Optimized Regional Caching for On-Demand Data Delivery *

Derek L. Eager Michael C. Ferris Mary K. Vernon

University of Saskatchewan
Saskatoon, SK Canada S7N 5A9
eager@cs.usask.ca

University of Wisconsin – Madison
Madison, WI 53706
{ferris, vernon}@cs.wisc.edu

ABSTRACT

Systems for on-demand delivery of large, widely-shared data can use several techniques to improve cost/performance, including: multicast data delivery, segmented data delivery, and regional (or proxy) servers that cache some of the data close to the clients. This paper makes three contributions to the state-of-the-art design of such systems. First, we show how segmented multicast delivery techniques, in particular the recently proposed high-performance *dynamic skyscraper* scheme, can be modified to allow each object to be *partially* or fully cached at regional servers. The new partitioned delivery architecture supports shared delivery between the regional and remote servers *and* improves performance even if one server delivers the entire object. The second contribution is an analytic model that can be solved to determine the full/partial object caching strategy that minimizes delivery cost in the context of a system that has homogeneous regional servers. Finally, results in the paper illustrate the use of the model and provide insight into how the optimal caching strategy is influenced by key system and workload parameters, including client request rate, the relative severity of the disk bandwidth and storage capacity constraints at the regional servers, and the relative costs of regional and remote delivery. Two important conclusions from the results are: (1) it is often cost-effective to cache the initial segments of many data objects rather than the complete data for fewer objects, and (2) the partitioned delivery architecture and caching partial objects can each greatly reduce delivery cost.

Keywords: proxy caching, skyscraper delivery, multicast, optimization

1. INTRODUCTION

This paper considers optimized regional (or proxy) caching strategies for the web and other on-demand data delivery systems that have two key features. First, the objects are sufficiently large and popular that it is cost-effective to use multicast or broadcast delivery methods. Such objects might include, for example, news clips, television shows, medical or recreational information services, popular product advertisements, or successful distance education content. Note that even if the network does not support true multicast, simulated multicast can still be used to conserve server resources such as disk bandwidth. Second, the system uses a segmented delivery technique in which each object is divided into fixed increasing-sized segments that are multicast separately. A client, or an agent acting on behalf of the client, can receive a small number of segments simultaneously, and can buffer segments that are received ahead of need. This greatly increases the (cost-)sharing of the larger segment multicasts, which in turn greatly reduces the server and network bandwidth required to support a given client workload.[19,20,1,13,10,12] Segmented delivery techniques were originally proposed for real-time delivery of video and other continuous media objects, but could also be employed for delivery of other large popular objects that do not have real-time constraints. We use the term multicast in the remainder of this paper to denote either multicast (e.g., via the Internet) or broadcast (e.g., via satellite or regional cable network).

The recently proposed dynamic skyscraper technique can provide true "on-demand" data delivery and has other cost/performance advantages over previously proposed segmented (as well as non-segmented) delivery schemes.[10] The first question addressed in this paper is how the dynamic skyscraper technique can be modified such that regional servers can cache just the first few segments of an object. Prior work on web caching as well as distributed video-on-demand (VOD) architectures has focused on the problem of determining on which server(s) to cache each entire data object so as to optimize system cost/performance.[16,2,4,6,21,5] However, in segmented multicasts, the

*This research was partially supported by the Natural Sciences and Engineering Research Council of Canada under Grant OGP-0000264, by the Air Force Office of Scientific Research under Grant F49620-98-1-0417, and by the National Science Foundation under Grant ACI-9619019.

initial segments are smaller and are multicast more frequently; thus large savings in remote server bandwidth can be obtained at small storage cost in the regional server. Since there is also greater (cost-)sharing per multicast for the later segments, it may be advantageous to cache the initial segments of many objects and to rely on remote multicast delivery of the later more widely (cost-)shared segments, rather than fully caching fewer (highly popular) objects. In addition, caching the initial segments allows the regional server to provide quick response while hiding the latency of communication with the remote server. Parallel work[17] further shows that caching initial segments can facilitate workahead smoothing for variable bit rate continuous media delivery. We propose a new *partitioned skyscraper architecture* to enable these potential benefits. This partitioned architecture can also be employed to improve dynamic skyscraper performance in the case that the entire object is delivered by a given server. A similar partitioned architecture can also be developed for other segmented delivery techniques.

Given a partitioned delivery architecture in which regional servers can cache partial objects and full objects, the next question addressed in this paper is whether an analytical model can be devised and solved to determine the minimum-cost regional caching strategy for given delivery cost models. For a system with a remote server (i.e., the information source) and a set of regional servers, one would like to answer questions such as:

1. If the regional service provider is different than the remote service provider, which full/partial objects should the regional server cache to minimize the use of the remote server, for a given fixed cost of the regional server?
2. If the organization that owns a remote information source also pays for the regional caching and delivery, how should the objects be partitioned between the remote server and the regional servers to minimize the overall cost of delivery to the end user?

