model also gives the performance optimization bounds under certain conditions for these segment-based proxy caching strategies. To provide a feasible way to evaluate different strategies, we also propose a new comprehensive performance evaluation metric, *weighted miss rate*, to evaluate both performance objectives in one unified metric.

To understand the aforementioned performance trade-off, we restructure the adaptive-lazy segmentation strategy with a heuristic replacement policy in order to achieve a better overall performance gain. The redesigned strategy dynamically divides the proxy into a 2-level cache and achieves both performance objectives adaptively. The evaluation results confirm our previous analysis and show the effectiveness of our proposed new performance metric.

The rest of the paper is organized as follows. In Section 2, we outline the exponential segmentation strategy and the adaptive-lazy segmentation strategy. The performance of both strategies is briefly compared to motivate our work in this study. The analytical model of both strategies and the corresponding analysis are presented in Section 3. Based on the analysis, we restructure the adaptive-lazy segmentation strategy and evaluate its performance in Section 4. Other related work is discussed in Section 5 before we conclude in Section 6.

2. RELATED WORK AND MOTIVATION

Among the existing caching strategies focusing on reducing the delayed start request ratio, studies[4,6] have shown that the exponential segmentation strategy performs the best. On the other hand, the adaptive-lazy segmentation strategy performs the best in improving the byte hit ratio.[6] In this section, we outline these two representative caching methods and briefly present their comparative performance results.

The following notations are used for both strategies.
(1) T_1: the time instance the object is accessed for the first time;
(2) T_r: the last reference time of the object. It is equal to T_1 when the object is accessed for the first time;
(3) T_c: the current time instance ;

2.1. Exponential Segmentation Strategy

The exponential segmentation strategy segments each media object exponentially. It then admits the segments of the object according to their relative positions in the object and their caching utilities by the admission policy. The segment replacement uses the LRU policy for the replacement of the beginning segments and always replaces the segment with the least caching utility for the later segments, respectively. More details can be found in Ref. 4.

2.1.1. Segmentation Method

A media object is divided into multiple equal-sized blocks. Multiple blocks are then grouped into a segment by the proxy. The size of a segment is sensitive to its distance from the beginning of the media object. The number of blocks grouped in segment i is 2^{i-1}. In general, segment i is twice as large as segment $i-1$. The purpose of this method is to allow the proxy to quickly discard a big chunk of cached media objects.

2.1.2. Admission Policy

A two-tiered approach is used for admission control. For a segment with a segment number smaller than a threshold, K_{min}, it is always eligible for caching. However, for a segment with a segment number equal to or larger than K_{min}, it is determined to be eligible for caching only if its caching utility is larger than some cached segments also with segment number equal to or larger than K_{min}. For this purpose, a portion of the cache space is reserved to store the beginning segments only while the remaining of the cache space is used to store the later segments. With such a cache admission control, at least the first K_{min} segments are stored for any cached objects by reserving a cache portion large enough for the beginning segments.

2.1.3. Replacement Policy

The caching utility of a segment depends on the reference frequency of an object and the segment distance. It is defined to be the ratio of reference frequency over the segment distance. The reference frequency is estimated as $\frac{1}{T_c - T_r}$. As a result, the caching utility of segment i of an object is defined as $\frac{1}{(T_c - T_r) \times i}$. According to the caching utility of the segment, two LRU stacks are maintained for the first K_{min} segments and the later segments. When an object is requested for the first time, the first K_{min} segments are always eligible for caching as a unit and the LRU scheme is used to find the replacement, while the later segments are always not cached for the first time. A later segment is eligible to cache only if its caching utility is greater than that of its replacement segment.

2.2. Adaptive-lazy Segmentation Strategy

In this strategy, each object is fully cached by the aggressive admission policy when it is accessed for the first time. The fully cached object is kept in the cache until it is chosen as an eviction victim by the replacement policy. At which time, the object is segmented using the lazy segmentation method and some segments are evicted by the replacement policy. From then on, the segments of the object are adaptively admitted or adaptively replaced segment by segment. More details can be found in Ref. 6. The following additional notations are needed to define this strategy.
(4) L_{sum}: the sum of the duration of each access to the object;
(5) n_a: the number of accesses to the object;
(6) L_b: the length of the base segment;
(7) n_s: the number of the cached segments of the object.
Thus, at time instance T_c, the access frequency F is denoted as $\frac{n_a}{T_r - T_1}$, and the average access duration L_{avg} is denoted as $\frac{L_{sum}}{n_a}$.

2.2.1. Aggressive Admission Policy

For any media object, the cache admission is considered aggressively in one of the following procedures whenever the object is accessed. (1) If the object is accessed for the first time, the whole object is subsequently cached regardless of the request's accessing duration. The cache space is allocated through the replacement policy if there is no sufficient space. (2) If the object has been accessed and is fully cached, no cache admission is necessary. (3) If the object has been accessed but it is not fully cached, the proxy aggressively considers to cache the $(n_s + 1)th$ segment if $L_{avg} \geq \frac{1}{2} \times (n_s + 1) \times L_b$. The inequality indicates that the average access duration is increasing to the extent that the cached n_s segments can not cover most of the requests while a total of $n_s + 1$ segments can for a normal distribution. Therefore, the proxy should consider the admission of the next uncached segment.

2.2.2. Lazy Segmentation Method

The basic idea of the lazy segmentation method is as follows. If the victim object chosen for replacement turns out to be fully cached, the proxy segments the object in the following way. The average access duration L_{avg} at that time instance is calculated. It is used as the length of the base segment of this object, that is, $L_b = L_{avg}$. Note that the value of L_b is fixed once it is determined. The object is then segmented uniformly according to L_b. After that, the first 2 segments are kept in cache, while the rest is evicted by the replacement policy.

2.2.3. Two-Phase Iterative Replacement Policy

By defining the utility function as $\frac{\frac{L_{sum}}{T_r - T_1} \times MIN\{1, \frac{\frac{T_r - T_1}{n_a}}{T_c - T_r}\}}{n_s \times L_b}$, the two-phase iterative replacement policy works as follows. Upon an object admission, if there is not enough cache space, the proxy chooses the object with the smallest utility value at that time as the victim, and the segment of this object is evicted in one of the two phases as follows. (1) First Phase: If the object is fully cached, the object is segmented by the lazy segmentation method. The first 2 segments are kept and the remaining segments are evicted right after the segmentation is completed. Therefore, the portion of the object left in cache is of length $2 \times L_b$. Given that $L_b = L_{avg}$ at this time instance, the cached 2 segments cover a normal distribution in the access duration. (2) Second Phase: If the object is partially cached, the last cached segment of this object is evicted. The utility value of the object is updated after each replacement and this process repeats iteratively until the required space is found.

2.3. Representative Performance Comparisons

In this section we present some representative performance data of these two strategies. The purpose of this is to show their different gains in the two performance objectives. The workloads used for the evaluation, WEB, VOD, PARTIAL and REAL, will be presented in section 4.2.

2.3.1. Comparisons on Complete Viewing Workloads

By using a complete viewing workload WEB, when the cache size is 10%, 20% and 30% of the total object size, the byte hit ratios achieved by the lazy segmentation and the exponential segmentation are about 50% and 13%, 67% and 29%, 75% and 39%, respectively (Note that in the following context, we always use lazy segmentation and exponential segmentation to represent their corresponding strategies.). The performance gap is more than 30% in average. At the same time, the delayed start request ratios achieved by the two strategies are about 44% and 12%, 34% and 3%, 29% and 2%, respectively. The gap is also about 30%. The performance result has a same trend when another complete viewing workload VOD is evaluated. More details will be presented in section 4.2.2.

