Video Staging in Video Streaming Proxy Server

W.K. CHEUK and D.P.K. LUN
Center of Multimedia Signal Processing
Department of Electronic and Information Engineering
The Hong Kong Polytechnic University, Hong Kong
Ph: (852) 2766 4365 Fax: (852) 2362 8439 Email: enpklun@polyu.edu.hk

Abstract

Video staging has been proposed as a mechanism for video streaming proxy servers to better exploit the temporal locality of requests made by clients. It divides a video stream into two parts, i.e. local storage and remote access. It opens a new possibility of scheduling in video streaming proxy server with the introduction of disk scheduling on top of network scheduling. However, a simple division cannot put all these resources into good use. In this paper, we devise a novel video staging algorithm. It considers memory consumption, network bandwidth and disk access requirement so that these resources usage can be optimized accordingly.

Index Terms: Video Streaming Proxy Server, Video Staging and Video Smoothing

1. Introduction

In recent years, video streaming has become a hot topic in the computing society. It involves large-scale audiovisual data transfer in the network with real time requirement. Due to the temporal locality effect, it is known that most data requested by different clients are similar within a short time frame. Rather than replicating the transmission from the media server, we can replicate the data in the proximity of those clients. Thus, we can save the network bandwidth requirement for the media server. Video streaming proxy server is developed to fulfill this purpose.

Traditional web proxies tend to store up whole piece of data. However, Zhang et al. [1][4] pointed out that this is not necessary in video streaming and proposed the video staging technique. At any time when a video is streaming through a video staging enabled proxy server, the server will first determine whether this video has to be stored. To store the stream, the stream will be divided into two parts by setting a bandwidth threshold $TH$ as shown in Figure 1. The lower part is bandwidth limited and is supposed to be obtained from the media server through a network channel with bounded bandwidth. The upper part contains sporadic component of the video and is stored in the proxy server; it is supposed to be obtained from the local disk upon future requests. Therefore, video staging can, on the first hand, exploit the temporal locality of the requested video streams to achieve backbone network bandwidth saving. On the other hand, it can smooth the delivery of video stream to suit current Internet topology.

To achieve video staging, we require continuous disk access to retrieve the upper part of the stream. However, disk access will become very inefficient for multiple requests within a short period. Since the requests of disk access are served one by one in a time-multiplexed manner, when this multiplexing cycle is too short, the throughput of the disk drops due to the disk access overhead. The solution is to expand the multiplexing cycle but then the system has to reserve a larger memory buffer to handle the increased amount of prefetched data. Consequently, memory buffer becomes an important parameter in the scheduling algorithm if an optimization of the overall system performance is of concern.

In this paper, we propose a new video staging algorithm that optimally utilizes the resources such as, network bandwidth, disk bandwidth and memory buffer. It allows a video stream to be divided into two parts with one of them being disk access friendly and the other one being network access friendly. Simulation results show that the proposed algorithm allows us to reduce the number of disk accesses as compared with the traditional video staging algorithm. It is achieved by better utilizing the network resource. Moreover, a unique resource allocation table can be obtained for each video stream. Proxy servers can base on these tables to adjust its resource allocation in order to achieve an optimal resource allocation scheme.

![Figure 1: Video Staging](image)
2. Video Staging Algorithm

Our algorithm is constructed based on the video smoothing algorithm proposed by Rexford et al. [2][3][5]. This algorithm tries to construct a bandwidth-limited schedule from the delivery schedule of a video stream. Due to this bandwidth restriction, memory buffer is required in order to store up the data for the upcoming need. As shown by Rexford, data transfer of a connection within a proxy server can be illustrated by a graph plotting its cumulative transferred data against time. As shown in Figure 2. Line A illustrates the arrival schedule that addresses the maximum possible data cumulated from the media server at the proxy. Line D shows the client data deadline schedule, illustrating the least amount of data the client has to receive in order to provide a correct playout on the client side. Between these lines, line S illustrates the actual delivery schedule for this connection in the proxy.