The model postulated in this paper can evaluate such questions for a system with one or more remote information sources and a set of homogeneous regional servers. The model is developed for systems that use the proposed partitioned dynamic skyscraper delivery technique, but it can be adapted to determine the optimal caching strategy in systems with heterogeneous regional servers or systems that use other segmented or non-segmented delivery techniques, including conventional first-come first-serve (FCFS) delivery. In Section 5, we illustrate the use of the model and obtain insight into how the optimal cache content is influenced by object popularity, the relative severity of the constraints on disk bandwidth and storage capacity at the regional servers, and other key system parameters. Two important conclusions from the experiments are: (1) it is often most cost-effective to cache many partial objects at the regional servers, and (2) the new partitioned dynamic skyscraper architecture and the caching of partial objects can each greatly reduce the minimum delivery cost.

Section 2 describes the dynamic skyscraper multicast technique. Section 3 defines the new partitioned dynamic skyscraper architecture that allows regional servers to store partial objects. Section 4 develops the model for determining the optimal cache content for a given relative cost of regional and remote delivery. Section 5 provides some results and insights from applying the model, and Section 6 provides conclusions for this work.

2. BACKGROUND: DYNAMIC SKYSCRAPER MULTICASTS

The parameters of a skyscraper multicast delivery system are defined in Table 1. The term *channel* refers to the entity used for a multicast and to the collection of server and network resources required to support the (multicast or broadcast) transmission of an object segment at a given delivery rate.

Parameter	Definition
C	total number of channels devoted to skyscraper multicasts
K	number of segments per object, and channels per skyscraper multicast
n	number of objects
N	number of groups of skyscraper channels ($N = C/K$)
s_j	size of the j'th segment (relative to the size of the first segment)
T_1	duration of a unit-segment transmission
W	the largest segment size

Table 1. Parameters of a Skyscraper System.

To simplify the system description and to gain initial insights into optimal regional caching strategies, the remainder of this paper assumes that all objects have the same length (i.e., amount of data) and are delivered at the same rate. However, the system and the associated delivery cost model can be modified to handle heterogeneous object lengths and delivery rates.

A key feature of a dynamic skyscraper system[10] is that each object is divided into K segments with a particular progression of relative sizes. The (smallest) initial segment is multicast most frequently, to reduce the time that a client must wait to receive it. Each larger segment is multicast less frequently to reduce bandwidth usage.

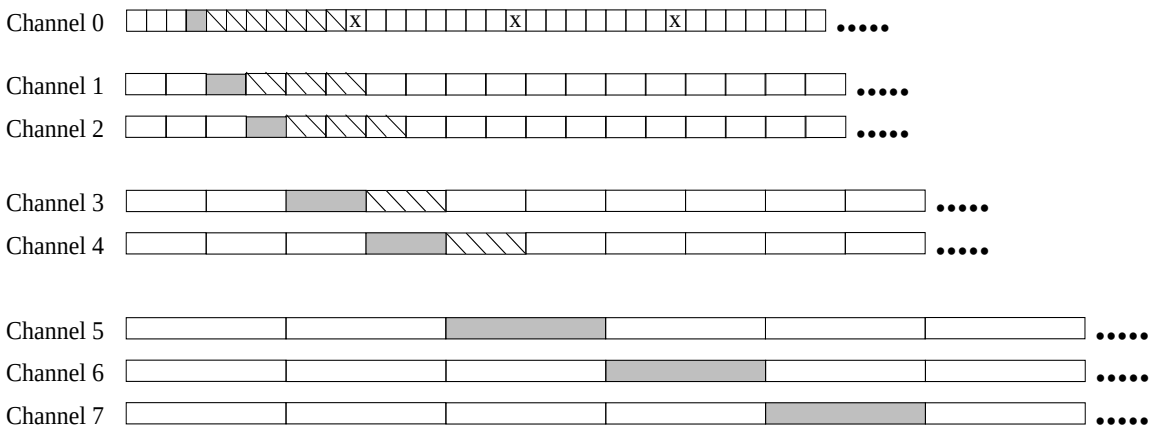


Figure 1. Example Skyscraper Transmission Schedule. ($K = 8$; $W = 8$)

Each segment is transmitted on a different channel according to a schedule such as the one illustrated in Figure 1. In this example, each object is divided into eight segments (i.e., $K = 8$), with relative sizes {1,2,2,4,4,8,8,8}, and the segments are each multicast on separate channels, numbered $0-7$, respectively. For example, a client who receives the gray-shaded transmission on channel 0 will receive the gray-shaded transmission on each of the other channels. This client will have received the entire object once the segment on channel 7 has been delivered.

