Batched Patch Caching for Streaming Media

Pascal Frossard and Olivier Verscheure, *Member, IEEE*

***Abstract*—This letter elaborates on a scheme that combines batch patching at an origin server and prefix/interval caching at an edge server receiving the clients' requests. We derive a cost function that factors in the aggregate backbone rate, the cache occupancy, and the disk bandwidth utilization. We define the optimal batched patch caching strategy as a function of the client request rate. Finally, we show how various strategies including full caching, no caching and pure prefix caching with no patching are optimal derivations of our scheme under different request rates. We demonstrate the benefits of our scheme compared to classical streaming strategies.**

***Index Terms*—Application-level multicast, caching proxy, multimedia streaming, prefix caching, , server scheduling.**

I. Introduction

THE widespread use of the Internet and the maturing digital video technology have led to an increase in various streaming media applications such as webcasts, distant learning, and corporate communications. As access providers are rolling out faster last-mile connections, the bottleneck is shifting upstream to the provider's backbone, peering links, and the best-effort Internet. This problem can be partially addressed by *edge delivery* of streaming objects from an edge server or via content distribution networks, together with multicast strategies to reduce the backbone bandwidth consumption. While the edge delivery of streaming media will increase its scale and reach, handling streaming objects brings additional complexities at the proxies due to the large object size, long-lived nature of the objects, and isochronous delivery requirements from the users.

Among the several schemes proposed to capitalize on the benefits of multicast for VOD services, Patching [1]–[3] and Batch Patching [4], [5] are certainly two of the most efficient techniques. Each batch of requests is served over one or two channels—either a regular channel alone or the combination of a regular channel and a patching channel. A regular channel delivers the full video from start to finish while a patching channel delivers only the missing part of the video from the start until the point at which the clients join the regular channel. The client receives both the patch and the ongoing stream and buffers the latter while playing back the former. Once the patch is exhausted, the client switches to the buffered regular multicast. The usage of edge servers can reduce the transmission costs by offering a caching opportunity close to the clients. Most of the caching schemes are either based on full caching (the complete video stream is stored at the proxy) or prefix caching [6], [7]. Additionally, caching the prefix hides the startup latency and jitter in the network, and thus allows for *user-transparent* request batching.

Manuscript received October 30, 2001. The associate editor coordinating the review of this letter and approving it for publication was Prof. K. Park.

The authors are with the IBM Research Division, Yorktown Heights, NY 10598 USA (e-mail: frossard@us.ibm.com).

Publisher Item Identifier S 1089-7798(02)04488-5.

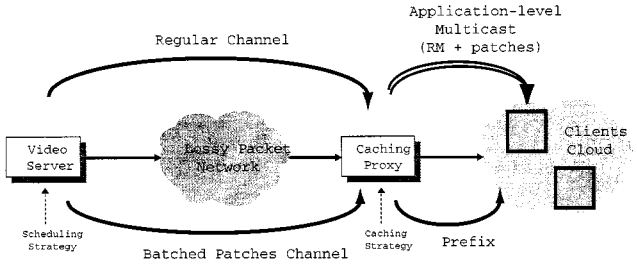

Fig. 1. Overall framework of Batch Patch Caching for Streaming Media.

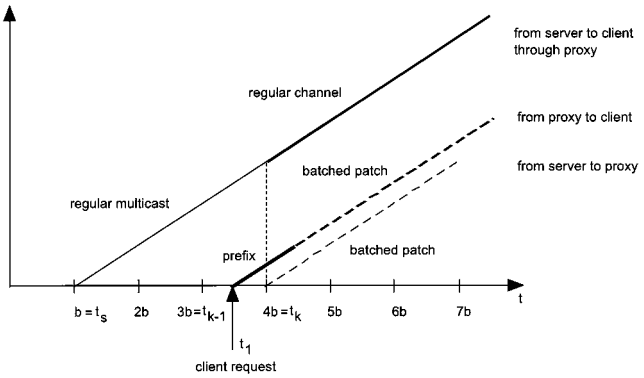

Fig. 2. Batched Patch Caching temporal scenario.

In this letter, we address the problem of efficiently streaming a set of heterogeneous videos from a remote server through an edge server to a clients' cloud so that the clients experience playback with theoretically *null* startup delays. We build on the optimized batch patching idea [4] by introducing a proxy cache in the path from the origin server to the clients cloud. We adopt the intuitive approach consisting of storing the first b units of time in the proxy cache. That is, the proxy *permanently* caches *prefixes* of b units of time. Moreover, we impose the proxy cache to play the role of a client for the origin server. That is, all the patches and regular multicasts streamed out of the server are requested by the proxy and are thereby streamed through it (see Fig. 1). This design approach has several advantages among which: 1) it eliminates the need for network-level multicast, which can be advantageously replaced by application-level multicast; 2) it allows for client-based stream adaptation (heterogeneous client capabilities); and 3) it has the potential to decrease the number of streams concurrently streamed to a given client. The former leads to a change of terminology. In the remainder, we use *regular channel* and *patch channel* instead of regular and patch multicasts.

