A Practical Performance Analysis of Stream Reuse Techniques in Peer-to-Peer VoD Systems*

Leonardo Bidese de Pinho and Claudio Luis de Amorim

Parallel Computing Laboratory, COPPE Systems Engineering Program,
Federal University of Rio de Janeiro, RJ, Brazil
{leopinho,amorim}@cos.ufrj.br

Abstract. Although many works have reported simulated performance benefits of stream reuse techniques to the scalability of VoD systems, these techniques have been rarely evaluated in practical implementations of scalable VoD servers. In this work we investigate the behavior of representative stream reuse techniques in the GloVE system, a low-cost VoD platform whose scalable performance depends on the combination of the stream techniques it uses. More specifically, we show experimental results focusing on the requirements of the amount of server's channels and aggregate bandwidth that GloVE demands for several combinations of stream reuse techniques. Overall, our results reveal that stream reuse techniques in isolation offer limited performance scalability to VoD systems and only balanced combinations of batching, chaining, and patching techniques explains the scalable performance of GloVE on delivering popular videos with low startup latency while using the smallest number of server's channels.

1 Introduction

In recent years, considerable research efforts have been concentrated on the design of scalable Video on Demand (VoD) systems since they represent a key-enabling technology for several classes of continuous media applications, such as distance learning and home entertainment. In particular, a central problem VoD designers face is that in a typical VoD system with many videos, even a transmission of a single high-resolution video stream in a compressed format consumes a substantial amount of resources of both the video server and the content distribution network. Given that users can choose any video and start playback at anytime, a significant investment on server's hardware and network's infrastructure will be required to support large audiences. Therefore, it is fundamental that a VoD server supports stream distribution strategies capable of using efficiently the available network resources in order to reduce its cost per audience ratio.

Typically, a VoD server supports a finite pool of stream channels, where the amount of channels is often determined by the server's bandwidth divided by

* This work was partially sponsored by the Brazilian agencies CAPES and CNPQ.

M. Danelutto, D. Laforenza, M. Vanneschi (Eds.): Euro-Par 2004, LNCS 3149, pp. 784–791, 2004.

the video playback rate. The reason that conventional VoD systems cannot scale is because they dedicate one different channel to each active client, in an one-to-one approach. As a result, the total number of clients a conventional VoD system supports is equal to the limited number of server's channels. Due to the scalability limitation of the one-to-one approach, several scalable streaming techniques have been proposed in the literature. Basically, such techniques allow multiple clients to share the stream contents that are delivered through each of the server's channels, in an one-to-many approach.

Three of the most well-known scalable streaming techniques are *Batching* [1] - where near requests to the same video are first enqueued and served afterwards by a single multicast stream from the server, *Chaining* [2] - in which clients on behalf of the server can send video streams to subsequent requests, provided the requests arrive within a certain time interval during which the prefixes are still in the playout buffers of the clients, and *Patching* [3] - where a new client that requests a video is first inserted into an available active multicast stream for the video and upon receiving the video stream the client will temporarily store it into a local buffer. In addition, the server will send to the new client an extra video stream, namely the patch, which contains the video segment that the client missed.

Recently, we introduced a novel scalable streaming technique, called the Cooperative Video Cache (CVC) [4] that combines chaining and patching under a P2P model. CVC implements a cooperative stream cache over the distributed collection of client's playout buffers[1] that store dynamically video streams the server sends to the clients. The key idea behind CVC is to use such a large dynamic stream cache as the primary source of video contents so that most client requests become cache hits with the server attending only the remainder cache misses. Experimental results of the CVC-based VoD prototype we developed, namely the Global Video Environment (GloVE) [5], demonstrated that CVC can reduce drastically VoD server's occupancy thus enabling scalable VoD systems to be built.

In contrast with existing works that evaluated performance of scalable streaming techniques through simulations, in this work we assess the efficiency of representative streaming techniques in practical situations. More specifically, since the GloVE prototype allows the evaluation of streaming techniques either separately or in any combination of them, we examined their relative performance contribution to the scalability of VoD systems that GloVE encompasses. In addition, we addressed the impact of the distribution of popularity of videos on the effective use of VoD system's resources.

The remainder of this paper is organized as follows. In Section 2 we describe the main characteristics of the GloVE platform and operation modes. In Section 3 we analyze GloVE's performance results for several scenarios of streaming techniques and workloads. In Section 4 we compare our work to related ones. Finally, we conclude in Section 5.