The segment transmissions for a given object are each repeated on their respective channels a specific number of times, constituting a *transmission cluster*, as identified by the gray and striped shading in the figure. Each transmission period marked with an X on channel 0 begins a new identically-structured transmission cluster which can be scheduled to deliver a new object's segments, in response to client requests. For the given sequence and alignment of relative segment sizes, a client that starts reception during any channel 0 period in a transmission cluster can receive each of the other seven segments of the object, one per channel, with no pauses between segments, requiring simultaneous reception of at most two transmissions by the client. For example, consider a client who requests the object just before the last striped unit-segment transmission on channel 0. This client will receive the last striped transmission on each of channels $0-4$, and then the gray-shaded transmissions on channels $5-7$. All such reception sequences, including the initial reception sequence for the cluster, share the same multicast transmission on channel K.[†]

Many different *relative* segment size progressions are possible.[10,12] Whichever progression is used, it is bounded by a parameter, W, and padded if necessary with W values up to length K, in order to limit the required client storage capacity. Note that a new transmission cluster begins on channel 0 after each W unit-segment transmissions (i.e., after a period of $W \times T_1$), as marked by the Xs in Figure 1. Note also that $W \times T_1$ is the duration of the transmission cluster on each of the K channels in the group. The client buffer space requirement can be derived from Figure 1 by observing that clients who begins in the last unit-segment transmission period for a cluster will need to buffer $W-1$ units of data once they start receiving the transmission on channel K which is shared with the clients who started $W-1$ units earlier. This buffering capability is easily accommodated by a commodity disk.[10]

The C channels that are provided for multicasting objects are organized into N groups of K channels each. The transmission clusters in the different groups of channels are *persistently staggered* such that a new transmission cluster starts on a different group at a fixed spacing of $\frac{W \times T_1}{N}$.

[†] The delivery technique is named "skyscraper" due to the shape that is formed by stacking the segment sizes one above the other.

If a client requests an object and it is not possible to join an on-going or scheduled transmission cluster for the object, the object is scheduled for delivery on the next available transmission cluster that will be multicast in the future. Requests that require a new transmission cluster are scheduled in first-come first-serve (FCFS) order because recent results show that for fixed length objects, FCFS outperforms other proposed scheduling algorithms including the *maximum factored queue length first* (MFQ) discipline if both the mean and the variability in client waiting time are considered.[18] Furthermore, the cluster assignment can be done when the request arrives, and thus the system can immediately inform the client when the multicast will begin.

The cost/performance of the dynamic skyscraper delivery technique is further enhanced by temporarily reassigning unused transmission periods to requests that are waiting for a unit-segment multicast in another active transmission cluster. This optimization, called *channel stealing*, and another optimization called *idle cluster catch-up*, significantly reduce average client wait (effecting true on-demand delivery), as illustrated in Figure 2. The details of these optimizations, and the dramatic improvement in performance of dynamic skyscraper delivery over prior segmented and non-segmented delivery techniques, are described in prior work.[10]

3. PARTITIONED SKYSCRAPER ARCHITECTURE

This section considers the application of dynamic skyscraper delivery in a system with regional (or proxy) servers that cache some of the data. For simplicity, we describe the system in terms of a single remote server (i.e., information source) and multiple regional servers. The extension to multiple remote servers is straightforward.

In one scenario, the information source might transmit via satellite and the regional servers might each transmit requested satellite content as well as locally-cached content via a cable network. In such a scenario, regional delivery of initial segments can be tightly synchronized with the remote delivery of later segments of the same object. With appropriate multicast support, or perhaps with simulated multicasts, the proposed partitioned dynamic skyscraper architecture can also be employed when objects are delivered over the Internet or other commodity networks. For this context, definition of an efficient mapping between channels and multicast groups and the small modifications that are needed to accommodate the uncertainty in network delivery times in the case of real-time playback, is left for future work.

3.1. A Naive Dynamic Skyscraper System

It is relatively straightforward to devise a system with regional caching that uses the dynamic skyscraper delivery technique if each object is either delivered entirely by the remote server or is cached entirely at the regional server. In this case, the regional server might (1) allocate (or account for) regional bandwidth to be used for remote server multicasts requested by its clients, based on the (projected) transmission time at the remote server, and (2) use its own transmission clusters for multicasts of locally stored objects. The regional server might also need to delay client requests for remote server multicasts and/or dynamically adjust the rate at which new regional transmission clusters are allocated, depending on the dynamic client workload and the total available regional network bandwidth.

The situation becomes more complex when considering how the multicasts might be organized if the initial k segments of some objects are cached at the regional server and the remainder of those objects are delivered by the remote server.[‡] With partial object caching, a shared implementation of dynamic skyscraper is required. In the most naive implementation, whenever delivery of a partially cached object is requested, remote and regional servers cooperate to provide a complete transmission cluster on each regional network that might (eventually) need the remote portion of the multicast. Clearly, this might be very wasteful of bandwidth, as it might be the case that the transmission is not used by the clients in a given region. However, if this is not done, then the problem arises as to how to accommodate a new client request for a partially cached object at a regional server that is not carrying (or planning to carry) a current (or scheduled) remote server multicast of the later segments of the object. It may again be wasteful of regional network bandwidth to schedule the entire regional portion of a transmission cluster for the object, since much of the new cluster will not be usable in the case that the remaining time for joining the remote multicast is very short.

Our proposed solution to this problem employs the concept of *decoupled mini*-transmission clusters (or mini-clusters) for the initial segments cached at the regional servers. These mini-clusters are allocated on-demand, allowing new clients to join on-going (or scheduled) remote server multicasts, as discussed next.