2.3.2. Comparisons on Partial Viewing Workloads

Using a partial viewing workload PART, when the cache size is 10%, 20% and 30% of the total object size, the byte hit ratio achieved by the lazy segmentation is 28, 42, and 7 percentage points, respectively, more than those of the exponential segmentation. At the same time, the delayed start request ratio achieved by the lazy segmentation is 34, 24, and 13, percentage points, respectively, more than those of the exponential segmentation. The results using another partial viewing trace reflects the similar trends. More details will be presented in section 4.2.3.

2.4. The Major Theme of This Study

Motivated by the performance comparisons presented in the previous subsection, we want to ask the following three questions:

- **Q1**: What causes the lazy segmentation to have a high byte hit ratio along with a high delayed start request ratio and how?
- **Q2**: How to evaluate the overall performance of any strategy by a pair of performance objectives with conflicting interests?

- **Q3**: Is it possible to achieve the objective of both a high byte hit ratio and a low delayed start request ratio?

We can qualitatively answer **Q1** as follows*. The lazy segmentation always favors to cache popular segments without giving a high priority to the beginning segments, while the exponential segmentation strategy sets an opposite priority. To provide more insights into this and for answering **Q2** and **Q3**, we have developed an analytical model to be presented in the next section.

3. PERFORMANCE ANALYSIS OF SEGMENT-BASED PROXY CACHING STRATEGIES

An analytical model is built to analyze the two representatives: exponential segmentation and lazy segmentation for the ideal situation where the replacement policy always finds the victim with the least utility value for being replaced. Thus the effect of other factors can be excluded so that we can understand the insights of the reason as discussed in subsection 2.4. More importantly, with insightful understanding, we are able to make performance optimization by answering **Q2**.

3.1. Analytical Model

During a time period, there are n objects O_i the proxy has served, where $1 \leq i \leq n$. Several assumptions are made as follows:

1. The request arrival interval process follows Poisson distribution with mean arrival rate λ. The request arrival interval process to each individual object O_i is independently sampled from the aggregate arrival interval process based on probability set p_i, where $\sum_{i=1}^{i=n} p_i = 1$.
2. The popularity of the object follows a Zipf-like distribution and it models the probability set p_i, where $f_i = \eta \times \frac{1}{i^\theta}$, $\theta > 0$ and $p_i = \frac{f_i}{\sum_{i=1}^{i=n} f_i}$, i =1, 2, ...,n; η is the normalization factor. In our experiments, it is set as 1.
3. Assume that all the clients view their objects completely. This is a strong assumption to ease the analysis. We expect the conclusion is also valuable for partial viewing cases.

The first assumption means that the request arrival interval process follows Poisson distribution with a mean arrival rate $\lambda_i = \lambda \times p_i$.

The second assumption leads to

$$\lambda_i = \lambda \times \frac{\eta \times \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \eta \times \frac{1}{i^\theta}}. \quad (1)$$

Since the popularity of each object p_i follows Zipf-like distribution, thus $p_i \geq p_{i+1}$ and $\lambda_1 \geq \lambda_2 \geq \lambda_3 ... \geq \lambda_n$.

Some notations are presented as follows:

- L_i: the length of each object, where $1 \leq i \leq n$;
- C: the total cache size;
- C_{prefix}: the reserved cache space for the beginning segments in the exponential segmentation;
- β : C_{prefix} is the β percentage of the total cache size, i.e., $C_{prefix} = C \times \beta$;
- C_{rest}: the cache space other than the C_{prefix} in the exponential segmentation, where $C_{rest} = C - C_{prefix} = C \times (1 - \beta)$.

*The evaluation in our previous study[6] can exclude several other reasons, such as the freeing of the reserved space in the lazy segmentation.

- α: percentage of beginning part of objects, if cached, no startup delay is perceived by clients. [†].

Thus, ideally, for exponential segmentation, assuming the C_{prefix} can cache the first t objects' prefix segments, t must satisfy the following condition:

$$\sum_{i=1}^{i=t} L_i \times \alpha \leq C_{prefix} \quad and \quad \sum_{i=1}^{i=t+1} L_i \times \alpha > C_{prefix}. \tag{2}$$

Ideally, assuming the rest of the cache size C_{rest} can cache the first m object's remaining segments, m must satisfy the following condition:

$$\sum_{i=1}^{i=m} L_i \times (1-\alpha) \leq C_{rest} \quad and \quad \sum_{i=1}^{i=m+1} L_i \times (1-\alpha) > C_{rest}. \tag{3}$$

For lazy segmentation, no cache space is allocated separately to cache the initial segments of the object, thus, ideally, assuming the whole cache could be used to cache the first k objects according to the popularity, k must satisfy the following condition:

$$\sum_{i=1}^{i=k} L_i \leq C \quad and \quad \sum_{i=1}^{i=k+1} L_i > C. \tag{4}$$

To this end, we express the delayed start request ratios for exponential segmentation and lazy segmentation as follows:

$$P_{delay-E} = \frac{\sum_{i=t+1}^{i=n} \lambda_i}{\sum_{i=1}^{i=n} \lambda_i} \tag{5}$$

and

$$P_{delay-L} = \frac{\sum_{i=k+1}^{i=n} \lambda_i}{\sum_{i=1}^{i=n} \lambda_i}, \tag{6}$$

respectively.

Without considering the misses when the object is accessed for the first time, their corresponding byte hit ratios are:

$$P_{hit-E} = 1 - \frac{\sum_{i=t+1}^{i=n} \lambda_i \times L_i \times \alpha + \sum_{i=m+1}^{i=n} \lambda_i \times L_i \times (1-\alpha)}{\sum_{i=1}^{i=n} \lambda_i \times L_i} \tag{7}$$

and

$$P_{hit-L} = 1 - \frac{\sum_{i=k+1}^{i=n} \lambda_i \times L_i}{\sum_{i=1}^{i=n} \lambda_i \times L_i}. \tag{8}$$

respectively.

Having the general model and the byte hit ratio and the delayed start request ratio, we now start to find answers to **Q1** and **Q2** in the following by analyzing and comparing them thoroughly in different situations.

3.2. Performance Objective Analysis

In order to find the right answer to **Q1**, we analyze the performance objectives one by one based on the model we have built.

[†]Note that if instead of caching the first α percentage, caching a constant length of the prefix segment L_{prefix} of each object will lead to the same results. They are equivalent.

3.2.1. Delayed Start Request Ratio

Equation 5 and Equation 6 indicate that the relationship between t and k determines which technique performs better in terms of the delayed start request ratio.

Based on Equations 2 and 4, by comparing

$$\sum_{i=1}^{i=t} L_i \leq \frac{C \times \beta}{\alpha} \quad and \quad \sum_{i=1}^{i=k} L_i \leq C,$$

we can get that if $\frac{\beta}{\alpha} > 1$, it will lead to $t > k$. Through Equation 5 and Equation 6, $t > k$ means that the exponential segmentation has a better (less) delayed start request ratio. Otherwise, $t \leq k$, and lazy segmentation will perform better.