Our scenario is illustrated in Fig. 2. The proxy divides the time axis into intervals $[t_{i-1}, t_i]$ of duration b units of time. As-

sume a request arrives at the proxy at time $t_1 \in [t_{k-1}, t_k)$. The proxy immediately starts streaming the requested asset to the client (i.e., the stored prefix of size b). Assume the most recent regular channel (RC) was started at time t_s, with $t_s < t_1$ is an integral number of b units of time. If t_k is such that $t_k < t_s + W$, the proxy joins the RC at time t_k and streams it through to the client, which buffers the stream while playing back the prefix. Also at time t_k, the proxy requests a patch of duration $t_k - t_s$ and caches it for future requests *within the same patching window* of length W. However, if $t_k \geq t_s + W$, a new regular channel of duration $T - b$ (the prefix of duration b is sitting in the proxy) is requested from the server at time t_k. A similar strategy has been proposed in [5].

II. Problem Formulation

We now derive a cost function that factors in both the aggregate backbone rate R, the cache occupancy S and the disk bandwidth utilization $D_{\{R,W\}}$ at the proxy under the scheduling/caching strategy described here above. First we compute the number of video chunks of duration b the edge server has to request from the origin server every patching window. Upon reception of a patch, the edge server stores it in the cache for a period of $W = Nb$ units of time, so that it is available for the last requests of the same patching window. Assume the requests are modeled by a Poisson process with parameter λ (i.e., empty batch of duration b with probability $p = e^{-\lambda b}$). The average number of patched chunks, μ, is therefore given by computing all the different possibilities along any patching window. It is given by

$$\mu = \frac{p^{N+1} - (N+1)p + N}{1-p}. \tag{1}$$

The total backbone bandwidth at stationary state is given by

$$R = \frac{\mu br + (T-b)r}{(N+1)b + \lambda^{-1}} \tag{2}$$

where r is the average transmission rate of the patches (assumed to be the streaming rate of the asset) and λ^{-1} is the average time interval between successive requests. The average storage capacity S needed for caching is then given by

$$S = br + \mu br. \tag{3}$$

We now compute the disk bandwidth utilization for both reading and writing operations. Recall that the proxy forwards the regular channel to the clients without accessing the disk at the edge-server[1]. The disk reading bandwidth, D_R is thus expressed as

$$D_R = r\frac{\lambda b \sum_{i=1}^{N+1} ib}{(N+1)b + \lambda^{-1}} = r\frac{\lambda b^2 (N+1)(N+2)}{2((N+1)b + \lambda^{-1})}. \tag{4}$$

The disk writing bandwidth D_W is given by

$$D_W = \frac{\mu br}{(N+1)b + \lambda^{-1}}. \tag{5}$$

[1]We assume an ideal network. In practice, a buffer may be required at the proxy to absorb the delay jitter. This buffer would require an extra term in both D_R and D_W without affecting the behavior of the cost function.

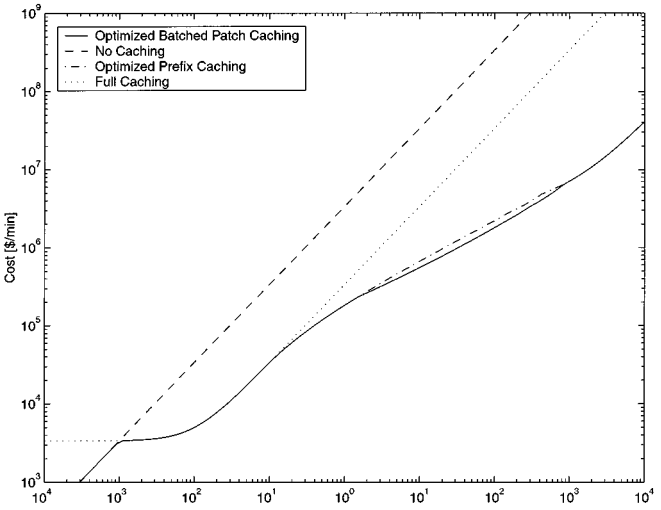

Fig. 3. Comparison of the cost of different streaming strategies as a function of the request rate λ ($\beta/\alpha = 10^{-3}, \gamma/\alpha = 10^{-1}, T = 90$ min, $r = 5$ Mbps).