Figure 2: Cumulative data transfer of a connection in a proxy server

We will apply Rexford’s algorithm in a slightly different scenario and call it as algorithm \( \Phi \) in the following context. It will work under a limited incoming bit-rate \( r \) and a delivery schedule \( S = S_k : k = 0,\ldots,N \) and generate a suggested schedule \( f = f_k : k = 0,\ldots,N \) for the server to deliver the video stream to the proxy. It is shown as the arrival schedule A as viewed from the proxy side in Figure 2. The algorithm is stated as follows:

Figure 3: Algorithm \( \Phi \)

1. Assign \( f_0 = S_0 \)
2. Begin at the end of the schedule with \( k = N \)
3. If \( f_k - S_{k-1} \leq r \) then 4 else 6
4. \( f_k = S_{k-1} \)
5. Iterate backward by \( k = k - 1 \), goto 7
6. \( f_k = f_k - r \)
7. If scheduling completed \( k = 0 \) then 8 else 3
8. End
9. If scheduling completed \( k = 0 \) then 3 else 10
10. End

It can be shown that [6] the relationship between incoming bandwidth and memory buffer size under algorithm \( \Phi \) is as follows:

**Theorem 1:**

If there are two schedules \( f = \Phi(D,r) \) and \( f' = \Phi(D,r') \) with two different incoming bandwidths such that \( r' > r \), then the memory requirement of \( f_k \) is greater than or equal to \( f_k' \) and thus \( \max \{ f_k - D_k \} \geq \max \{ f_k' - D_k \} \).

Moreover, it can be shown that [6] \( \Phi \) always minimizes the maximum memory buffer requirement as well.

**Theorem 2:**

If there are two schedules \( f = \Phi(D,r) \) and \( f' = \Phi(D,r') \) with the same incoming bit-rate \( r \) then the memory requirement of \( f \) will be smaller or equal to \( f_k' \).

As a result, we conclude that algorithm \( \Phi \) can minimize its memory buffer usage with a given bandwidth. It gives an optimal schedule in terms of memory buffer size.

However, the network is the only transportation medium considered in Rexford’s algorithm. It does not consider the use of proxy server local storage. By incorporating the mechanism of video staging into Rexford’s algorithm, we develop an efficient video staging smoothing algorithm denoted as \( \Gamma \).

Assume the network bandwidth constraint is \( r \) and the maximum proxy buffer size allocated to support this video stream is \( b \) then the proposed algorithm \( \Gamma \) with \( g = \Gamma(D,r,b) \) is stated as follows:

Figure 4: Algorithm \( \Gamma \)

1. Assign \( g_0 = S_0 \)
2. Begin at the end of the schedule with \( k = N \)
3. If \( g_k - S_{k-1} \leq r \) then 4 else 6
4. \( g_k = S_{k-1} \)
5. Iterate backward by \( k = k - 1 \), goto 7
6. If \( g_k - r \) \( \geq b \) then 7 else 8
7. \( g_k = S_{k-1} - r \)
8. **End**
This algorithm traverses the video schedule backward from the end to the beginning. It sets the end of the video staged schedule be equal to the end of the original video schedule and works out the target schedule \(g\) by considering the network bandwidth and introducing disk accesses whenever necessary. In iteration, algorithm \(\Gamma\) considers the sufficiency of network bandwidth. Assume at a particular time \(k\) that the network is anticipated to fail to support the required bandwidth; the schedule will record that some of the data should be obtained from the memory buffer of the proxy, in which the data should have been retrieved from the disk well before \(k\). The problem is how to determine the disk access time. Algorithm \(\Gamma\) suggests that at the time that the amount of data needed to be retrieved from the buffer exceeds the proxy buffer size \(b\) allocated to the stream, a disk access of size \(b\) should be scheduled. The algorithm then resets: the requirement of obtaining the data from buffer arranges the process as described above until the beginning of the schedule.

As the algorithm traverses backward in time, the time we assign the actual disk access precedes the delivery of any data relied on this disk access. Moreover, as each of those disk accesses is scheduled to retrieve the maximum memory buffer size \(b\) allocated to this stream so that it can cover the maximum length of video stream delivery without making another disk access.

In effect, this algorithm divides a video stream into two parts. The first part is a bandwidth-limited stream \(g^N\) with a bandwidth not exceeding \(r\) and the second part is a stream \(g^D\) consists of several discontinuous impulses with a volume not exceeding \(b\).

The upper part of Figure 5 shows the cumulative video schedule, traditional video staging schedule and the proposed video staging schedule of a video title “Star Wars” between frames 100000 to 105000. The title is recorded at 30fps with an average bit rate of 740Kbps. We allocate 740Kbps bandwidth and 925KB memory buffer (equivalent to 10s of storage) for both video staging algorithms. The lower part of Figure 5 shows both the proposed and traditional video staging schedules (non-cumulative) in logarithmic scale. The grey line indicates the original schedule and the black line indicates the corresponding video staging schedule.

