Design and analysis of a look-ahead scheduling scheme to support pause-resume for video-on-demand applications

Philip S. Yu, Joel L. Wolf, Hadas Shachnai

IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA

Abstract. In a video-on-demand (VOD) system, subscribers can choose both the movie they wish to view and the time they wish to view it. In such an environment there are invariably “hot” videos which are requested by many viewers. The requirement that each viewer be able to independently pause the video at any instant and later resume the viewing with little delay can cause difficulties in batching viewers for each showing. Under batching, a single video stream is shared by multiple concurrent viewers and a resume request has to wait for additional stream capacity to become available before actual resumption can occur. The conventional approach to the support of on-demand pause-resume provides one video access stream to disks for each video request. This can greatly increase the disk arm requirements of a VOD system. In this paper, we propose a more efficient mechanism to support the pause-resume feature using look-ahead scheduling with look-aside buffering. The idea is to use buffering to increase the number of concurrent viewers supportable. The concept of look-ahead scheduling is not to back up each viewer with a real stream capacity so he can pause and resume at any time, but rather with a (look-ahead) stream that is currently being used for another showing which is close to completion. Before the look-ahead stream becomes available, the pause and resume features have to be supported by the original stream through (look-aside) buffering of the missed content. It is shown via simulations that the proposed scheme can provide a substantially greater throughput than the approach without batching. Furthermore, for a given amount of buffer, the improvement in throughput grows more than linearly with the stream capacity of the server. In other words, the look-ahead stream scheduling scheme operates with good economy of scale because it is easier to form look-ahead streams for video servers with larger stream capacity.

Key words: Video-on-demand – Pause-resume – Performance analysis

1 Introduction

Multimedia servers are quite different from those of conventional computer file systems. For one thing, multimedia information, including motion video and audio, requires a guaranteed transfer rate. In NTSC video, for example, retrieval from or update to a storage system must have a sustained and almost constant rate of 30 frames/s. Besides the strict timing requirement, multimedia storage systems typically require large storage capacity, since the data rates for motion video are quite high. For example, 1.5 Mbps for MPEG-compressed video [6] corresponds to 1.5 GB for each 2-h movie.

A common application of multimedia servers is video-on-demand (VOD) [11, 13] service. In a VOD system, subscribers can choose both the video they wish to view and the time they wish to view it. This contrasts with services in which users can choose only from a small set of selections and/or watch them at prespecified times. For the purpose of this paper we consider VOD systems in which the service is homogeneous. In other words, a large number of multimedia streams of the same format, for example MPEG compressed, are stored and retrieved.

Pause and resume operations are among the most commonly used features on VCRs. In a VOD environment, there are inevitably “hot” videos which are requested by many viewers. Batching multiple viewing requests for the same video can greatly increase the number of viewers supportable by the video server. However, the requirement that each viewer be able to independently pause the video at any instant and later resume the viewing has caused difficulties in batching the viewers of each showing. In the VOD environment, viewers may be forced to wait before stream capacity becomes available to start the showing. However, once viewers have been admitted into the showing, it is generally felt undesirable to force them to incur a long wait for stream capacity to resume after a pause. To make sure that the pausing viewers can resume at any time, conventional approaches provide one video disk stream for each video request. Thus no batching is permitted. For each VOD server, there is a maximum number of video disk streams that can be supported. This is referred to as the stream capacity of the server. For example, if each disk can support \( N_{\text{stream}} \)
simultaneous video streams, a video server with \( N_{disk} \) disks will have a stream capacity of \( N_{stream} \cdot N_{disk} \). (Note that \( N_{stream} \) will depend upon the disk arm scheduling algorithm and the amount of buffer available to support each round of retrieval [9, 15].) Clearly, the number of viewers supportable under the conventional approach is at most equal to the stream capacity of the server.