Exponential segmentation always caches beginning segments of all objects, which leads to $k < t$. Thus, in terms of the delayed start request ratio, exponential segmentation normally performs better than lazy segmentation.

3.2.2. Byte Hit Ratio

Based on Equation 7 and Equation 8, we can see the relationships among t, m, and k are deterministic to the byte hit ratio.

We now provide a thorough evaluation of all possible situations as follows.

- $m = t$:

  Equation 7 can be written as:

  $$P_{hit-E} = 1 - \frac{\sum_{i=t+1}^{i=n} \lambda_i \times L_i}{\sum_{i=1}^{i=n} \lambda_i \times L_i}. \tag{9}$$

  Compared with Equation 8, the problem once again comes to the relationship of t and k. If $t > k$, then exponential segmentation performs better. If $t < k$, lazy segmentation performs better.

  When $m = t$, there are m or t objects cached by the exponential segmentation totally. Thus, we can get $t = k$. Thus, lazy segmentation will perform the same as exponential segmentation in terms of the byte hit ratio.

- $m < t$:

  Equation 7 can be written as:

  $$P_{hit-E} = 1 - \frac{\sum_{i=t+1}^{i=n} \lambda_i \times L_i + \sum_{i=m+1}^{i=t} \lambda_i \times L_i \times (1-\alpha)}{\sum_{i=1}^{i=n} \lambda_i \times L_i}. \tag{10}$$

  Equation 8 can be written as:

  $$P_{hit-L} = 1 - \frac{\sum_{i=k+1}^{i=t} \lambda_i \times L_i + \sum_{i=t+1}^{i=n} \lambda_i \times L_i}{\sum_{i=1}^{i=n} \lambda_i \times L_i}. \tag{11}$$

  Thus, we must compare $\sum_{i=m+1}^{i=t} \lambda_i \times L_i \times (1-\alpha)$ and $\sum_{i=k+1}^{t} \lambda_i \times L_i$. If $\sum_{i=m+1}^{i=t} \lambda_i \times L_i \times (1-\alpha) > \sum_{i=k+1}^{t} \lambda_i \times L_i$, lazy segmentation performs better. Otherwise, exponential segmentation performs better.
  When $m < t$, there are m objects fully cached for exponential segmentation. Thus, for lazy segmentation, there are more objects fully cached and we can get that $k > m$. Therefore, we further analyze the case when $k > m$ and $t > m$ as follows.

  Since

  $$\sum_{i=m+1}^{i=t} \lambda_i \times L_i \times (1-\alpha) = (\sum_{i=m+1}^{i=k} \lambda_i \times L_i + \sum_{i=k+1}^{i=t} \lambda_i \times L_i) \times (1-\alpha) \tag{12}$$

and

$$\sum_{i=k+1}^{t} \lambda_i \times L_i = \sum_{i=k+1}^{t} \lambda_i \times L_i \times \alpha + \sum_{i=k+1}^{t} \lambda_i \times L_i \times (1-\alpha), \tag{13}$$

we must compare $\sum_{i=m+1}^{i=k} \lambda_i \times L_i \times (1-\alpha)$ and $\sum_{i=k+1}^{t} \lambda_i \times L_i \times \alpha$.

From Equations 2, 3, 4, we have

$$t = \frac{C}{L_{ave}^t} \times \frac{\beta}{\alpha}, \quad m = \frac{C}{L_{ave}^m} \times \frac{1-\beta}{1-\alpha}, \quad k = \frac{C}{L_{ave}^k}, \tag{14}$$

where L_{ave}^t denotes the average value of L_1 to L_t, L_{ave}^m denotes the average value of L_1 to L_m, and L_{ave}^k denotes the average value of L_1 to L_k.

Assume objects are of equal length, then $L_{ave}^t = L_{ave}^m = L_{ave}^k = L_{ave}$. (Note that this assumption simplifies the condition. However, we have proved it does not affect the conclusion we will draw. Details of the proof are omitted in this study.) Thus

$$t = k \times \frac{\beta}{\alpha}, m = k \times \frac{1-\beta}{1-\alpha}.$$

Since $\lambda_i \geq \lambda_{i+1}$, we get

$$\sum_{i=m+1}^{i=k} \lambda_i \times L_i \times (1-\alpha) \quad \geq \quad \sum_{i=m+1}^{i=k} \lambda_{k+1} \times L_i \times (1-\alpha) = \lambda_{k+1} k(\beta-\alpha) L_{ave} \tag{15}$$

and

$$\sum_{i=k+1}^{t} \lambda_i \times L_i \times \alpha \quad \leq \quad \sum_{i=k+1}^{t} \lambda_{k+1} \times L_i \times \alpha = \lambda_{k+1} k(\beta-\alpha) L_{ave}. \tag{16}$$

Based on Equations 15 and 16, it is clear that exponential segmentation performs worse in byte hit ratio when $m < t$. This confirms the performance comparisons in section 2.3.

- $m > t$:

  Equation 7 can be written as:

$$P_{hit-E} = 1 - \frac{\sum_{i=t+1}^{i=m} \lambda_i \times L_i \times \alpha + \sum_{i=m+1}^{i=n} \lambda_i \times L_i}{\sum_{i=1}^{i=n} \lambda_i \times L_i}. \tag{17}$$

  Equation 8 can be written as:

$$P_{hit-L} = 1 - \frac{\sum_{i=k+1}^{i=m} \lambda_i \times L_i + \sum_{i=m+1}^{i=n} \lambda_i \times L_i}{\sum_{i=1}^{i=n} \lambda_i \times L_i}. \tag{18}$$

  Thus, we must compare $\sum_{i=t+1}^{i=m} \lambda_i \times L_i \times \alpha$ and $\sum_{i=k+1}^{m} \lambda_i \times L_i$ (Note that if $k+1 > m$, $\sum_{i=k+1}^{m} \lambda_i \times L_i$ is defined as $-\sum_{i=m+1}^{k} \lambda_i \times L_i$). If $\sum_{i=t+1}^{i=m} \lambda_i \times L_i \times \alpha > \sum_{i=k+1}^{m} \lambda_i \times L_i$, exponential segmentation performs worse. Otherwise, lazy segmentation performs worse.

  When $m > t$, there are only t objects that are fully cached for exponential segmentation, so it must be $k > t$. Thus, the further analysis will be done when $m > t$ and $k > t$ as follows.

  Since

$$\sum_{i=t+1}^{i=m} \lambda_i \times L_i \times \alpha = \sum_{i=t+1}^{i=k} \lambda_i \times L_i \times \alpha + \sum_{i=k+1}^{i=m} \lambda_i \times L_i \times \alpha \tag{19}$$

and

$$\sum_{i=k+1}^{m} \lambda_i \times L_i = \sum_{i=k+1}^{m} \lambda_i \times L_i \times \alpha + \sum_{i=k+1}^{m} \lambda_i \times L_i \times (1-\alpha), \tag{20}$$

it comes to compare $\sum_{i=t+1}^{i=k} \lambda_i \times L_i \times \alpha$ and $\sum_{i=k+1}^{m} \lambda_i \times L_i \times (1-\alpha)$.
Since $\lambda_i \geq \lambda_{i+1}$, with the same assumption as before that objects are of same length, it is easy to get

$$\sum_{i=t+1}^{i=k} \lambda_i \times L_i \times \alpha \quad \geq \quad \sum_{i=t+1}^{i=k} \lambda_{k+1} \times L_i \times \alpha = (\alpha-\beta)k\lambda_{k+1}L_{ave} \tag{21}$$

and

$$\sum_{i=k+1}^{m} \lambda_i \times L_i \times (1-\alpha) \quad \leq \quad \sum_{i=k+1}^{m} \lambda_{k+1} \times L_i \times (1-\alpha) = (\alpha-\beta)k\lambda_{k+1}L_{ave}. \tag{22}$$

Based on Equations 21 and 22, we can get that when $m > t$, lazy segmentation performs worse in terms of byte hit ratio under these assumptions. However, $m > t$ leads to $k > t$, recall the analysis conclusion in 3.2.1, when $k > t$, lazy segmentation will perform better in terms of the delayed start request ratio.

