TOWARDS ENHANCED RESOURCE SHARING IN VIDEO STREAMING WITH GENERALIZED ACCESS PATTERNS

Bashar Qudah and Nabil J. Sarhan

Department of Electrical and Computer Engineering
Wayne State University
Detroit, MI 48202, USA
Email: {bqudah,nabil}@wayne.edu

ABSTRACT

Recent workload characterization studies show that a large number of video streaming sessions do not access videos in a sequential manner from the beginning to the end, as assumed in most prior work. In this paper, we study the impact of realistic access patterns on resource sharing and propose and evaluate five enhancements to reduce server load and improve customer-perceived quality-of-service (QoS).

1. INTRODUCTION

Video streaming applications have grown dramatically in popularity over the Internet and cellular and cable TV networks. This paper focuses on streaming of stored video content, also known as *Video-on-Demand* (VOD).

Unfortunately, the number of video streams that can be supported concurrently is severely limited by the required real-time and high-rate transfers. Resource sharing techniques [7, 1, 5, 6, 8, 12, 10, 9, 13] face this challenge by utilizing the multicast facility.

Resource sharing has been studied extensively when videos are accessed sequentially from the beginning to the end. We refer to this access pattern as *Full-Content Access*. Actual workloads, however, suggest a different pattern, referred to as *Selected-Content Access* in this paper. It is a generalized pattern that captures the fact that a considerably large number of sessions start from a position other than the beginning of the video and finish before reaching the end [2, 3]. This pattern is common, especially for long videos. Popular interactions (such as jump forward, jump backward, and pause) can be considered as a series of selected-content access periods. Unlike other prior work, the study [11] used such realistic workload but focused on reducing the data requested by clients in interactive operations through enhanced data buffering and relied on traditional stream merging techniques without changing their stream merging decisions.

In this paper, we study the impact of selected-content access on streaming servers delivering data in a client-pull fashion using stream merging techniques, such as *Patching* [7] and *Earliest Reachable Merge Target (ERMT)* [5]. The *server-push* approach is not discussed because it is not suitable for selected-content access, can be used only for popular videos, may lead to server underutilization, and imposes waiting time on the majority of requests. Moreover, we propose five enhancements to reduce server load and improve the client perceived quality-of-service (QoS). We evaluate the effectiveness of the proposed enhancements through simulation.

The rest of this paper is organized as follows. Section 2 provides background information. Section 3 presents the proposed enhancements. Section 4 discusses the performance evaluation methodology and main results.

2. BACKGROUND INFORMATION

Let us discuss the main client-pull resource sharing techniques: Batching, Patching, and ERMT. While Batching services all waiting requests for a video using one full multicast video stream and requires only one client download channel at the video playback rate. Patching expands the multicast tree dynamically to include new requests. A new request joins the latest *regular* stream for the video and receives the missing portion as a *patch*. Hence, it requires two download channels and additional client buffer space. When the playback of the patch is completed, the client continues the playback of the remaining portion using the data received from the multicast stream and already buffered locally. To avoid the continuously increasing patch lengths, regular streams are retransmitted when the required patch length for a new request exceeds a pre-specified value called *regular window* (Wr). ERMT is a near optimal hierarchical stream merging technique. It also requires two download channels but makes each stream sharable and thus leads to a dynamic merge tree. A new client joins the closest reachable stream (*target*) and receives the missing portion by a new stream (*merger*). After the merger stream finishes and merges into the target, the later get extended to satisfy the playback requirement of the new client(s), and this extension can affect its own merge target.

This work is supported in part by NSF grant CNS-0626861.

Scheduling is another important aspect. The server maintains a waiting queue for every video and applies a scheduling policy to select an appropriate queue for service. All requests in the selected queue can be serviced using one channel. *First Come First Serve* (FCFS) [4] and *Maximum Queue Length* (MQL) [4] are two commonly used policies. FCFS selects the queue with the oldest request, whereas MQL selects the longest queue. MQL yields higher throughput and is used here.