In the lower half of Figure 5, the cutoff bandwidth is shown. Data have to be obtained from both the server and the disk of the proxy. Comparing with the traditional video staging, the proposed algorithm required less disk accesses as shown in Figure 5, since the traditional video staging schedule follows the original schedule and go below the cutoff bandwidth most of the time. In this way, the network bandwidth cannot be fully utilized and wasted. In the contrary, the proposed video staging schedule does not always follow the original schedule even if the latter has dropped below the cutoff bandwidth. This is because the proposed algorithm can make use of the extra bandwidth to store up the upcoming data in the memory buffer until it is full. It thus maximizes the utilization rate of the network bandwidth. Consequently, the data required to retrieve from the local storage is reduced and fewer disk accesses are required. The amount of saving increases when we increase the cutoff bandwidth since, when the cutoff bandwidth go up, an even larger part of the original schedule will go below the cutoff bandwidth. The traditional approach will generate more wastage of network bandwidth.

![Figure 5: Video staging schedules for “Star Wars” using different algorithms](image)

### 3. Resource Requirement

The proposed algorithm takes the network bandwidth and memory buffer size into consideration to construct the network and disk schedules. The network schedule remains bandwidth limited and is invariant to the change of the given network bandwidth and memory buffer supplied to the scheduling algorithm. However, it can be shown that the number of disk accesses required will increase as either the given network bandwidth or memory buffer decreases.

**Theorem 3:**

For two different incoming bit rates \(r'>r\), if \(a\) and \(a'\) denote the corresponding number of disk accesses required when applying algorithm \(\Gamma\) with delivery schedule \(D\) and proxy buffer size \(b\), then the corresponding number of disk accesses will have a relation of \(a>a'\).

**Proof:** Consider the first two disk accesses \(s_m=D_m\) and \(g_m'=D_{m'}\) of schedule \(g\) and \(g'\) obtained by \(\Gamma(D,r)\) and \(\Gamma(D,r')\) respectively. Prior to any disk access, algorithm \(\Gamma\) behaves the same way as \(\Phi\), therefore \(g_k>g_k'\) for all \(k>m\) and \(k>m'\). As a result \(g-D\) must reach the buffer limit before \(g'\). A disk access has to be made by \(g\) before...
the one made by $g'$, if any. Now, consider the interval between two consecutive disk accesses made by $g$, say $g_m = D_m$ and $g_n = D_n$ where $n > m$. For $g$, suppose there are also two disk accesses within the region and the $j$-th access is made on $n + j = m$ so that $g_j \geq g_k = D_k$. As have $r' > r$, according to algorithm $\Gamma$, the largest possible value of $g'$ at time $m + 1$ can only develop to $b' > g_{m+1} > g_k = D_k$. Therefore, $g'$ can make second disk access at time $m$, but it will never have more than two disk accesses during this interval.

Therefore, $g'$ will make its first disk access later than and for every two consecutive disk accesses of $g$ th can only be at most two disk accesses of $g'$, so we conclude that the number of disk accesses of $g'$ will smaller than $g$ if their corresponding incoming $r$ constraints $r'$ and $r$ have a relation of $r' > r$.

Similarly, a change in memory buffer size would affect the number of disk accesses required.

**Theorem 4:**

Suppose there are two different proxies applying algorithm $\Gamma$ with the same delivery schedule $D$ and an incoming rate constraint $r$. If their buffer sizes are $b$ and $b'$ such that $b' > b$, the corresponding number of disk accesses are $a$ and $a'$, respectively then there will be a relation that $a > a'$.

**Proof:** Suppose $g$ and $g'$ are the corresponding schedules of the two proxies with buffer size $b$ and $b'$. As $b' > b$, $g$ will make its first disk access later than $g'$. When we consider the interval between two consecutive disk accesses of $g$, suppose $g'$ has also made a disk access within the interval, it will only be able to make another at the beginning of the interval. It will not be possible to have more than two disk accesses within this interval. Therefore, the number of disk accesses of $g'$ will be smaller than $g$.

The effects are shown in Figure 6. Depending on its own delivery schedule, each video stream forms a unique resource requirement table. Proxy server can base on these tables to adjust its resource allocation in order to achieve an optimal resource allocation scheme.

**Figure 6: Disk Storage Requirement Map**

**Acknowledgement**

This project is supported by the Hong Kong Polytechnic University.

**Reference**