[‡]Note that, for a given object, it is always more cost-effective to cache an earlier segment than a later segment of the object, due to the relative size and multicast frequency of the segments.

3.2. A Cost-Effective Partitioned Dynamic Skyscraper Architecture

The definition of mini-transmission clusters is enabled by the recursive structure of dynamic skyscraper transmission clusters. For example, the portion of the transmission cluster in Figure 1 that is carried on channels 0 through 4 is composed of two consecutive smaller transmission clusters with $K = 5$ and $W = 4$, while the portion carried on just channels 0 through 2 is composed of four consecutive smaller transmission clusters with $K = 3$ and $W = 2$.

A mini-transmission cluster is defined as one of these smaller clusters, with the K parameter (denoted by k) equal to the number of segments stored regionally and the W parameter (denoted by w) equal to the (relative) size of the longest segment stored regionally. The basic idea is to deliver just a single instance of the mini-cluster in response to a user request, rather than the multiple repetitions of which a complete transmission cluster would normally be composed. The multicasts of the remaining segments by the remote server retain the same delivery structure (for those segments) that is illustrated in Figure 1. In the following, we denote the segments delivered by mini-transmission clusters the *leading segment set*, and the remaining segments the *trailing segment set*.

As for full transmission clusters (with K segments), the channels for mini-cluster transmissions at each regional server are organized into groups of k channels each, and the channels for trailing-segment transmissions at the remote server are organized into groups of $K - k$ channels each. The clusters in the different groups at each server are persistently staggered. For example, at each regional server a new mini-cluster starts on a different group every $w \times T_1$ divided by the number of groups of mini-cluster channels.

A new client request for an object that is not currently being multicast (or scheduled to be multicast) at the regional server or at the remote server, queues at the regional server for a mini-cluster, and queues at the remote server for a trailing segment set transmission cluster. In this way, the transmissions of the leading and trailing sets of segments are *decoupled*. However, allocation is coordinated, so that the mini-cluster at the regional server is scheduled as early as possible (so as to minimize delay) and the transmission cluster at the remote server is scheduled as late as possible relative to the scheduling of the mini-cluster. This maximizes opportunities for other clients to join the trailing segment multicasts while avoiding any pause in the delivery for the first client.

A new client request for an object that is not currently being multicast (or scheduled for multicast) at the regional server, but is within the *catch-up window* of an in-progress (or scheduled) remote transmission for the trailing segment set, must be allocated (1) a new mini-cluster at the regional server, and (2) regional network bandwidth for the remainder of the remote server transmission cluster.[§]

One further optimization is made possible by the mini-clusters. Consider Figure 1, and suppose that the regional server caches the first five segments. In this case, a mini-cluster can be allocated to a new client request that arrives just before the multicast of the sixth segment by the remote server. This is (six unit periods) later than the last unit-segment multicast on channel 0 in the standard skyscraper transmission cluster as defined in Section 2. This policy maximizes the duration of the *total catch-up window* for the trailing segment set transmission cluster, which can be shown to be of length $(W - s_{k+1} + s) \times T_1$, where s is the sum of the sizes of the segments cached regionally, and s_{k+1} is the size of the first segment delivered by the remote server.

A new client request can be satisfied by an in-progress mini-cluster for the requested object at the regional site, as long as it arrives within the *mini-catch-up window* for that cluster. This catch-up window is at most of length $(w - 1) \times T_1$. However, if the mini-cluster begins closer than $(w - 1) \times T_1$ before the end of the catch-up window of the corresponding trailing segment set transmission cluster, the mini-catch-up window will be smaller, since it is only possible for a request to join an in-progress mini-cluster if it can also join in with the corresponding trailing segment set cluster. The variable size of the mini-catch-up windows is accounted for in the optimization model developed in the next section.

Increasing the size of the total catch-up window implies increases in the client storage requirement and in the number of segment multicasts a client must be able to receive simultaneously. Specifically, the maximum storage requirement (as measured in unit-segments) is $W - s_{k+1} + s$ rather than $W - 1$. For the {1,2,2,4,4,...} segment size progression, if $k = K - 1$ or if the first two segments delivered by the remote server are of the same size, clients must

[§] Note that as long as more than one segment is stored at the regional server, the sum of the sizes of the segments cached regionally is greater than the size of the first segment delivered by the remote server. In this case, the mini-cluster can be scheduled at any point within the catch-up window. In contrast, storing only the first unit-segment of an object regionally requires significantly tighter synchronization between remote and regional servers to avoid pauses in delivery. (Consider the case in which a regional client request arrives just after the beginning of a remote server multicast of the first two-unit segment.) Thus, in the remainder of the paper we assume $k \neq 1$.

be able to receive up to three (rather than two) segments simultaneously; otherwise clients must be able to receive up to four simultaneous segment multicasts.