3. SUPPORT FOR SELECTED-CONTENT ACCESS

Selected-content access has two conflicting effects on how much data is to be delivered by the streaming server. The shorter session lengths reduce the required data per client. However, the overlap of data and thus data sharing are also reduced because not all requests for the same video start from the same position.

This environment complicates resource sharing since later streams do not always access later data in the video. In Patching, for example, it is no longer satisfactory to consider only the last regular stream when servicing a new request. Instead, the server must examine all regular streams as potential merge targets. In this context, we define a *regular stream* as any non-patch stream, which may start from a position other than the beginning of the video and may finish before reaching the end.

For this environment, we propose three enhancements in stream merging and a new class of scheduling policies.

3.1. Proposed Stream-Merging Enhancements

The following proposed enhancements are applicable to all stream merging techniques.

Adopt Earlier Streams and Their Children (AESnC)
Traditionally, stream relationships (between a merger and a merge target) are determined when a merger stream starts or when merging occurs. In the new environment, a start of a potential target stream may require revising some of the previous decisions. For example, there were no potential targets for Stream 1 when it was initiated, as shown in Figure 1. Similarly, Stream 2 had no target when it started since it needed later data. With the original stream merging algorithms (such as Patching and ERMT), both streams will be regular streams (i.e., they will continue until possibly the end of the video and do not merge with other streams). A possible optimization is to downgrade Stream 1 to a patch and use Stream 2 as its merge target. In other words, Stream 2 *adopts* Stream 1 (which is an earlier stream) and all its possible children. A similar situation happens for Streams 4 and 5.

Serve Later Clients at No Cost (SLC@NC)
This relatively simple enhancement lets waiting requests to be served by an existing regular stream when their requested

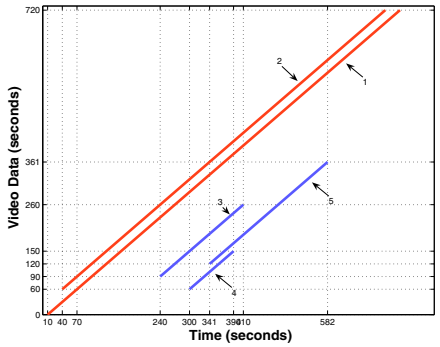


Figure 1: ERMT [Selected-Content Access with No Enhancements]

starting position (i.e., playback point) is reached by that stream. Hence, these requests will be served by Batching at no extra server cost.

Admit Clients with Active Wait (ACwAW)
With this enhancement, the server admits the client to an existing or a new stream when the current position delivered by the stream is earlier than the requested starting position by no more than AW_{max}. The client watches the earlier data while actively waiting for its requested data. For customer satisfaction, AW_{max} must be set to a small value.

3.2. Proposed Fuzzy Queue Length (FzQL) Scheduling

Selected-content access complicates scheduling. A video waiting queue may contain requests for clients varying in their desired starting positions. So, there are different subgroups of requests that can be serviced concurrently. Thus, we introduce the *virtual-queue* concept and define it as a subgroup of the waiting requests for a certain video with starting positions within AW_{max} from the one with the earliest starting position. MQL scheduling selects the queue (or virtual queue in this context) with the largest number of requests. The objective function that MQL tries to maximize is simply the virtual queue length. The proposed *Fuzzy Queue Length* (FzQL) policy, however, utilizes the $SLC@NC$ enhancement by favoring the streams that are more likely to be joined later by other waiting requests. In computing the objective function for a certain virtual queue vq, it considers not only the requests in vq but also the waiting requests for all starting positions further than those in vq. This is because these requests are likely to join the stream serving vq if they have to wait longer for service. FzQL is applicable only to regular streams and thus conventional MQL is used for other streams.

