# Operation and Cost Optimization of a Distributed Servers Architecture for On-Demand Video Services

S.-H. Gary Chan, *Member, IEEE*

***Abstract*—Distributed servers architecture offers storage and streaming scalabilities for video services. In this letter, we propose and analyze an on-demand scheme in which the local servers store the beginning portion (i.e., the "prefix") of videos and deliver it by means of unicast streams to the clients. The clients are able to merge onto on-going multicast streams delivered from the repository by means of their set-top buffers. Given a certain limited repository (and thereof multicast) bandwidth, we investigate how the total cost of the local servers can be minimized. We show that if the local storage is the main cost, the size of the prefixes is likely to increase with the video popularity. On the other hand, if the server cost mainly comes from streaming capacity, the size of the prefixes is likely to decrease asymptotically with the video popularity.**

***Index Terms*—Client buffering, distributed servers architecture, multicasting, on-demand video services, server caching.**

## I. Introduction

ADVANCES in broadband networking has made the delivery of video to the home a reality [1]. In a video system, repository servers such as libraries or jukeboxes (collectively referred to as a repository in this paper) are used to store all the videos of interest to users distributed in a network. In order to accommodate a large number of concurrent users, requests for the same video can be served with a single multicast stream. In spite of this, for a large user pool, the streaming requirement of the repository can still be high. Furthermore, since the repository may not be co-located with the users, the network transmission cost may be high. Distributed servers architecture is hence proposed to address the above problems [2]. In this system, local or satellite servers are placed close to user clusters so as to pre-buffer or pre-cache videos according to their local demand. In this way, by increasing the number of repository servers, we achieve scalable storage; and by increasing the number of local servers, we achieve scalable streaming capacity.

Clearly, there is a tradeoff between the limited repository bandwidth and the cost of local servers (consisting of their total storage and streams)—the lower the available repository bandwidth is, the higher is the cost of local servers. It is therefore important to address how to minimize the cost of the local servers given a certain (limited) repository bandwidth. In this paper, we consider an operation of such a distributed servers architecture. Our results show that, if storage is the main cost of a server, larger caches should be allocated to those more popular videos. On the other hand, if streaming is the main cost of a server, smaller caches should be allocated to those more popular videos. While most of the previous work on video services focuses on either client buffering (see, for examples, [3]–[10]) or server caching (see, for examples, [2], [11]), we consider for the first time how server caching and client buffering can be jointly combined in a distributed servers architecture for on-demand (i.e., zero-delay) service, and its cost optimization issues given a certain target arrival rate that the system is designed under.

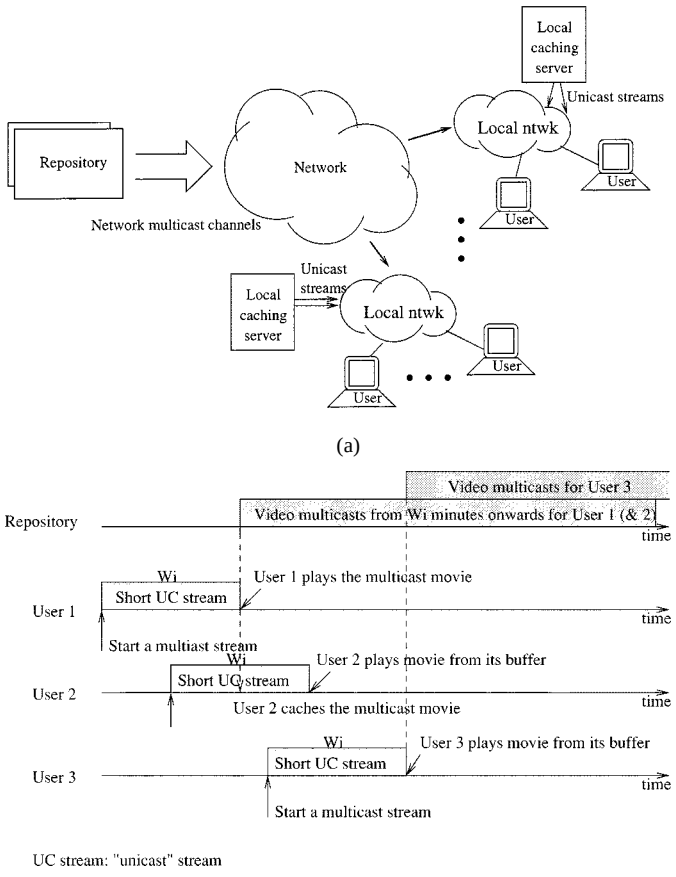


Fig. 1. The system (a) and operation (b) for a distributed servers architecture with local caching servers and client buffering.