In reality, exponential segmentation always caches all objects' beginning segments, thus, $m > t$ is always true.

The results of the above analysis show that the performance of segment-based caching strategies is always a trade-off between the byte hit ratio and the delayed start request ratio. They are affected by the relationships of t, m, which are determined by α, β, n and L_{ave}. (Note that for the lazy segmentation strategy, in the sense that we do not reserve a part of the cache space for caching the beginning segments of objects, $\beta = 0$; however, if counting the cache space used for caching the beginning segments of objects, a dynamically changing non-zero β is used. For the prefix caching, β is set to 100%.) Based on the analyis results, if m is decreased, the achieved byte hit ratio is reduced. However, the decrease of m leads to decrease of t, which results in a reduced delayed start request ratio. This seems to indicate that we can use the byte hit ratio to trade for delayed start request ratio. Whether this is true or not is critical to **Q3**. And if this is true, how to evaluate the overall performance of any strategy so that they are comparable is interesting as **Q2** states. We will answer these questions heuristically after we derive the performance bounds for each performance objective.

3.3. Performance Bound Analysis

We have learned that these two performance objectives are always a trade-off between each other. However, how much performance can be optimized is not answered yet. In this section, we will give performance bounds based on the model so that they can guide the performance optimization under certain conditions as our assumptions state. We also assume the objects are of equal length as before. That is, $L_{ave}^t = L_{ave}^m = L_{ave}^k = L_{ave}$.

3.3.1. Delayed Start Request Ratio

For the exponential segmentation strategy, substituting Equation 1 in Equation 5, we get

$$P_{delay-E} = \frac{\sum_{i=t+1}^{i=n} \lambda \times \frac{\eta \times \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \eta \times \frac{1}{i^\theta}}}{\sum_{i=1}^{i=n} \lambda \times \frac{\eta \times \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \eta \times \frac{1}{i^\theta}}} = \frac{\sum_{i=t+1}^{i=n} \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \frac{1}{i^\theta}}. \tag{23}$$

Carefully using the series theory and integration on Equation 23,

- $\theta = 1$

$$P_{delay-E} = \frac{\sum_{i=t+1}^{i=n} \frac{1}{i}}{\sum_{i=1}^{i=n} \frac{1}{i}} \leq \frac{\int_t^n \frac{1}{i} di}{\int_1^{n+1} \frac{1}{i} di} = \frac{\ln n - \ln t}{\ln (n+1)}$$

and

$$P_{delay-E} = \frac{\sum_{i=t+1}^{i=n} \frac{1}{i}}{\sum_{i=1}^{i=n} \frac{1}{i}} \geq \frac{\int_{t+1}^{n+1} \frac{1}{i} di}{1 + \int_{1}^{n} \frac{1}{i} di} = \frac{\ln \frac{n+1}{t+1}}{1 + \ln n}.$$

Having $t = \frac{C}{L_{ave}} \times \frac{\beta}{\alpha}$, $U = \frac{C}{L_{ave}}$, we have

$$P_{delay-E}^{Max} = \frac{\ln n - \ln U \times \frac{\beta}{\alpha}}{\ln (n+1)} \tag{24}$$

and

$$P_{delay-E}^{Min} = \frac{\ln (n+1) - \ln (U \times \frac{\beta}{\alpha} + 1)}{1 + \ln n}. \tag{25}$$

For Equation 24 and 25, the larger the value of β, the smaller the values of $P_{delay-E}^{Max}$ and $P_{delay-E}^{Min}$, and the smaller the value of β, the larger the values of $P_{delay-E}^{Max}$ and $P_{delay-E}^{Min}$.

- $\theta \neq 1$

$$P_{delay-E} = \frac{\sum_{i=t+1}^{i=n} \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \frac{1}{i^\theta}} \leq \frac{\int_{t}^{n} \frac{1}{i^\theta} di}{\int_{1}^{n+1} \frac{1}{i^\theta} di} = \frac{n^{1-\theta} - t^{1-\theta}}{(n+1)^{1-\theta} - 1},$$

and

$$P_{delay-E} = \frac{\sum_{i=t+1}^{i=n} \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \frac{1}{i^\theta}} \geq \frac{\int_{t+1}^{n+1} \frac{1}{i^\theta} di}{1 + \int_{1}^{n} \frac{1}{i^\theta} di} = \frac{(n+1)^{1-\theta} - (t+1)^{1-\theta}}{n^{1-\theta} - \theta}.$$

Having $t = \frac{C}{L_{ave}} \times \frac{\beta}{\alpha}$, $U = \frac{C}{L_{ave}}$, we have

$$P_{delay-E}^{Max} = \frac{n^{1-\theta} - (U \times \frac{\beta}{\alpha})^{1-\theta}}{(n+1)^{1-\theta} - 1} \tag{26}$$

and

$$P_{delay-E}^{Min} = \frac{(n+1)^{1-\theta} - (U \times \frac{\beta}{\alpha} + 1)^{1-\theta}}{n^{1-\theta} - \theta}. \tag{27}$$

$t^{1-\theta}$ is an *increasing* function when $0 < \theta < 1$, and a *non-increasing* function when $\theta > 1$. Thus, the larger the value of β, the smaller the values of $P_{delay-E}^{Max}$ and $P_{delay-E}^{Min}$, and the smaller the value of β, the larger the values of $P_{delay-E}^{Max}$ and $P_{delay-E}^{Min}$.

For the lazy segmentation strategy, substituting Equation 1 in Equation 6, we get

$$P_{delay-L} = \frac{\sum_{i=k+1}^{i=n} \lambda \times \frac{\eta \times \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \eta \times \frac{1}{i^\theta}}}{\sum_{i=1}^{i=n} \lambda \times \frac{\eta \times \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \eta \times \frac{1}{i^\theta}}} = \frac{\sum_{i=k+1}^{i=n} \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \frac{1}{i^\theta}}. \tag{28}$$

Carefully using the series theory and integration on Equation 28,

- $\theta = 1$

$$P_{delay-L}^{Max} = \frac{\ln n - \ln U}{\ln (n+1)} \tag{29}$$

and

$$P_{delay-L}^{Min} = \frac{\ln(n+1) - \ln(U+1)}{1 + \ln n}. \tag{30}$$

- $\theta \neq 1$

$$P_{delay-L}^{Max} = \frac{n^{1-\theta} - (U)^{1-\theta}}{(n+1)^{1-\theta} - 1} \tag{31}$$

and

$$P_{delay-L}^{Min} = \frac{(n+1)^{1-\theta} - (U+1)^{1-\theta}}{n^{1-\theta} - \theta}. \tag{32}$$

Equations 24, 26 and Equations 29, 31 give the upper bounds for the exponential segmentation and the lazy segmentation strategies, respectively, with different θ conditions. Equations 25, 27 and Equations 30, 32 give lower bounds for them in the ideal situation.