Problem Formulation: Let the tuple $(\alpha, \beta, \gamma_1, \gamma_2)$ denotes the set of costs attached to the different resources described hereabove. We define the cost function C as

$$C(\lambda, b, N) = \alpha R + \beta S + \gamma_1 D_R + \gamma_2 D_W. \tag{6}$$

The problem may thus be formulated as follows: *Given* a media asset of duration T and streaming bandwidth r being requested at an average rate λ^{-1}, *find* the tuple (b, N) that minimizes the cost function C, *possibly* under a limited client buffer B_c. Note that the client buffer size is given by

$$B_c = \min\{(N+1)br, (T - (N+1)b)r\}. \tag{7}$$

This generic problem formulation encompasses several classical schemes including full caching and no caching strategies. It thus provides a unified framework toward the optimal streaming strategy, depending on the asset and network characteristics. The following section presents the results obtained through standard numerical optimization techniques under different scenarios.

III. Results and Discussions

Assume the relative cost of storage is low compared to that of the backbone bandwidth (e.g., $\beta/\alpha = 10^{-3}$, as proposed in [8]). For the sake of simplicity, also assume the disk bandwidth cost is similar in reading and writing operations (i.e., $\gamma_1 = \gamma_2 = \gamma$).

Fig. 3 compares the cost function for different streaming strategies. The ratio between the disk and network bandwidth costs has been set to 10^{-1}. It is indeed cheaper to increase the disk bandwidth than to increase the available backbone bandwidth (i.e., add servers to edge server clusters). We observe that the proposed scheme outperforms strategies based on no caching and on full caching of the media asset. It is also better than pure prefix caching [9] for medium-to-high request rates (log-log scales). Similar behaviors are observed for different α/γ ratios, the curves being however translated along the λ axis.

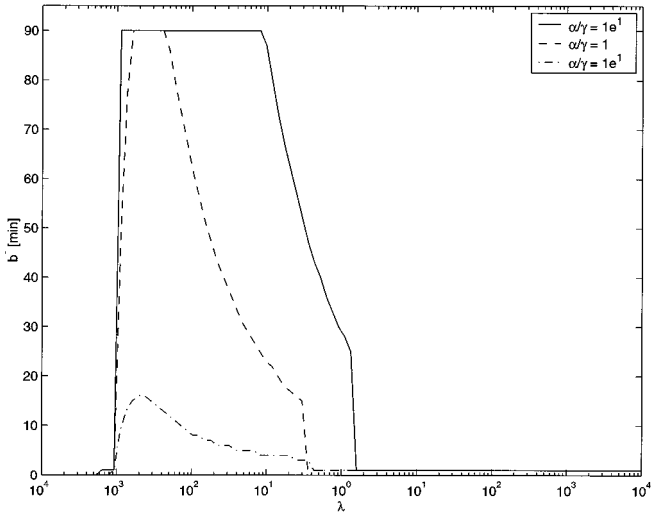

Fig. 4. Optimal batching period b as a function of the request rate λ and the network/disk cost ratio α/γ ($\beta/\alpha = 10^{-3}, T = 90$ min, $r = 5$ Mbps).

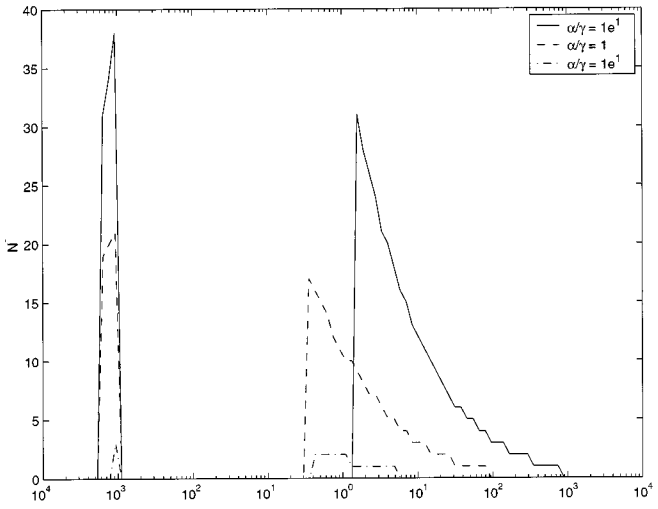

Fig. 5. Optimal batching window size N as a function of the request rate λ and the network/disk cost ratio α/γ ($\beta/\alpha = 10^{-3}, T = 90$ min, $r = 5$ Mbps).