In this paper, we propose an efficient mechanism to support the pause and resume feature while allowing batching of concurrent viewers of the same video. The objective is to support a larger number of viewers than the stream capacity by employing the new concepts of look-ahead stream scheduling with look-aside buffering. This is referred to as the LASS scheme. Since VOD systems are often disk arm limited and the price of memory typically decreases far more rapidly than the price of disks, we explore techniques using memory buffering to reduce the disk arm requirements or to increase throughput (the maximum viewers supportable) for a given stream capacity. The intent is to allow batching of concurrent viewers and avoid backing up each viewer by a real video stream as much as possible. Instead, streams which are currently being used for other showings but are close to completion are used to back up batched viewers, so that pause and resume can be done at any time. These are referred to as look-ahead streams. When a stream designated as a look-ahead stream for another showing completes, if the backed up viewer is not in pause mode, we look for another look-ahead stream to replace it so the completing stream can be used instead to accommodate a showing for other waiting viewers. (Thus the identity of a viewer’s look-ahead stream changes over time.) Otherwise, the completing stream will be used to support the resume of the pausing viewer.

We note that before the look-ahead stream becomes available, pausing and resuming have to be supported by the original stream through (look-aside) buffering of the missed content. That is, starting at the pausing point, the missed contents of the video are kept for the viewer in a look-aside buffer. Further details are given in Sect. 2. Real stream capacity will be reserved and put aside only if look-ahead streams are not available. An algorithm is provided to dynamically select the look-ahead streams and manage the buffer. Since all stream and buffer capacities reserved are put into common pools to be allocated on demand when the viewers go into pause mode, a substantial saving in hardware resources can be achieved. The scheme also provides a means to trade off system throughput and resume delay for a fixed amount of buffer. We develop a detailed simulation model to study the performance of the LASS scheme and conduct a sensitivity analysis of its buffer requirements. It is found that with sufficient buffer LASS can substantially improve the throughput of the video server compared to the conventional approach with no batching of viewers. Furthermore, for a given amount of buffer, the improvement in throughput grows more than linearly with the stream capacity of the server. That is to say, the LASS scheme operates with good economy of scale because it is easier to form look-ahead streams for video servers with larger stream capacity.

We briefly comment on some related work to support other aspects of VOD. Significant results were presented in [9, 13] regarding admission control techniques and the choice of service size to support multimedia applications. In [10], the concept of wait tolerance is explored to improve the batching effectiveness on viewer scheduling which determines the order that the viewer requests are scheduled. The proposed LASS scheme can work with any admission policy and viewer scheduling scheme. It provides an efficient way of utilizing disk stream capacity to support the scheduling of batched viewers with buffer. Certainly, LASS will be more effective if the admission policy and viewer scheduling scheme result in more and larger batches of viewers. The issues associated with supporting fast-forward and rewind are addressed in [2, 4]. Various papers have studied disk scheduling issues. In [15], a new formulation for disk arm scheduling schemes called grouped sweeping scheduling is proposed and analyzed. The goal is to minimize the buffer requirement. A similar concept is considered in [5]. Furthermore, [12] studies storage management and disk access algorithms in a disk array environment using this grouping approach. In [1, 13, 3], the issue of scalable data placement and scheduling algorithms in a disk array environment using this grouping approach. In [14], the issue of scalable data placement is considered. A somewhat different scheduling and placement perspective is taken in [14]. Nevertheless, both disk scheduling and video placement, each of which can affect the stream capacity of the VOD server, are orthogonal to the purpose of this paper: we assume a given stream capacity for the VOD server and try to maximize the number of viewers supportable by the server.

In Sect. 2, the proposed LASS scheme is presented. Section 3 contains a mathematical model and analysis of LASS. We describe the simulation model, assumptions and performance results in Sect. 4. Concluding remarks are provided in Sect. 5.

2 The LASS scheme

The preliminary concept of the LASS scheme is explained in Sect. 2.1. We also describe the stream and buffer states and provide examples. In Sect. 2.2 we present the stream scheduling algorithm where it is guaranteed that a paused viewer will be able to resume without incurring any delay. In Sect. 2.3 we describe an extension of the algorithm to provide a means to control the probability of delayed resume without strictly guaranteeing the capability to immediately resume. This offers a trade-off between the quality and the cost of the VOD service. The pause and resume operations are discussed in Sect. 2.4. Finally, we consider stream and viewing completions in Sect. 2.5.

2.1 Preliminaries

2.1.1 Basic concept

As mentioned above, the intent of look-ahead scheduling is not to back up each viewer with a real stream capacity so he can pause and resume at any time, but to back viewers up with a stream which is currently being used for another showing to be completed in the near future. This stream will be referred to as a look-ahead stream. We note that a look-ahead stream cannot simultaneously back up multiple viewers. Thus a stream can only be the look-ahead stream