[1] Reserved memory space used by the client equipment to hide the network jitter and inherent variations of VBR videos.

2 The GloVE System

The GloVE (Global Video Environment) is a scalable VoD system that implements the cooperative video cache while supporting a P2P system with a centralized metadata component, namely the CVC Manager (CVCM). The CVCM monitors globally the video contents stored in the playout buffers of the clients. The CVC Client (CVCC) software allows the content of any client buffer to be shared with other connected clients, reducing the demand on the CVC Server (CVCS). GloVE assumes that the communication network has IP multicast support and that the VoD server manages the video as a sequence of blocks that can be randomly accessed. Further details of GloVE can be found in [6].

The design of CVCM allows to select different operation modes according to several combinations of stream reuse techniques (e.g., Batching, Chaining, and Patching)[5]. In this article, however, we restrict our analysis to the following modes: *Batching*, in which late requests for a video will form a group that will be serviced from a previous client that is in the prefetching phase; *Chaining*, CVCM implements a slightly different form of chaining in that every stream has only one receiver; *Patching+Batching*, which implements a combination of patching and batching; *CVC+Batching*, which adds batching to the CVC technique.

3 Experimental Analysis

3.1 Evaluation Methodology

The experiments we describe below were performed on a 6-node cluster of Intel Pentium IV 2.4 GHz, 1 GB RAM, running Linux kernel 2.4.18 using a 3Com Fast Ethernet switch with IP Multicast support. One node executed both CVCS and CVCM whereas each of the other nodes executed multiple instances of clients. We used a MPEG-1 video (352x240, 29.97 frames/s) of Star Wars IV movie, with average video playback rate near to 1.45 Mbps. To allow multiple clients per node, we developed an emulator of MPEG-1 decoder, which uses a trace file containing the playback duration of each segment of the video. Our workload ranges from medium to high client arrival rates according to a Poisson process with values of 6, 10, 20, 30, 60, 90, and 120 clients/min. We assumed that the CVCS can deliver at most 56 MPEG-1 streams simultaneously, and supports two types of client access: *smooth access* when clients request video blocks according to the video's playback rate, and *burst access* when clients request blocks as fast as possible and the playout buffers are filled at the transmitting rate determined by the availability of CVCS' bandwidth. We used 64 KB blocks as the system access unit and the prefetch limit[2] of 32 blocks, which can hold a video segments up to 11s. We evaluate performance of GloVE under practical network conditions, using playout buffer sizes of 64, 128, and 256 blocks, which can store video sequences that last near to 22, 44, and 88s, respectively. Given

[2] Minimal amount of video blocks that must be stored in the playout buffer in order to initialize video's playback.

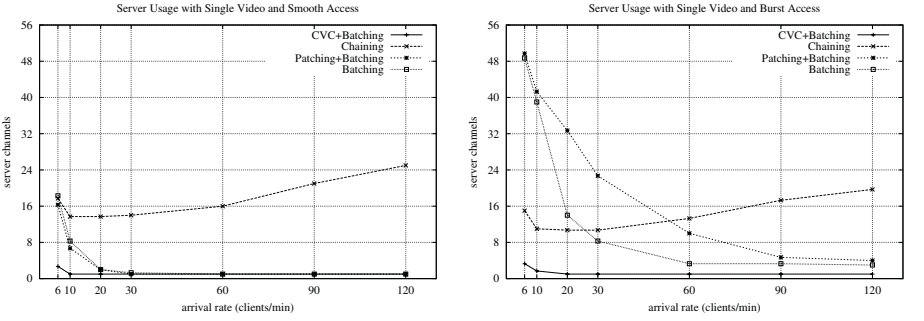

Fig. 1. Channels usage for each GloVE mode and access type. (a) Smooth. (b) Burst.

to the restricted space, we only show the results for 128 blocks, which hold video segments up to 44s. More detailed results are reported in [6]. In the experiments with multiple videos, we emulated a collection of eight videos, where the video popularity follows a general Zipf distribution with $\alpha = 0.7$ [1]. In addition, we also investigated the sensitivity of the VoD system to $\alpha = 0$ which represents the uniform distribution, and $\alpha = 1$, which represents Zipf without skew.

3.2 Experimental Results

We present the results of our experiments in three parts: single video, multiple videos, and a sensitivity analysis of server's performance to videos' popularity.