## II. Distributed Servers Architecture

### A. System Description

We show in Fig. 1(a) a distributed servers architecture consisting of a repository, a multicast-capable network, and some local servers. The local servers store the beginning portions (i.e.,

Manuscript received April 12, 2001. The associate editor coordinating the review of this letter and approving it for publication was Dr. J. Choe. This work was supported by the Areas of Excellence (AoE) Scheme on Information Technology funded by the University Grant Council in Hong Kong.

The author is with the Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (e-mail: gchan@cs.ust.hk).

Publisher Item Identifier S 1089-7798(01)08456-3.

the "prefixes") of the videos and each client has a buffer of limited size. In order to provide on-demand service, the local server unicasts the prefixes of the videos to the users upon their arrival. While receiving (and playing) its unicast stream, the client can use its buffer to cache an on-going stream multicast from the repository. Once the play-point of the video is available from the client's buffer, the unicast stream from the local server can then be relinquished and the client is said to be "merged" with the multicast stream. Therefore, the unicast streams as provided by the local servers are "transient" in nature and of short duration.

We consider that there are enough streams in both the local servers and the network so that the probability of running out of such streams is negligible (a requirement for on-demand service). Under this condition, the servicing of a video is independent of the servicing of the other videos, and we can focus our discussion on a particular video $i$ with length $T_{h,i}$ minutes. We show the operation of the system considered in this paper in Fig. 1(b). Let $W_i$ be the prefix length for the video, which also corresponds to the buffer allocated at the user's set-top box. All the three users arriving into the system are first served by their respective local servers for a time of $W_i$ minutes. The first user after the start of a video multicast initiates (or "pulls") a new multicast from the repository at a time $W_i$ minutes later (as the maximum amount of data that a client can prebuffer is $W_i$ minutes). The multicast from the repository streams from video segment $W_i$ minutes onwards. Clearly, while the other clients are being served by the unicasts from their respective local servers, the multicast would be started which the clients cache for their later use.

### B. Analysis

In this section, we analyze the system in terms of the storage and streaming requirements at the local servers and consider how the cost can be minimized, given a certain repository bandwidth (which also corresponds to the multicast channels used). Let $N_v$ be the number of videos in the system, and $S$ be the total number of local servers. Requests for video $i$ arrive at the local server $s$ ($1 \le s \le S$) according to a Poisson process with rate $\lambda_{i,s}$ req/min. By Little's formula, the unicast streams required at the local server $s$ is clearly given by

$$U_{i,s} = \lambda_{i,s} W_i. \tag{1}$$

We further define the aggregate request rate for video $i$ as $\lambda_i = \sum_{s=1}^{S} \lambda_{i,s}$, and the overall external request rate as $\Lambda = \sum_{i=1}^{N_v} \lambda_i$.

We next derive the repository bandwidth required for video $i$ given $W_i$. The average interval between successive multicast channel allocation is given by $W_i + 1/\lambda_i$ (as the average interarrival time of requests is $1/\lambda$). Since each multicast channel is held for a duration of $T_{h,i} - W_i$ minutes, by Little's formula, the number of multicast streams (and hence the repository bandwidth) required for the video is given by

$$M_i = \frac{T_{h,i} - W_i}{W_i + 1/\lambda_i}. \tag{2}$$

In general the total server cost is a function of its total storage given by $C = \sum_{s=1}^{S} \sum_{i=1}^{N_v} W_i = S \sum_{i=1}^{N_v} W_i$, and its total streaming given by $U = \sum_{s=1}^{S} \sum_{i=1}^{N_v} U_{i,s} = \sum_{i=1}^{N_v} \lambda_i W_i$. Denote $f(C, U)$ as such cost function.[1] Our cost optimization problem is hence to find $W_i^*$ such that $f(C, U)$ is minimized, subject to a certain constraint on repository bandwidth $\sum_{i=1}^{N_v} M_i \le N_c$, for some value of $N_c$. This problem can be solved by considering the Lagrangian function $L$

$$L = f(C, U) + g\left(\sum_{i=1}^{N_v} M_i - N_c\right) \tag{3}$$

where $g$ is the Lagrangian multiplier depending on the system parameters. $W_i^*$ can then be solved by setting $\partial L / \partial W_i = 0$ and using the constraint $\sum_i M_i = N_c$. As illustrative examples, we consider two simple cases in which the server cost mainly comes from either storage or streaming and can be modeled by a linear function, i.e., either $f(C, U) = C$ or $f(C, U) = U$ (the proportionality constant has been absorbed without loss of generality).