An important observation is that mini-clusters can also be used when the entire objects are delivered by a given server. The increased size of the catch-up window and the more efficient use of bandwidth for less popular objects suggest that decoupled mini-clusters can lead to improved performance in this case as well as when object delivery is shared between remote and regional servers. The optimization model developed in the next section assumes the use of mini-clusters (of size k segments) for each object, regardless of whether or how it is cached.

4. THE OPTIMIZATION MODEL

This section develops an analytic model that permits calculation of the set of object segments that should be cached at the regional (or proxy) servers in order to minimize delivery cost for a given skyscraper configuration.

The problem is more complex than the problems examined in previous papers that address optimal object allocation in distributed VOD systems, owing to (1) the need to capture the behavior that occurs with segmented multicast delivery, in which clients will join (and cost-share) dynamically-scheduled segment multicasts, as in the partitioned dynamic skyscraper system, and (2) the possibility that objects may be partially cached at the regional servers. Specifically, the model must express a suitable measure of system performance for any set of allowed segment allocation decisions, in the presence of the fixed-size total catch-up windows and variable-size mini-catch-up windows described in Section 3. Nevertheless, we are able to develop a fairly simple optimization model by focusing on the required remote and regional bandwidth required to support a given assignment of objects and client workload, rather than on more detailed performance measures such as average client delay or blocking probability. Note that the required regional *network* bandwidth is the same regardless of whether object segments are cached regionally or whether they are delivered from the remote server. However, the regional *server* bandwidth, and the remote server and network bandwidth are a function of the caching decisions. Candidate object distributions can thus be compared with respect to the impact they have on the predicted bandwidth requirements.

We desire an estimate of required bandwidth that is easy to compute, and that corresponds to a good operating point. Applying concepts from asymptotic bounds analysis,[14] our estimate for the required number of channels to support a given client workload is the average number of channels that would be in use if an infinite number of channels were available.[¶] As we discuss below, this estimate is readily computed with only minimal statistical assumptions. Furthermore, in a wide variety of contexts, this form of estimate has been found to yield an operating point close to the knee of the cost-performance curve, at which point the achieved performance per unit cost is maximized. As we illustrate in Figure 2 for several dynamic skyscraper systems, the proposed bandwidth estimate is close to the knee of the curve of mean client wait versus the inverse of the number of channels provided in the dynamic skyscraper delivery system. Furthermore, the examples illustrate that mean client wait at the knee is typically not much larger than the minimum achievable for the given skyscraper parameters. (That is, the system operates much like a closed queueing system due to the fact that the queue for new transmission clusters can never be greater than the number of objects.) Since we will be computing optimal regional cache content for a fixed skyscraper configuration (i.e, K, W, and k), the content that minimizes the proposed bandwidth requirement should yield average client wait that is close to minimum.

4.1. Optimal Channel Capacity for Non-Partitioned Systems

The estimate of required bandwidth is developed first for the simpler context of a dynamic skyscraper system that does not have regional servers and does not use mini-clusters to improve performance (i.e., $k = 0$). Letting $C^* = K \times N^*$ denote the estimate of the required number of channels, and λ_i denote the rate of requests for object i, we obtain

$$C^* = K \times N^* = K \times WT_1 \sum_{i=1}^{n} \frac{1}{(W-1)T_1 + \frac{1}{\lambda_i}}$$

The factor WT_1 is the duration of a transmission cluster on each channel. The i'th term in the sum is the inverse of the average time between transmission clusters that deliver object i, assuming requests for a new transmission

[¶] Or, equivalently, the largest number of channels such that any fewer would be guaranteed to result in queueing of client requests for any client request arrival process with the given per-object average request rates.

cluster have zero wait time (or an infinite number of channels are available), and assuming that the average arrival rate of requests for the object when a transmission cluster is not available for catch-up is equal to the overall average arrival rate. (Note that the latter assumption implies that $1/\lambda_i$ is the average time from the end of a transmission cluster catch-up window for object i until the next request for that object.) Thus, the i'th term gives the allocation rate for new transmission clusters for object i if an infinite number of channels is available. Summing over all objects that use skyscraper multicasts gives the total maximum allocation rate, and multiplying this maximum allocation rate by the duration of a transmission cluster gives the average number of groups of channels that would be in use if an infinite number of groups were available. Multiplication by the number of channels per group gives an estimate of the total number of channels that should be provided.[‖]