Figure 1 shows the occupancy of server's channels, or simply server occupancy, using block request rate with smooth and burst accesses for a single video and different GloVE modes. Server occupancy indicates the relative degree of scalability of a particular technique. Specifically, the lower server occupancy a technique produces for a given amount of active clients the higher is its scalability. CVC+Batching with smooth access occupies only one channel for almost all arrival rates we measured. This mode allows stream reuse for incoming requests up to 30s apart[3]. So, only for very low arrival rates with intervals between requests greater than 30s that misses to the playout buffers become significant. Intuitively, stream reuse is higher for smooth accesses because they generate long prefetches whereas burst accesses often issue short prefetches. Notice that, this difference in access behavior is not shown in the figure due to the buffer size we used. The Chaining mode is influenced negatively by prefetching for arrival rates higher than 10 clients/min. In this case, the higher is the arrival rate, the higher is the probability of a client to arrive while all previous ones either issued prefetching or are transmitting video streams, so that they can not be providers of video segments and a new server channel will be required. Also, as the prefetching phase is shorter with burst accesses, server's occupancy presented slightly smaller values in this mode. The Patching+Batching mode is influenced strongly by the client access type. The fact is that this mode initially

[3] Due mostly to playout buffer of 44s minus the prefetch limit of 11s.

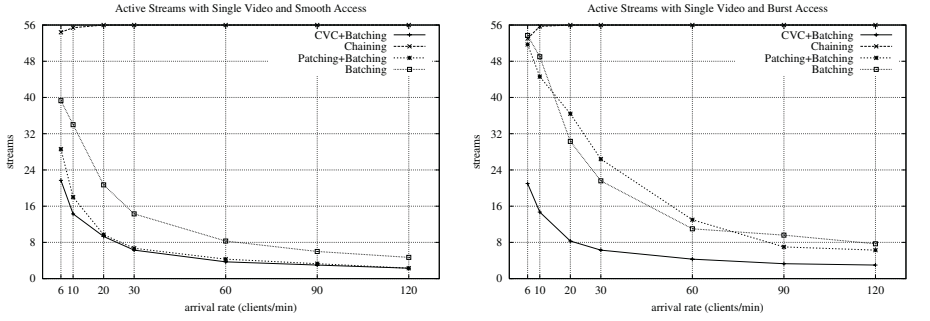

Fig. 2. Active streams for each GloVE mode and access type. (a) Smooth. (b) Burst.

can only use batching because patching requires multicast streams to be available. When using burst accesses the server occupancy tends to be very high for arrival rates less than 60 clients/min. In contrast, with smooth access the chances for Batching increase significantly, so that the minimum server occupancy is reached at 30 clients/min. The client access type also determines the efficiency of the Batching mode. Specifically, when using the server with smooth access the channel's occupation is very similar to that achieved by Patching+Batching. The main difference between the two modes is due to the higher average prefetching that Batching generates for burst accesses and arrival rates between 20 and 90 clients/min. When Patching is used, the average prefetching is substantially reduced, decreasing the opportunity of stream reuse for the next arriving clients.

Figure 2 shows the amount of active streams for each server mode when using a single video system. The values in the figure indicate the aggregated bandwidth that each mode requires. As shown in the figure, the minimum amount of active streams is two, one stream from each of the server and the first client. For the CVC+Batching mode the number of active streams is practically the same whether using smooth or burst accesses to the server. Another factor that affects the number of active streams is the type of stream reuse that is used most frequently. Whenever possible Patching is preferable since it generates fewer streams. Otherwise, the server will attempt to employ Chaining which generates a new stream if the server succeeds. The latter occurs often when the time interval between requests is not greater than than the capacity of the playout buffer in seconds. Intuitively, with Chaining the number of active streams is equal to the amount of clients, since the streams that Chaining generates have only one receiver and is independent of the client access type. Patching+Batching is highly influenced by client access type. For smooth access, the large amount of multicast streams that Batching generates increases the use of Patching as well. Burst accesses, particularly for arrival rates less than 60 clients/min, produce shorter prefetches, decrease the use of Batching, which in turn reduce Patching, and increase the amount of streams with a single receiver. In the Batching mode, the number of active streams depends on the client access type only for arrival rates less than 30 clients/min. In this case, when clients use burst accesses, the