Figs. 4 and 5 show the optimized batched patch caching strategy as a function of the request probability, for different relative weighting of the network and disk bandwidth costs. We see that batched patch caching is the optimal strategy for low request rates. When the request rate increases, it becomes cheaper to perform prefix caching (no patching, $N = 0$) or eventually full caching if the network bandwidth cost is important (i.e., α larger than γ). Recall indeed that for large values of N, the cost of disk writing bandwidth becomes important and may eventually balance out the advantage of patching. However, for large request rates, batched patch caching becomes the optimal strategy again as it minimizes the network cost. Finally for very large λ's, the disk bandwidth cost prevails and our scheme reduces once again to simple prefix caching (see also Fig. 3).

Fig. 6 shows the evolution of the cost value when the required client buffer size B_c is limited. The cost diverges from its value without constraint when the optimal solution consists in neither no caching nor full caching (i.e., $\lambda = 10^{-1}$). Interestingly the cost remains close to the optimum for client buffers as low as a tenth of the media asset size. Finally, even for very small client buffers (i.e., one minute worth of storage), our strategy remains an order of magnitude less expensive than full caching.

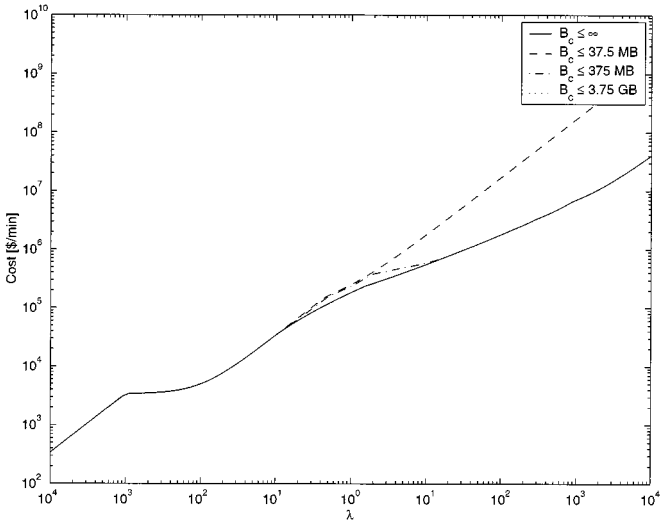

Fig. 6. Optimized Batched Patch Caching Cost as a function of the request rate λ, for different client buffer size ($\beta/\alpha = 10^{-3}, \gamma/\alpha = 10^{-1}, T = 90$ min, $r = 5$ Mbps).

IV. Conclusion

We presented a joint scheduling and partial caching strategy for streaming media. We proposed an optimization framework under which our strategy demonstrated its flexibility. Our scheme indeed optimally adapts itself to various request rates by reducing to simpler schemes such as no caching, full caching and prefix caching with no patching. The potential of batched patch caching is particularly interesting for medium request rates where the reduction in backbone bandwidth is significant compared to the incurred disk utilization.

References

[1] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true on-demand services," in *Proc. ACM Multimedia*, Sept. 1998, pp. 191–200.

[2] S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal patching scheme for efficient multimedia streaming," in *Proc. Int. Conf. on Network and Operating System Support for Digital Audio and Video*, June 1999.

[3] Y. Cai, K. Hua, and K. Vu, "Optimizing patching performance," in *Proc. ACM/SPIE Multimedia Computing and Networking Conf.*, vol. 3654, Jan. 1999, pp. 204–215.

[4] P. P. White and J. Crowcroft, "Optimized batch patching with classes of service," *ACM Commun. Rev.* , vol. 30, no. 4, Oct. 2000.

[5] O. Verscheure, C. Venkatramani, P. Frossard, and L. Amini, "Joint server scheduling and proxy caching for video delivery," *Computer Commun.*, vol. 25, no. 4, pp. 413–423, Mar. 2002.

[6] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proc. IEEE Infocom*, vol. 3, 1999, pp. 1310–1319.

[7] Y. Wang, Z.-L. Zhang, D. Du, and D. Su, "A network conscious approach to end-to-end video delivery over wide area networks using proxy servers," in *Proc. IEEE Infocom*, vol. 2, 1998, pp. 660–667.

[8] S.-H. Chan and F. Tobagi, "Distributed servers architecture for networked video services," *IEEE/ACM Trans. Networking*, vol. 9, pp. 125–136, Apr. 2001.

[9] S.-H. G. Chan, "Operation and cost optimization of a distributed servers architecture for on-demand video services," *IEEE Commun. Lett.*, vol. 5, pp. 384–386, Sept. 2001.