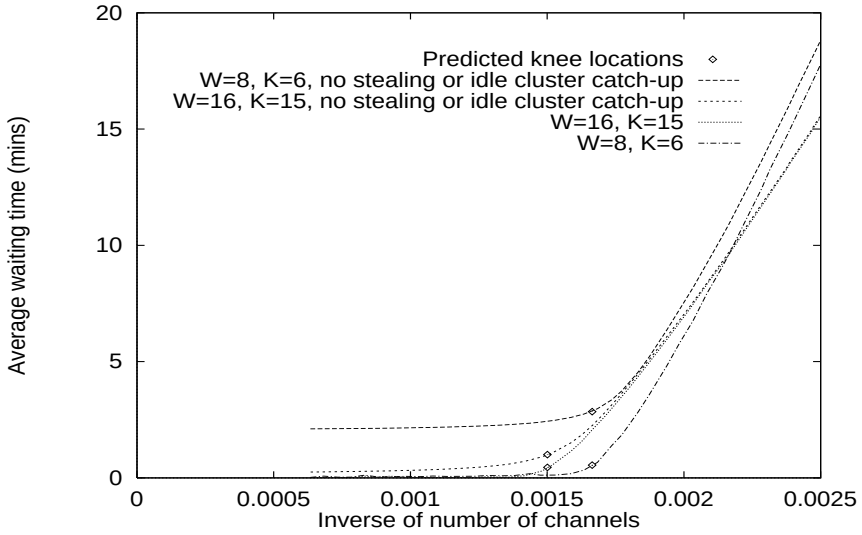


Figure 2. Average Client Waiting Time vs. Inverse of Number of Channels.
(224 120-minute objects; $\lambda = 8.0$)

Figure 2 gives sample results illustrating how close C^* is to the knee of the curve of average client waiting time versus the inverse of the number of channels, for several skyscraper systems.[**] Note that in these and other configurations we have examined (not shown), the C^* estimates are close approximations to the knees of the curves, for the most basic dynamic skyscraper scheme as well as the significantly enhanced scheme that employs channel stealing and catch-up with idle clusters.

4.2. Optimal Object Assignments for Regional Servers

The specific optimization problem considered is that of determining the regional cache contents that minimize the overall cost of delivery, as represented by a weighted sum of the required number of channels at the remote server and at each regional server, subject to constraints on the bandwidth and storage capacity at each regional server.

The optimization model parameters are defined in Table 2. The key model outputs are the θ_i values that specify whether object i should be cached (fully or partially) at the regional servers. It is assumed that all objects have the same segmentation parameters (K, W, k, w), and that each object can have 0, k or K segments stored regionally. It is also assumed that the regional sites are homogeneous in storage and bandwidth capabilities, as well as in client request rates and object selection frequencies, and thus, that all of the regional servers will store the same

[‖] Note that the dependence of C^* on T_1 is removed if the arrival rates are expressed in terms of requests per unit-segment transmission period.

[**] The waiting time curves in Figure 2 were derived using simulation in which the 95% confidence intervals are within 2% of the reported values. The total request arrival rate is eight requests per minute, arrivals are Poisson, and the object selection frequencies are given by the Zipf(0) distribution on 1000 objects. The system contains the most popular 224 objects because, for the modeled object selection frequencies, (1) the popularities of the first 100 objects matches reasonably well with a particular measurement of the 100 most popular objects in a video rental outlet,[7] and (2) 80% of the requests are for the 224 most popular objects, which is a popular definition of the hot set that might use skyscraper delivery.

Input	Parameter Definition
k	number of segments in the leading segment set
K	number of segments per object
n	number of objects
$N_{channels}$	maximum number of channels at each regional server
$N_{segments}$	maximum storage capacity (measured in number of unit-segments) at each regional server
P	number of regional servers
s_j	size of the j'th segment (relative to the size of the first)
T_1	duration of a unit-segment transmission
w	the largest segment size in the leading segment set
W	the largest segment size in the trailing segment set
β	the cost of a regional server channel, relative to that of a remote server channel
λ_i	total arrival rate of requests for object i
Output	**Parameter Definition**
C_{remote}	number of channels needed for remote server multicasts
$C_{regional}$	number of channels needed for regional server multicasts
$D_{regional}$	storage needed at each regional server, in number of unit-segments
$X_i^{l,R}, X_i^{l,Rr}, X_i^{l,r}, X_i^{t,R}, X_i^{t,Rr}, X_i^{t,r}$	maximum rate at which transmission clusters can be allocated for multicasts of the leading/trailing (l/t) segment set for object i, distinguished by whether the segments of the object are only stored at the remote server (R), are partially cached at the regional servers (Rr), or are fully cached at the regional servers (r)
θ_i^R	equals 1 if object i is stored only at the remote server; 0 otherwise
θ_i^{Rr}	equals 1 if only the leading segment set of object i is cached regionally; 0 otherwise
θ_i^r	equals 1 if object i is entirely cached regionally; 0 otherwise

Table 2. Parameters for the Regional Cache Optimization Model.

segments/objects. These assumptions simplify the exploration of the system design space and are thus appropriate for gaining initial insights. The model can easily be modified to include more general formulations in the future.

With the above assumptions, the optimization problem is formally described as follows:

$$\min_{\theta} \quad C_{remote}(\theta) + P\beta C_{regional}(\theta)$$

$$\begin{aligned} \text{subject to} \quad & C_{regional}(\theta) \le N_{channels} \\ & D_{regional}(\theta) \le N_{segments} \\ & \theta_i^R + \theta_i^{Rr} + \theta_i^r = 1, \quad i = 1, 2, \ldots, n \\ & \theta_i^R, \theta_i^{Rr}, \theta_i^r \in \{0, 1\}, \quad i = 1, 2, \ldots, n \end{aligned}$$

Here the notation θ represents the vector whose components are θ_i^R, θ_i^{Rr}, θ_i^r, $i = 1, 2, \ldots, n$.