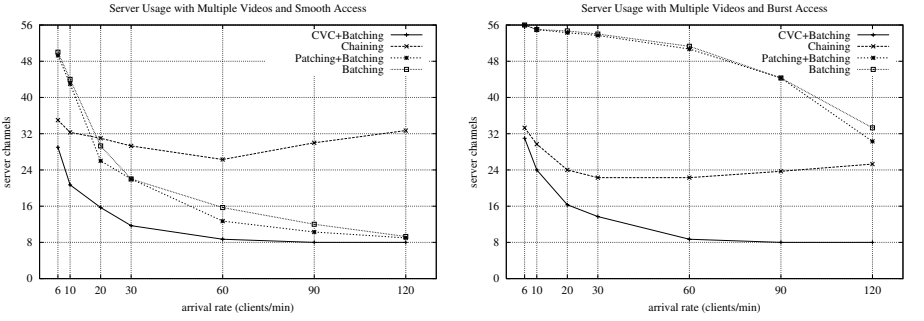

Fig. 3. Channels usage for each GloVE mode. (a) Smooth access. (b) Burst access.

majority of streams comes from the server itself, because there is no opportunity for Batching. For smooth accesses, Batching dominates stream generation.

Figure 3 presents server occupancy according to the different GloVE modes when the VoD server offers eight videos. The videos' popularity follows a Zipf distribution with $\alpha = 0.7$. In the best case, at least one channel will be used for each different video the clients request. In the worst case, one channel will be used for each client request, which is ultimately the behavior of a conventional VoD system. The curves related to CVC+Batching indicate that the minimum amount of busy channels occurred at 60 clients/min. This is not surprisingly since for a single video the referred minimum was near to 10 clients/min. The negative impact of the client access type on server occupancy was restricted to small arrival rates. Similarly to the single video case, Chaining still suffers from the negative impact of prefetching, especially for arrival rates higher than 30 clients/min. At this rate, we noticed the minimum server occupancy of 22 channels. Chaining achieved better performance when clients used burst accesses to the server. For Patching+Batching, most of the observations we made for single videos hold also for multiple videos. The main difference is that there exist considerably fewer opportunities to apply either Batching or Patching. For arrival rates ranging from 90 to 120 clients/min, the minimum server occupancy is reached with smooth accesses. Note that with burst accesses, 30 channels is the minimum value of server occupancy, which was achieved with the highest arrival rate. The behavior of Batching is very similar to that of Patching+Batching we analyzed above.

In Fig. 4 we show the amount of active streams for a server with multiple videos. The minimum number of active streams tends to be 16 provided that all the videos are requested twice at least. In CVC+Batching, the curves remained similar for both burst and smooth access types. As explained for a single video, the emphasis on using preferably Patching led to the least amount of active streams. Naturally, Chaining tends to be used more for requests within shorter interarrival rates. In Chaining the number of active streams is equal to the number of active clients, as in the case of single video. Comparing Patching+Batching with Batching, it becomes clear that both behave similarly. The main difference is that Patching+Batching generates fewer streams due to Patching. Also, while

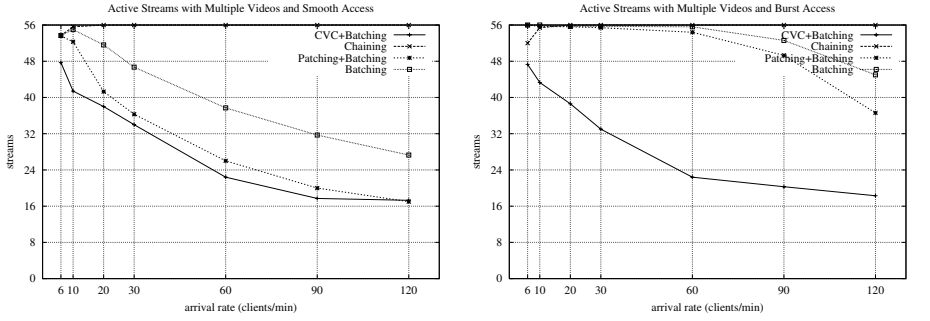

Fig. 4. Active streams for each GloVE mode. (a) Smooth access. (b) Burst access.