We present two alternative implementation of FzQL, called *Maximum Fuzzy Queue Length (MFzQL)* and *Weighted Fuzzy Queue Length (WFzQL)*. MFzQL works as follows. Each request in a virtual queue vq has full membership and thus will be counted as one. The other requests (belonging to other virtual queues) will be counted as fractions inversely propor-

tional to how far their requested positions are away from vq's earliest starting position (Ps_{vq}). Let us assume that for request i with starting position Ps_i, $\Delta_i = Ps_i - Ps_{vq}$. For a video of Length L, let us assume that the count of requests with starting positions in $[t_s, t_e[$ is $C(t_s, t_e)$. Thus for vq, the objective function is given by

$$F_m(vq) = \sum_{i=1}^{C(Ps_{vq},L)} 1.0/max(1, \lceil \Delta_i/AW_{max} \rceil),$$

where $AW_{max} > 0$. The counting process is illustrated in Figure 2. The symbol x indicates a waiting request for a certain starting position.

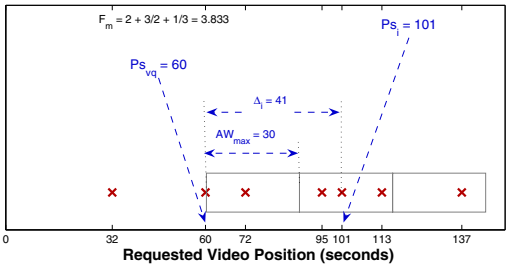


Figure 2: Illustration of Computing the MFzQL Objective

The WFzQL implementation favors the queues that are less likely to be served at no cost in the future by weighting F_m with the time T_{NoCost} required to serve the group at no cost using a regular stream. Hence, the objective function is given by $F_w(vq) = F_m(vq) \times min(T_{NoCost}, L)$.

4. EVALUATION

We study the impact of selected-content access and the effectiveness of the proposed enhancements (including FzQL) through extensive simulation. Only a subset of the results is shown for space limitation. We consider five performance metrics: customer defection (i.e., turn-away) probability, average waiting time for service, average waiting time to view the requested data, unfairness against unpopular videos, and unfairness against unpopular requested starting positions. The metrics are listed in order of importance. The last metrics is introduced here to quantify the bias against unpopular positions, whose requests may have higher defection rates. It is computed as the standard deviation of the defection rate for various position intervals in the video. The average waiting time to view the requested data includes the waiting time for service and the subsequent time to reach the requested position in the video.

4.1. Workload Characteristics

We assume the request arrival follows a Poisson distribution and the access to videos is highly localized and follows a Zipf-like distribution. The access pattern is based loosely on [3, 11]. Table 1 summarizes the main workload characteristics.

Table 1: Default Parameter Values

Parameter	Default Value(s)
Arrival Rate (λ)	30 Req/min
Number of Videos (N)	120
Video Length (L)	120 min
Video Popularity	Zipf-like with skewness θ = 0.271
Waiting Tolerance Model	Exponential with μ_{tol} = 2 min
Session Length Distribution	Normal with μ_{len} = 30 or 60 min
Session Starting Pos. Dist.	Zipf-like with skewness = 0.0
% Sessions Starting at Pos. 0	20%
Max. Active Wait (AW_{max})	2 min