Solution of this model requires a method of computing the number of channels needed at the remote server as well as at each regional server, as a function of the client workload (i.e., request arrival rates for each object), the partitioned skyscraper configuration, and the object segments cached at the regional servers. For this purpose, we use the same basic approach that was used in Section 4.1, entailing finding the maximum allocation rates of new (mini- and trailing segment) transmission clusters, assuming an infinite number of channels is available.

The maximum allocation rate for new transmission clusters in a system with no regional caching and no mini-clusters is the inverse of the average time between requests for a new transmission cluster when there is no queueing, as given by $(W-1)T_1 + \frac{1}{\lambda_i}$. The maximum allocation rates for trailing segment set transmission clusters in the partitioned dynamic skyscraper architecture are given by similar expressions that depend on whether the transmission cluster is delivered by the remote or regional server. Recall from Section 3 that if the multicast uses mini-clusters of size k segments, the duration of the total catch-up window (for trailing segment set clusters) is $W - s_{k+1} + \sum_{j=1}^{k} s_j$.

Noting that λ_i/P is the arrival rate of client requests for object i at a regional server, the maximum allocation rates for trailing segment set clusters are as follows:

$$X_i^{t,R} = X_i^{t,Rr} = \frac{1}{\left(W - s_{k+1} + \sum_{j=1}^{k} s_j\right) T_1 + \frac{1}{\lambda_i}}$$

$$X_i^{t,r} = \frac{1}{\left(W - s_{k+1} + \sum_{j=1}^{k} s_j\right) T_1 + \frac{P}{\lambda_i}}$$

The maximum allocation rates for mini-clusters are further complicated by the fact that the mini-catch-up window for such a cluster has a variety of possible lengths, depending on when the mini-cluster begins in relationship to the end of the catch-up window of the corresponding trailing segment set transmission cluster. To compute the average catch-up window size for mini-clusters for object i, we make two assumptions. First, we assume that the average arrival rate of mini-cluster requests for object i during the last wT_1 of the catch-up window of an object i trailing segment set transmission cluster is equal to the overall average arrival rate of mini-cluster requests for that object. This implies that the fraction of mini-clusters for object i that will begin during this time period is given by wT_1 times the allocation rate of object i trailing segment set transmission clusters. Second, we assume that such arrivals may occur anywhere within this time period with equal probability. Thus, the average catch-up window size of the corresponding mini-clusters is given by $\frac{w-1}{2}T_1$. All other mini-clusters have a full catch-up window of size $(w-1)T_1$. The three mini-cluster allocation rates are therefore given by the following equations:

$$X_i^{l,r} = \frac{1}{\left(X_i^{t,r} wT_1 \frac{w-1}{2} + (1 - X_i^{t,r} wT_1)(w-1)\right) T_1 + \frac{P}{\lambda_i}}$$

$$X_i^{l,Rr} = \frac{1}{\left(X_i^{t,Rr} wT_1 \frac{w-1}{2} + (1 - X_i^{t,Rr} wT_1)(w-1)\right) T_1 + \frac{P}{\lambda_i}}$$

$$X_i^{l,R} = \frac{1}{\left(X_i^{t,R} wT_1 \frac{w-1}{2} + (1 - X_i^{t,R} wT_1)(w-1)\right) T_1 + \frac{1}{\lambda_i}}$$

As in the simpler system model of Section 4.1, multiplying each of the above allocation rates by the duration of, and number of channels in, a transmission cluster of the corresponding type, yields an estimate for the number of channels required for that particular type of transmission cluster. For specific object segment allocations, as reflected through the θ_i values, the required number of channels and the required regional storage can thus be computed as follows:

$$C_{remote}(\theta) = \sum_{i=1}^{n} (\theta_i^R X_i^{t,R} + \theta_i^{Rr} X_i^{t,Rr})(K-k) W T_1 + \theta_i^R X_i^{l,R} k w T_1$$

$$C_{regional}(\theta) = \sum_{i=1}^{n} \theta_i^r X_i^{t,r} (K-k) W T_1 + (\theta_i^r X_i^{l,r} + \theta_i^{Rr} X_i^{l,Rr}) k w T_1$$

$$D_{regional}(\theta) = \sum_{i=1}^{n} \left((\theta_i^r + \theta_i^{Rr}) \sum_{j=1}^{k} s_j + \theta_i^r \sum_{j=k+1}^{K} s_j \right)$$

The above equations estimate the required number of channels (or disk I/O and network I/O bandwidth) at each type of server, assuming the network supports multicast or broadcast delivery. This is also the number of network channels required for the remote (or regional) multicasts if the respective server is operating over a broadcast network such as a satellite or cable network. The calculations are somewhat imprecise for the required network channels if the servers are operating over a switched network, such as the Internet, because the (average) bandwidth needed for the multicast depends on the (average) number of clients receiving the multicast, and this in turn may depend on the object that is being transmitted. Recall that the required regional network bandwidth is not affected by whether the objects are stored at the remote or regional server. Thus, the only extension that is needed to make the model precise for a switched network is to factor in the average remote network bandwidth required to multicast each object. In the interest of gaining initial insights, we defer this extension to future work.