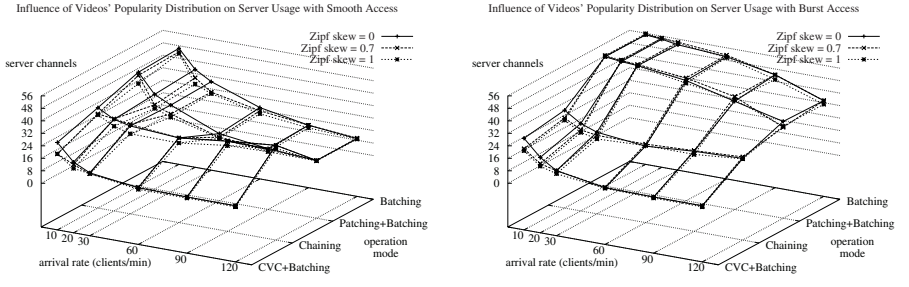

Fig. 5. Channels usage for different Zipf skews. (a) Smooth access. (b) Burst access.

the former achieved the minimum of 17 streams, the latter created 27 streams at least. Due to space limitation we do not present results for start-up latency. However, as reported in [6], the average start-up latency we measured was low ranging from 10 to 13s and 1.2 to 9.8s for smooth and burst access, respectively.

Figure 5 illustrates the impact of the distribution of video's popularity on the occupancy of server's channels. The variations of the popularity distribution of videos did not impact significantly the amount of channels the VoD system required to service the client requests. Indeed, none of the operation modes presented variations higher than 20% on the occupancy of server's channels.

4 Related Work

The original works that introduced the above techniques are Batching [1], Chaining [2], Patching [3], and CVC [4]. GloVE uses Chaining and Patching as originally proposed, and exploits Batching but in a different way from the original work. Specifically, in GloVE clients instead of the server can apply the Batching technique. The work in [7] presents a comparison of stream merging algorithms, but does not report results of any practical implementation. An additional analysis of related works focusing on P2P systems can be found in [6].

5 Conclusions

In this work we compared the performance of stream reuse techniques implemented in a practical P2P VoD system, namely the Global Video Environment (GloVE). In particular, we described different operation modes of GloVE according to the combination of several stream reuse techniques it uses, namely Batching, Patching, Chaining, and CVC. Also, we measured the influence of client access type (either smooth or burst) on server's performance. Finally, we analyzed the impact of video popularity distribution on system behavior.

Overall, the CVC+Batching mode outperformed the other modes for VoD servers with either single or multiple videos. Also, the client access type does not significantly affect CVC+Batching performance, suggesting that CVC+Batching will work efficiently for different VoD server designs. Furthermore, the results revealed that some skews on videos' popularity distribution will not impact substantially the resulting VoD system's performance. Thus, we speculate that a VoD system coupled with CVC+Batching will attain scalable performance for large audiences.

Currently, we are working on mechanisms that will support scalable VoD systems for mobile environments with heterogeneous devices. Also, we plan to extend GloVE to dynamically self-adapt to variations on network and peer conditions.

References

1. Dan, A., Sitaram, D., Shahabuddin, P.: Dynamic Batching Policies for an On-Demand Video Server. Multimedia Systems **4** (1996) 112–121
2. Sheu, S., Hua, K.A., Tavanapong, W.: Chaining: A Generalized Batching Technique for Video-On-Demand. In: Proceedings of the International Conference on Multimedia Computing and Systems. (1997) 110–117
3. Hua, K.A., Cai, Y., Sheu, S.: Patching: A Multicast Technique for True Video-on-Demand Services. In: Proceedings of the ACM Multimedia. (1998) 191–200
4. Ishikawa, E., Amorim, C.: Cooperative Video Caching for Interactive and Scalable VoD Systems. In: Proceedings of the First International Conference on Networking, Part 2. Lecture Notes in Computer Science 2094 (2001) 776–785
5. Pinho, L.B., Ishikawa, E., Amorim, C.L.: GloVE: A Distributed Environment for Scalable Video-on-Demand Systems. International Journal of High Performance Computing Applications (IJHPCA) **17** (2003) 147–161
6. Pinho, L.B., Amorim, C.L.: Assessing the Efficiency of Stream Reuse Techniques in Peer-to-Peer Video-on-Demand Systems. Technical Report ES-626/04, COPPE/UFRJ Systems Engineering Program (2004)
7. Bar-Noy, A., Goshi, J., Ladner, R.E., Tam, K.: Comparison of stream merging algorithms for media-on-demand. In: Proceedings of the SPIE Multimedia Computing and Networking (MMCN), San Jose, CA (2002)