3.3.2. Byte Hit Ratio

For exponential segmentation, based on Equation 7, substituting the λ_i from Equation 1, we get

$$P_{hit-E} = \quad 1 - \frac{\alpha \times \sum_{i=t+1}^{i=n} \frac{1}{i^\theta} + (1-\alpha) \times \sum_{i=m+1}^{i=n} \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \frac{1}{i^\theta}}. \tag{33}$$

Carefully using the series theory and integration on Equation 33,

- $\theta = 1$

$$P_{hit-E}^{Max} = 1 - \frac{\alpha \times \ln \frac{n+1}{U \times \frac{\beta}{\alpha} + 1} + (1-\alpha) \times \ln \frac{n+1}{U \times \frac{1-\beta}{1-\alpha} + 1}}{1 + \ln n} \tag{34}$$

and

$$P_{hit-E}^{Min} = 1 - \frac{\alpha \times \ln \frac{n}{U \times \frac{\beta}{\alpha}} + (1-\alpha) \times \ln \frac{n}{U \times \frac{1-\beta}{1-\alpha}}}{\ln(n+1)}. \tag{35}$$

- $\theta \neq 1$

$$P_{hit-E}^{Max} = 1 - \left(\frac{(n+1)^{1-\theta} - \alpha \times (\frac{\beta}{\alpha} \times U + 1)^{1-\theta}}{n^{1-\theta} - \theta} - \frac{(1-\alpha) \times (\frac{(1-\beta)}{(1-\alpha)} \times U + 1)^{1-\theta}}{n^{1-\theta} - \theta}\right) \tag{36}$$

and

$$P_{hit-E}^{Min} = 1 - \left(\frac{n^{1-\theta} - \alpha \times (\frac{\beta}{\alpha} \times U)^{1-\theta}}{(n+1)^{1-\theta} - 1} - \frac{(1-\alpha) \times (\frac{(1-\beta)}{(1-\alpha)} \times U)^{1-\theta}}{(n+1)^{1-\theta} - 1}\right). \tag{37}$$

For lazy segmentation, based on Equation 8, substituting the λ_i from Equation 1, we get

$$P_{hit-L} = 1 - \frac{\sum_{i=1}^{i=n} \frac{1}{i^\theta} - \sum_{i=1}^{i=k} \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \frac{1}{i^\theta}} = \frac{\sum_{i=1}^{i=k} \frac{1}{i^\theta}}{\sum_{i=1}^{i=n} \frac{1}{i^\theta}} \tag{38}$$

Carefully using the series theory and integration on Equation 38,

- $\theta = 1$

$$P_{hit-L}^{Max} = \frac{1 + \ln U}{\ln(n+1)} \tag{39}$$

and

$$P_{hit-L}^{Min} = \frac{\ln(U+1)}{1 + \ln n}. \tag{40}$$

- $\theta \neq 1$

$$P_{hit-L}^{Max} = \frac{U^{1-\theta} - \theta}{(n+1)^{1-\theta} - 1} \tag{41}$$

and

$$P_{hit-L}^{Min} = \frac{(U+1)^{1-\theta} - 1}{n^{1-\theta} - \theta}. \tag{42}$$

Equations 34, 36 and Equations 39, 41 give the upper bounds for the exponential segmentation and the lazy segmentation strategies, respectively, with different θ conditions. Equations 35, 37 and Equations 40, 42 give the lower bounds for the ideal situation.

It is important to note that these upper and lower bounds are based on the model we built for ideal situations. Thus, the upper bounds we found here are valid upper bounds for general situations, while the lower bounds are only valid for ideal situations.

3.4. A New Overall Performance Objective

Based on the modeling, we understand the inherent trade-off between the two performance objectives, which wakes it difficult to evaluate the overall performance of a strategy. In this section we will provide a way to overcome this difficulty by defining a new performance metric, *weighted miss rate*, which flexibly combines the byte hit ratio and the delayed start request ratio together.

We use P_{Hit} and P_{Delay} to represent the byte hit ratio and the delayed start request ratio the strategy can achieve. They can be considered as that among the first α percentage of the data of the objects, there is P_{delay} percentage that can not be delivered from the proxy on average. Among the remaining $1-\alpha$ percentage of the data of the objects, there is $P_{Hit} - \alpha \times (1 - P_{Delay})$ percentage of the data cached in the proxy. Thus, $1 - (P_{Hit} - \alpha \times (1 - P_{Delay}))$ represents the uncached percentage that can not be delivered from the proxy. Assigning different weights to the two parts as w_a and w_b ($w_a + w_b = 1$), we have the weighted miss rate defined as follows:

$$P_{WMR} = w_a \times (1 - (P_{Hit} - \alpha \times (1 - P_{Delay}))) + w_b \times (P_{Delay}). \tag{43}$$

Equation 43 implies that the smaller P_{Delay} is, the higher P_{Hit} is and the smaller P_{WMR} is. Thus, the strategy design and optimization objective is to minimize the weighted miss rate. With the same assumptions as before, w_a and w_b can be set to be the same when the startup delay and the delay caused by cache miss from the later segments are of the same importance. Specifically for the exponential and lazy segmentation strategies, the minimum weighted miss rates thus are

$$\frac{\alpha}{2} + \frac{(2-\alpha) \times (n+1)^{1-\theta} - (\frac{\beta \times C + \alpha \times L_{ave}}{\alpha \times L_{ave}})^{1-\theta}}{2 \times n^{1-\theta} - 2 \times \theta} - \frac{(1-\alpha) \times (\frac{(1-\beta) \times C}{(1-\alpha) \times L_{ave}} + 1)^{1-\theta}}{2 \times n^{1-\theta} - 2 \times \theta},$$

and

$$\frac{\alpha}{2} + \frac{2-\alpha}{2} \times \frac{(n+1)^{1-\theta} - (\frac{C}{L_{ave}} + 1)^{1-\theta}}{n^{1-\theta} - \theta}, \tag{44}$$

when $\theta \neq 1$.

4. REVISIT ADAPTIVE-LAZY SEGMENTATION STRATEGY

Having obtained the answers to **Q1** and **Q2**, and being motivated by them, we now try to answer **Q3**. We want to know if the trade-off between the two performance objectives can be leveraged to get both a good byte hit ratio and a low delayed start request ratio. For this purpose, we re-design the adaptive-lazy segmentation strategy with a heuristic replacement policy and evaluate it using different workloads in this section.

4.1. Three-phase Iterative Replacement Policy

We have learned that the adaptive-lazy segmentation strategy has no reserved space for caching the beginning segments of objects, which is equivalent to have a dynamically changing β if counting all the space occupied by the beginning segments of different objects. If we set a fixed and proper β, it is possible to get a good balance between the byte hit ratio and the delayed start request ratio. However, if a fixed β is set, the performance will be workload-dependent since it is not realistic to assume a known number of objects a proxy would serve *a priori*.[‡]

To keep the adaptiveness while still achieving good performance results, a three-phase iterative replacement policy is designed heuristically. We expect a more accurate (though dynamically changing) β can be set through the three-phase iterative replacement policy. We first define the startup length and the utility function of an object.