4.2. Result Presentation and Analysis

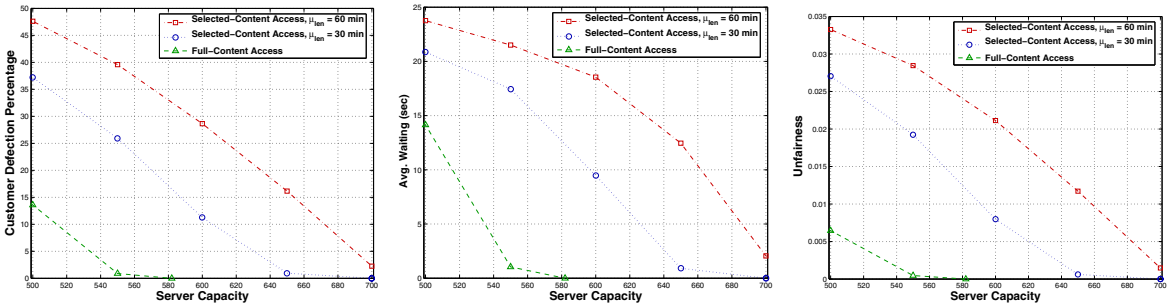
Figure 3 demonstrates the impact of the video access pattern on performance when ERMT is used. The same behavior happens with Patching and thus its results are not shown. Two means for the session length are examined (μ_{len} = 60 and 30) in the selected-content access. Interestingly, although with full-content access, the session length is equal to the entire video length, it leads to better performance than the selected-content access. The reason is due to the reduction in data sharing, caused by the variation in the requested starting position.

Figure 4 shows the effectiveness of the proposed stream-merging enhancements and WFzQL when ERMT is applied. The "No Enhancements" case uses MQL and $AW_{max} \approx 0$. The "All Enhancements" cases use WFzQL and the three enhancements, but the $ACwAW$ enhancement is effectively disabled when $AW_{max} \approx 0$. The individual effects of various policies and enhancements (except for $ACwAW$) are not shown to save space. The results for MFzQL are also not shown because it is outperformed by WFzQL. The results indicate that applying the first two enhancements and WFzQL improves the service significantly based on every metric, except for the starting-position unfairness when the server capacity is low. Applying ACwAW improves the service further in terms of the two most important metrics, but as expected, it introduces additional delay until the desired position in the video is reached.

Figure 5 depicts the improvements in the two most important metrics achieved by the proposed schemes (including WFzQL) for two different means of session length. The results for both Patching and ERMT are shown. These results demonstrate that the proposed schemes improve the server throughput and reduce the waiting time significantly. The reductions in customer defections and waiting times are $35\% - 100\%$ and $21\% - 100\%$, respectively.

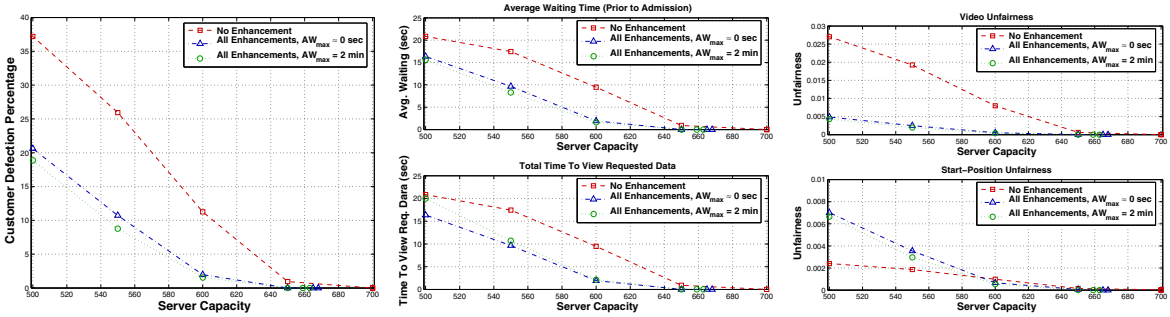
5. CONCLUSION

We have studied the impact of selected-content access on stream-merging video streaming servers and have proposed five enhancements to reduce the load on the server and to improve