For storage optimization and the case of general interest $T_{h,i} \gg W_i$, the Lagrangian function can be approximated by

$$L \approx S \sum_{i=1}^{N_v} W_i + g\left(\sum_{i=1}^{N_v} \lambda_i T_{h,i} \Big/ (1 + \lambda_i W_i) - N_c\right)$$

which yields

$$W_i^* = \sqrt{K_i} - \frac{1}{\lambda_i} \tag{4}$$

where $K_i \equiv g T_{h,i}$ and $g = (\sum_i \sqrt{T_{h,i}}/N_c)^2$. Note that if the movie lengths are roughly the same, the equation indicates that, in order to minimize the (storage) cost of the local servers, the prefix length should increase with video popularity. This is expected because keeping a larger prefix for those popular videos leads to more significant decrease in the repository bandwidth requirement.

Regarding the optimization of streaming cost and considering the general case of interest $T_{h,i} \gg W_i$, it is not difficult to show that, by following similar steps as above,

$$W_i^* = \sqrt{\frac{K_i'}{\lambda_i}} - \frac{1}{\lambda_i} \tag{5}$$

where $K_i' \equiv g' T_{h,i}$ and $g' = (\sum_i \sqrt{\lambda_i T_{h,i}}/N_c)^2$. The equation says that if the movie lengths are roughly the same, as opposed to the case of storage minimization, the prefix length decreases asymptotically with video popularity. This is again expected since keeping a smaller prefix for popular videos decreases the local streaming requirement.

## III. Illustrative Numerical Results

We consider a system in which the access probability for video $i$ is given by $\nu_i$, where $\nu_i$ is Zipf-distributed, i.e., $\nu_i \propto 1/i^{\zeta}$, for some parameter $\zeta$. Therefore, the request rate for video

[1]Note that we factor in the cost of the repository by adding it into $f(C, U)$.

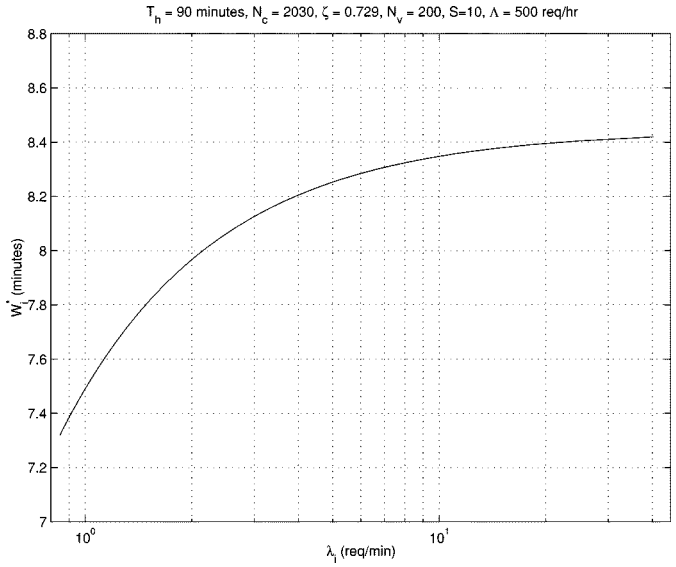


Fig. 2. $W_i^*$ vsersus $\lambda_i$ for the distributed servers architecture, with the local storage minimized.

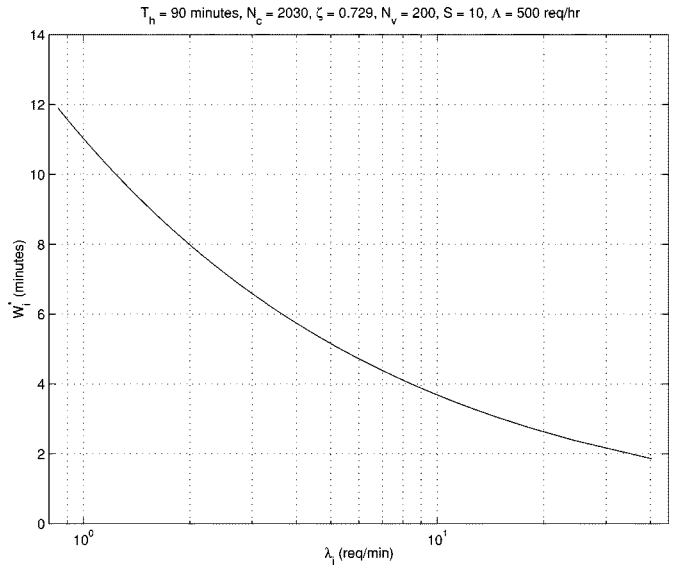


Fig. 3. $W_i^*$ versus $\lambda_i$ in the distributed servers architecture, when streaming is the main server cost.