The *startup length* of an object is defined to be the first α percentage of the data of the object that, if cached by the proxy, produces no startup delay. Note that α can vary depending on the bandwidth between the proxy and the client. For a low bandwidth link, α can generally be set larger. For a high bandwidth link, α can generally be set smaller.

The caching utility function is defined as follows (similar to that in Ref.[6]):

$$\frac{F \times \frac{L_{sum}}{n_a} \times \min(1, \frac{\frac{T_r - T_1}{n_a}}{T_c - T_r})}{n_s \times L_b}. \tag{45}$$

Several factors have been considered in this utility function:
(1) the average duration of accesses $\frac{L_{sum}}{n_a}$;
(2) the access frequency F (defined as before);
(3) the length of the cached data $n_s \times L_b$, which is the cost of the storage;
(4) the probability of the future access $\min(1, \frac{\frac{T_r - T_1}{n_a}}{T_c - T_r})$.

The replacement policy considers the possibility of future accesses as follows: it compares $T_c - T_r$, the time interval between now and the most recent access, and $\frac{T_r - T_1}{n_a}$, the average time interval for an access happening in the past. If $T_c - T_r > \frac{T_r - T_1}{n_a}$, the possibility that a new request will arrive soon for this object is small. Otherwise, it is highly possible that a request will come soon.

The three-phase iterative replacement policy works as follows. Upon object admission, if there is insufficient cache space, the caching utility of each currently cached object is calculated. The object with the smallest utility is chosen as the victim object, and partial cached data of this object is evicted in one of the following three phases.

- First Phase: If the object is fully cached, the object is segmented by the lazy segmentation method. The first 2 segments are kept and the rest of the segments are evicted right after the segmentation is completed.
- Second Phase: If the object is partially cached with more than 1 segment, the last cached segment of this object is evicted.
- Third Phase: If the victim has only the first segment and is to-be-replaced, then its startup length and the base segment length is compared.
  - If its startup length is less than the base segment length, the startup length is kept and the rest is replaced. Since the portion being kept has a small storage size, its caching utility value will be increased significantly, and the object has a much less chance to be chosen by the replacement policy as victim again soon. That is to say, the object has a better chance to stay in the cache to lower the delayed start request ratio. Depending on the value of its utility, the startup length of an object could still be replaced if it is smaller than any cached object's caching utility.

[‡]If we try to refine the exponential segmentation by setting up a more proper but fixed β, the same problems exist.

– If the startup length is greater than or equal to the base segment length, it will be replaced totally.

The utility value of the object is updated after each replacement and this process repeats iteratively until the demanded space is found.

With the three-phase replacement policy, the disk cache is automatically divided into two portions. One portion is used to cache the segments of the objects. The other portion is used to cache the startup length of the objects. Though the cache size is fixed, the size of each portion is varying. The disk thus becomes a flexible 2-level cache. The size of each level dynamically changes according to reality. Thus, in the running time, the new adaptive-lazy segmentation strategy has an ever-changing β value and could be prefix caching or adaptive-lazy segmentation or exponential segmentation at different time periods. So it combines their merits dynamically. Since each object goes through at most three phases before it is fully replaced, this policy is called three-phase replacement policy.

4.2. Evaluation of Three-phase Adaptive-lazy Segmentation Strategy

Having designing the new heuristic strategy, we want to evaluate if the results are as our expectation.

4.2.1. Workload Summary

To evaluate the performance, we conduct simulations based on both synthetic workloads and a real workload extracted from enterprise media server logs. Three synthetic workloads are designed. The first simulates accesses to media objects in the Web environment in which the length of the video varies from short ones to longer ones. The second simulates video access in a video-on-demand (VOD) environment in which only longer streams are served. Both workloads assume full viewing client sessions. We use WEB and VOD as the name of these workloads. These workloads assume a Zipf-like distribution ($p_i = f_i / \sum_{i=1}^{N} f_i, f_i = 1/i^{\theta}$) for the popularity of the media objects. They also assume request inter arrival to follow Poisson distribution ($p(x, \lambda) = e^{-\lambda} \times (\lambda)^x / (x!), x = 0, 1, 2...$).

In Web access case, client accesses to video objects may be incomplete, that is, a session may terminate before the full media object is delivered. We simulate this behavior by designing a partial viewing workload based on the WEB workload. We use PART as the name of the workload. In this workload, 80% of the sessions terminate before 20% of the object is delivered.

For the real workload named REAL, we obtained logs from HP Corporate Media Solutions covering the period from April 1 to April 10, 2001. There are a total of 403 objects, and the unique object size accounts to 20G. There are 9000 requests, which run for 916427 seconds, roughly 10 days. No Poisson or Zipf distribution is used. Our analysis shows that 83% requests only view the objects for less than 10 minutes and 56% requests only view the objects for less than 10%. Only about 10% requests view the whole objects.

Table 1 lists some known properties of the synthetic workloads and the real workload.

Workload Name	Num of Request	Num of Object	Size (GB)	λ	θ	Range (minute)	Duration (day)
WEB	15188	400	51	4	0.47	2-120	1
VOD	10731	100	149	60	0.73	60-120	7
PART	15188	400	51	4	0.47	2-120	1
REAL	9000	403	20	-	-	6 - 131	10

Table 1. The Workload Summary

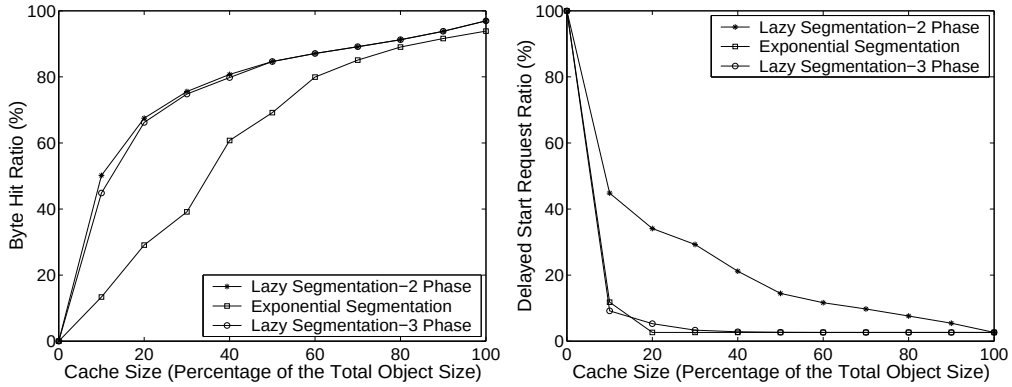

Figure 1. WEB: Byte Hit Ratio & Delayed Start Request Ratio

4.2.2. Evaluation on Complete Viewing Workloads

In the figures, *Lazy Segmentation-2 Phase* represents the adaptive-lazy segmentation strategy with two phase replacement policy as we proposed in,[6] while *Lazy Segmentation-3 Phase* represents the redesigned strategy with three phase replacement policy. *Exponential Segmentation* denotes the exponential segmentation strategy. For exponential segmentation, the client will not experience a startup delay if the first 6 segments are cached in the proxy. For a fair comparison, the same length is used in the other two strategies.