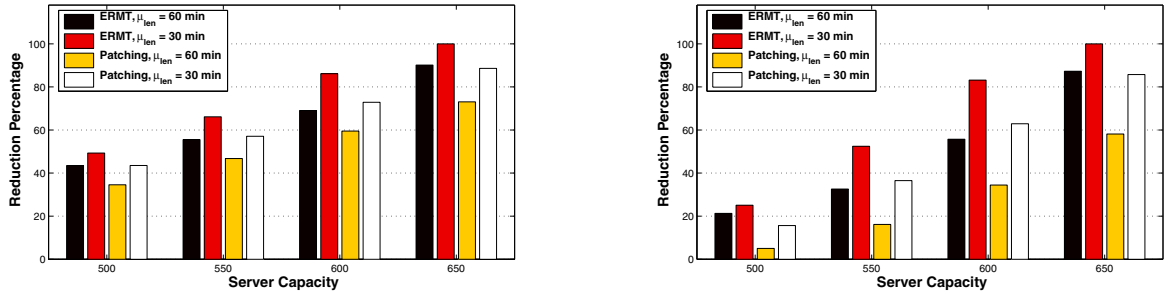
(a) Defection Rate (b) Average Waiting Time (c) Video Unfairness

Figure 3: Impact of Access Pattern [ERMT]



(a) Defection Rate (b) Average Waiting Time (c) Unfairness

Figure 4: Impact of Proposed Enhancements [ERMT, Selected-Content Access, $\mu_{len} = 30$ minutes]



(a) Defection Rate (b) Average Waiting Time

Figure 5: Achieved Reductions in Defection Rate and Waiting Time [$AW_{max} = 2$ minutes]

customer-perceived QoS. We have evaluated the effectiveness of the proposed enhancements through simulation, considering five performance metrics. The results show that selected-content access reduces significantly data sharing among requests and that the proposed enhancements are very effective in improving the sharability and reducing the waiting times.

6. REFERENCES

[1] Y. Cai and K. A. Hua. An efficient bandwidth-sharing technique for true video on demand systems. In *Proc. of ACM Multimedia*, pages 211–214, Oct. 1999.

[2] L. Cherkasova and M. Gupta. Characterizing locality, evolution, and life span of accesses in enterprise media server workloads. In *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video (NOSSDAV)*, pages 33–42, May 2002.

[3] C. Costa, I. Cunha, A. Borges, C. Ramos, M. Rocha, J. Almeida, and B. Ribeiro-Neto. Analyzing client interactivity in streaming media. In *Proc. of The World Wide Web Conf.*, pages 534–543, May 2004.

[4] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *Proc. of ACM Multimedia*, pages 391–398, Oct. 1994.

[5] D. L. Eager, M. K. Vernon, and J. Zahorjan. Optimal and efficient merging schedules for Video-on-Demand servers. In *Proc. of ACM Multimedia*, pages 199–202, Oct. 1999.

[6] D. L. Eager, M. K. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. *IEEE Trans. on Knowledge and Data Engineering*, 13(5):742–757, Sept. 2001.

[7] K. A. Hua, Y. Cai, and S. Sheu. Patching: A multicast technique for true Video-on-Demand services. In *Proc. of ACM Multimedia*, pages 191–200, 1998.

[8] L. Juhn and L. Tseng. Harmonic broadcasting for Video-on-Demand service. *IEEE Trans. on Broadcasting*, 43(3):268–271, Sept. 1997.

[9] B. Qudah and N. J. Sarhan. Analysis of resource sharing and cache management techniques in scalable video-on-demand. In *Proc. of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, page 327334, Sept. 2006.

[10] B. Qudah and N. J. Sarhan. Towards scalable delivery of video streams to heterogeneous receivers. In *Proc. of ACM Multimedia*, pages 347–356, Oct. 2006.

[11] M. Rocha, M. Maia, I. Cunha, J. Almeida, and S. Campos. Scalable media streaming to interactive users. In *Proc. of ACM Multimedia*, pages 966–975, Nov. 2005.

[12] N. J. Sarhan and C. R. Das. Caching and scheduling in NAD-based multimedia servers. *IEEE Trans. on Parallel and Distributed Systems*, 15(10):921–933, Oct. 2004.

[13] N. J. Sarhan and B. Qudah. Efficient cost-based scheduling for scalable media streaming. In *Proc. of Multimedia Computing and Networking Conf. (MMCN)*, Jan./Feb. 2007.