$i$ is given by $\lambda_i = \nu_i \Lambda$ req/min. As a baseline, we consider a system with $\Lambda = 500$ req/min, $N_v = 200$, $\zeta = 0.729$ [8], [9], $T_{h,i} \equiv T_h = 90$ minutes for all videos, $N_c = 2030$, and $S = 10$.

We first consider the case that the storage cost is minimized. We show in Fig. 2 the optimal prefix length $W_i^*$ minutes for video $i$ (corresponding also to the client buffer size) with respect to $\lambda_i$. As noted before, the more popular a video is, the larger is its $W_i^*$, which flattens off as $\lambda_i$ increases (the graph terminates at the lowest and highest video request rate in the system). Clearly the client buffer does not have to be large (less than 10 minutes in this case). With respect to minimizing the cost of local streaming, we show in Fig. 3 $W_i^*$ vs. $\lambda_i$. As opposed to the previous case, $W_i^*$ now decreases with video popularity. Similar to the previous case, the client buffer does not have to be large ($\leq$20% of the video length).

## IV. Conclusions

We have studied in this paper the use of distributed servers architecture to achieve both storage and streaming scalabilities. We consider an on-demand system in which the local servers store the beginning portion (i.e., the "prefix") of the videos and serve users on-demand using short unicast streams. The remainder of the video is delivered from the repository in a multicast manner. By means of buffering, the clients merge onto the ongoing multicast streams. We have analyzed an operation of the system, and investigated how the total cost of the local servers can be minimized given a certain repository (or multicast) bandwidth. We find that if the cost of the local servers mainly comes from storage, the prefix length should increase with its popularity (according to $\sqrt{K} - 1/\lambda$ for linear cost and uniform movie length, where $K$ is a constant depending on system parameters and $\lambda$ is the request rate of the video). On the other hand, if the server cost mainly comes from streaming, the prefix size decreases asymptotically with video popularity (according to $\sqrt{K'/\lambda} - 1/\lambda$ for linear cost and uniform movie length, where $K'$ is another constant).

## References

[1] V. O. K. Li and W. Liao, "Distributed multimedia systems," *Proceedings of the IEEE*, vol. 85, pp. 1063–1108, July 1997.

[2] S.-H. G. Chan and F. A. Tobagi, "Caching schemes for distributed video services," in *Proc. 1999 IEEE Int. Conf. on Communications (ICC'99)*, Vancouver, Canada, June 1999, pp. 994–1000.

[3] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *Proc. ACM Multimedia 98*, New York, USA, Sept. 14–16, 1998, pp. 191–200.

[4] S. Sen, L. Gao, J. Rexford, and D. Towsley, "Optimal patching schemes for efficient multimedia streaming," in *Proc. NOSSDAV'99*, 1999.

[5] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," in *Proc. ACM Multimedia 99*, Orlando, FL, Oct. 30–Nov. 5, 1999, pp. 199–202.

[6] S. W. Carter, D. Long, and J.-F. Pâris, "An efficient implementation of interactive video-on-demand," in *Proc. 8th Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, San Francisco, CA, Aug. 29–Sept. 1, 2000, pp. 172–179.

[7] S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *Multimedia Syst.*, vol. 4, pp. 197–208, Aug. 1996.

[8] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "Design and analysis of permutation-based pyramid broadcasting," *ACM/Springer Multimedia Systems*, vol. 7, no. 6, pp. 439–448, 1999.

[9] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," *ACM Computer Commun. Rev.* , vol. 27, pp. 89–100, Oct. 1997.

[10] L. Gao, J. Kurose, and D. Towsley, "Efficient schemes for broadcasting popular videos," in *Proc. NOSSDAV'98*, Cambridge, U.K., July 1998.

[11] W. Liao and V. O. K. Li, "The split and merge protocol for interactive video-on-demand," *IEEE Multimedia Mag.*, pp. 51–62, Oct.–Dec. 1997.