Figure 1 shows the performance results using the WEB workload. As shown in Figure 1(a), compared with the lazy segmentation-2 phase, the lazy segmentation-3 phase achieves a little lower byte hit ratio when the cache size is of 10%-50% of the total object size. However, Figure 1(b) shows that it achieves a much lower delayed start request ratio, very close to that of the exponential segmentation, except when the cache size is 10%-40%.

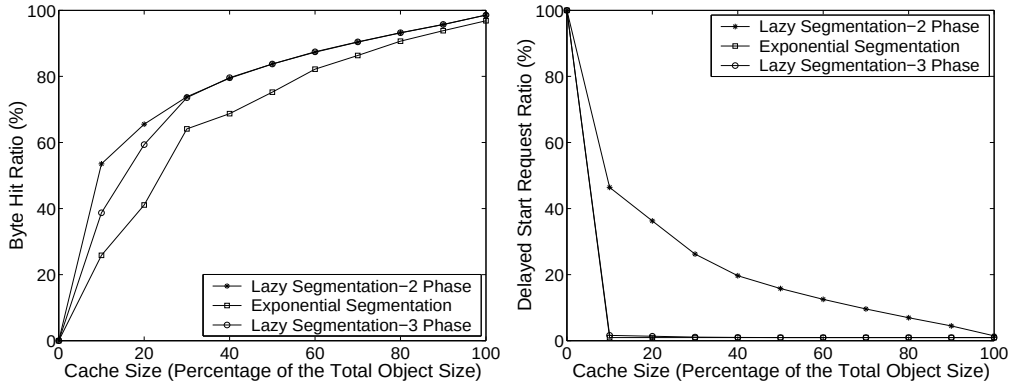

Figure 2. VOD: Byte Hit Ratio & Delayed Start Request Ratio

Figure 2 shows the performance results based on the VOD workload. Trends similar to those in Figure 1 occur in Figure 2. Due to the average longer accessing time of the VOD sessions, Figure 2(b) shows a very close result of the delayed start request ratio to that of exponential segmentation.

4.2.3. Evaluation on Partial Viewing Workloads

Figure 3 shows the performance results of the PARTIAL workload. It shows similar trends to those of WEB. Due to the 80% prematurely terminated sessions, the performance gaps are larger when compared with the lazy segmentation-2 phase for the byte hit ratio and with exponential segmentation for the delayed start request ratio.

Performance results using the REAL workload are shown in Figure 4. Figure 4(a) shows very close results on the byte hit ratio for lazy segmentation-3 phase and lazy segmentation-2 phase. This is due to the fact that a

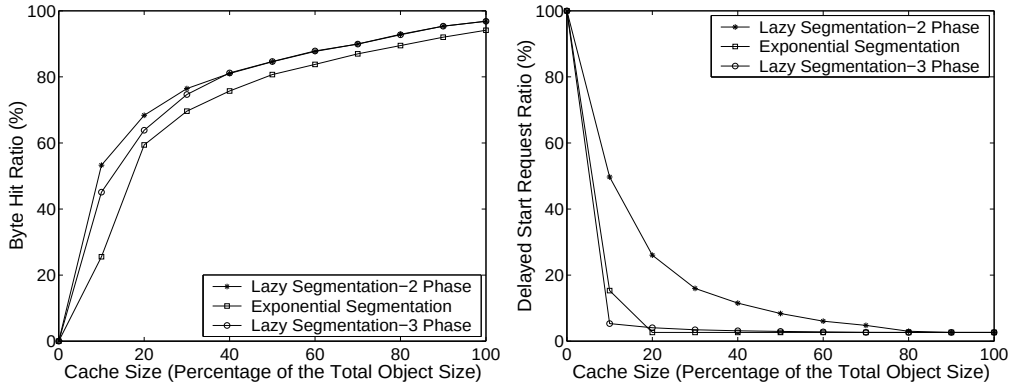

Figure 3. PART: Byte Hit Ratio & Delayed Start Request Ratio

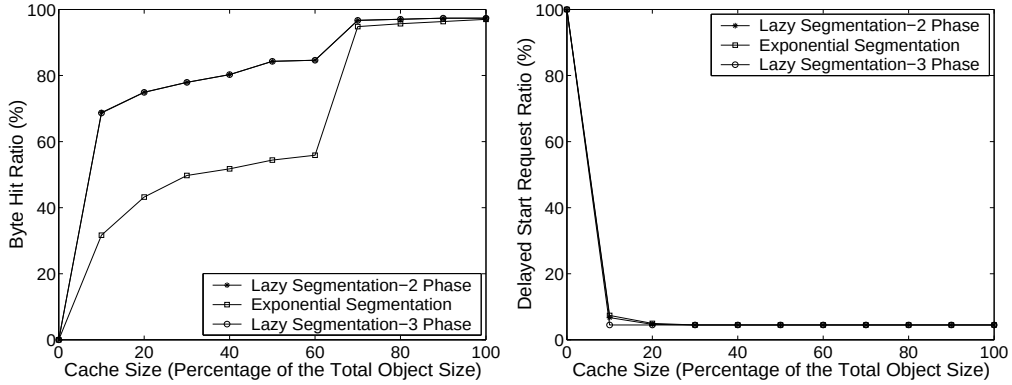

Figure 4. REAL: Byte Hit Ratio & Delayed Start Request Ratio

lot of sessions are terminated earlier, thus a lot of space can be used to cache the frequently accessed beginning segments of different objects. Figure 4(b) shows that for the delayed start request ratio, lazy segmentation-3 phase achieves a better result than that of lazy segmentation-2 phase. Having evaluated the workloads extracted from the HP Media logs in different periods, we found that lazy segmentation-3 phase can achieve a much larger performance gain. Therefore, Figure 4 indicates that even in a situation that does not favor the lazy segmentation-3 phase strategy, our proposed strategy can still outperform other strategies.

So far our heuristic approach to trade a little reduction of the byte hit ratio for a large decrease of the delayed started request ratio is thus verified to be effective.

4.2.4. Evaluation Using Weighted Miss Rate

As shown in Figure 5, the lazy segmentation-3 phase always achieves the minimum weighted miss rate. For the WEB, VOD and PART workloads, the weighted miss rate achieved by lazy segmentation-2 phase is closer to exponential segmentation than lazy segmentation-3 phase. For the REAL workload, it is close to the lazy segmentation-3 phase due to the large amount of early terminated sessions. This indicates that a lot of performance gain on the byte hit ratio of lazy segmentation-2 phase is offset by the deterioration of the delayed start request ratio. These results consistently confirm the effectiveness of our proposed overall performance metric.

5. OTHER RELATED WORK

Proxy caching of streaming media has been explored in Ref. 1–4, 10–19. Prefix caching and its protocol consideration as well as partial sequence caching are studied in Ref. 1, 10, 20 . In video staging,[12] a portion of bits from the video frames whose size is larger than a predetermined threshold is cut off and prefetched to the proxy to reduce the bandwidth on the server proxy channel. In Ref. 2, 3, 15, a similar idea is proposed for caching scalable

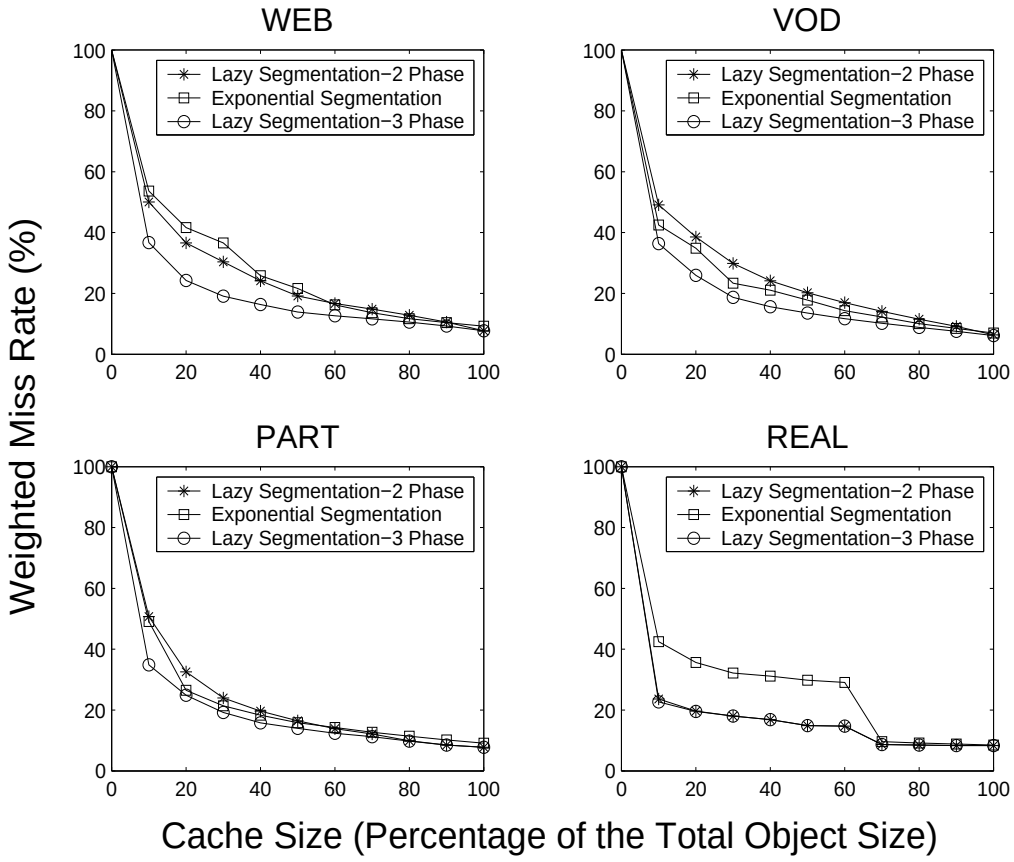

Figure 5. Weighted Miss Rate

videos that co-operates with the congestion control mechanism. In Ref. 14, the proposed approach attempts to select groups of consecutive frames by the selective caching algorithm, while in Ref. 21, the algorithm may select groups of non-consecutive frames for caching in the proxy. The caching problem for layered encoded video is studied in Ref. 16. The cache replacement of streaming media is studied in Ref. 18, 19.

6. CONCLUSION

Server traffic reduction and client startup delay are two different performance objectives of proxy caching in streaming media delivery systems. Current segment-based proxy caching approaches rarely achieve good performance on both fronts. In this paper, through model-driven analysis on two types of existing segment-based proxy caching strategies, we have analytically illustrated the trade-offs between these two objectives. To provide a common base to evaluate different strategies with these two conflicting performance objectives, a comprehensive performance metric is proposed. To verify the effectiveness of our proposed metric and to explore the performance trade-off, a heuristic adaptive-lazy segmentation strategy is restructured with a heuristic three-phase iterative policy for segment replacement. Comprehensive simulations using both synthetic and real stream media workloads have been conducted. The performance results confirmed our analysis and understanding of the trade-off.

ACKNOWLEDGMENTS

This work is supported by NSF grants CCR-0098055 and ACI-0129883, and a grant from Hewlett-Packard Laboratories. We thank the constructive comments from the anonymous referees.

REFERENCES

1. S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proceedings of IEEE INFOCOM*, March 1999.
2. R. Rejaie, M. Handley, H. Yu, and D. Estrin, "Proxy caching mechanism for multimedia playback streams in the internet," in *Proceedings of International Web Caching Workshop*, March 1999.
3. R. Rejaie, M. H. H. Yu, and D. Estrin, "Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet," in *Proceedings of IEEE INFOCOM*, March 2000.
4. K. Wu, P. S. Yu, and J. Wolf, "Segment-based proxy caching of multimedia streams," in *Proceedings of WWW*, May 2001.
5. Y. Chae, K. Guo, M. Buddhikot, S. Suri, and E. Zegura, "Silo, rainbow, and caching token: Schemes for scalable fault tolerant stream caching," in *IEEE Journal on Selected Areas in Communications, Special Issue on Internet Proxy Services*, Sept. 2002.
6. S. Chen, B. Shen, S. Wee, and X. Zhang, "Adaptive and lazy segmentation based proxy caching for streaming media delivery," in *Proceedings of ACM NOSSDAV*, June 2003.
7. B. Wang, S. Sen, M. Adler, and D. Towsley, "Proxy-based distribution of streaming video over unicast/multicast connections," in *Proceedings of IEEE INFOCOM*, June 2002.
8. L. Cherkasova and M. Gupta, "Characterizing locality, evolution, and life span of accesses in enterprise media server workloads," in *Proceedings of ACM NOSSDAV*, May 2002.
9. M. Chesire, A. Wolman, G. Voelker, and H. Levy, "Measurement and analysis of a streaming media workload," in *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, March 2001.
10. M. Y. Chiu and K. H. Yeung, "Partial video sequence caching scheme for vod systems with heteroeneous clients," in *Proceedings of the 13th International Conference on Data Engineering*, April 1997.
11. S. Acharya and B. Smith, "Middleman: A video caching proxy server," in *Proc. of ACM NOSSDAV*, June 2000.
12. Z. Zhang, Y. Wang, D. Du, and D. Su, "Video staging: A proxy-server based approach to end-to-end video delivery over wide-area networks," in *IEEE Transactions on Networking*, Aug. 2000.
13. 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.
14. W. Ma and H. Du, "Reducing bandwidth requirement for dilivering video over wide area networks with proxy server," in *Proceedings of International Conferences on Multimeida and Expo.*, July 2000.
15. R. Rejaie, M. Handely, and D. Estrin, "Quality adaptation for congestion controlled video playback over the internet," in *Proceedings of ACM SIGCOMM*, Sept. 1999.
16. J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross, "Distributing layered encoded video through caches," in *Proceedings of IEEE INFOCOM*, April 2001.
17. S. Lee, W. Ma, and B. Shen, "An interactive video delivery and caching system using video summarization," in *Computer Communications*, **25**, pp. 424–435, March 2002.
18. R. Tewari, A. D. H. Vin, and D. Sitaram, "Resource-based caching for web servers," in *Proceedings of SPIE Conference on Multimedia Computing and Networking*, Jan. 1998.
19. M. Reisslein, F. Hartanto, and K. W. Ross, "Interactive video streaming with proxy servers," in *Proceedings of the First International Workshop on Intelligent Multimedia Computing and Networking*, Feb. 2000.
20. S. Gruber, J. Rexford, and A. Basso, "Protocol considertations for a prefix-caching for multimedia streams," in *Computer Network*, **33(1-6)**, pp. 657–668, June 2000.
21. Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints," in *IEEE Journal on Selected Areas in Communications*, Sept. 2002.