There are several points worth noting about the model. First, the model is valid for multiple remote servers if each stores a distinct subset of the objects in the system and if the network used by each remote server has the same cost for the same required bandwidth. In this case C_{remote} is the aggregate number of channels that must be provided at the collection of remote servers. Second, the model can be generalized for heterogeneous regional servers by developing separate calculations, similar to those given above, and summing over the appropriate objects, for each distinct regional server. Third, C_{remote}, $C_{regional}$ and $D_{regional}$ are linear functions of the binary variables θ. Thus, the optimization model is a mixed integer linear program (MIP),[15] for which reliable solution techniques exist.

5. RESULTS

In this section we provide results that illustrate the capabilities of the optimization model as a system design tool. The results also yield insight into the form of the optimal caching strategy at the regional servers for various system parameters and cost considerations.

We perform three types of experiments with the model:

1. In Section 5.1, we investigate the optimal strategy when the objective is to minimize the use of remote server channels (i.e., $\beta = 0$), under the constraints of fixed regional server capacity and bandwidth. These results will show that the form of the optimal assignments differs, depending on whether the active constraint is the bandwidth or the capacity of the regional server.

2. In Section 5.2, we examine how the optimal regional cache contents change as the relative cost of the regional channels, β, increases. These results will show that the regional servers should cache segments for less popular objects as β increases.

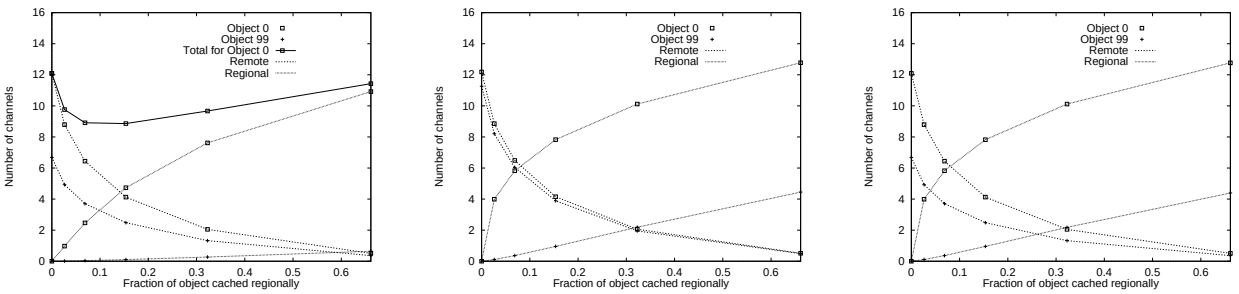
3. In Section 5.3 we show how the allocations and system cost change as the capacity and bandwidth at the regional servers are varied.

All of the results will show a high tendency to cache partial objects. The results will also show that, compared with the minimum cost solution when the regional servers can only cache whole objects and the servers employ the traditional skyscraper architecture, the partitioned skyscraper architecture can result in substantial cost savings, and the capability to cache partial objects can result in further substantial cost savings. Together these two new improvements yield substantial cost savings except when client request rate is very high *and* regional server resources are severely limited.

The optimization model was formulated in an algebraic modeling language, GAMS,[3,11] to enable solution using a variety of MIP solvers. The most effective algorithms currently implemented are branch and bound (branch and cut) methods with subproblems solved using a variant of the simplex method for linear programming.[8] After some experimentation with the model formulation and solvers, we used the XPRESS[9] code with default settings. The GAMS system is appropriate for the speed of model development and analysis required in this research. However, it is possible that specialized algorithms for this problem could be more efficient. Design and implementation of such algorithms might be appropriate for routine use of the model.

Each experiment reported below solves the model to optimality under the assumption that there will be at most one contiguous set of objects that are fully cached at the regional servers. This assumption was enforced by adding the following constraint to the model:

$$\sum_i \max\left(\theta_i^r - \theta_{i-1}^r, 0\right) \leq 1.$$



(a) $\lambda' = 10$, $P = 10$ (b) $\lambda' = 100$, $P = 10$ (c) $\lambda' = 10$, $P = 1$

Figure 3. C_{remote} and $C_{regional}$ for Objects 0 and 99 vs. Fraction Cached.

This inequality, which greatly reduces the model solution time, was only added after substantial experimentation with the model showed it was always satisfied by the optimal solutions.

All experiments in this paper assume a system with: (1) 100 objects having selection frequencies modeled by the Zipf(0) distribution, (2) $K = 12$, (3) relative segment size progression of {1,2,2,4,4,8,8,...}, and (4) $W = 64$. For this segment size progression and these values of K and W, the total length of each object is 189 unit-segments. In order for the results to be independent of object size (in bytes) and delivery time, we define client request arrival rate, λ', in terms of requests *per unit-segment multicast period*[††], and regional server capacity, $N_{segments}$, in terms of the number of unit-segments that can be cached. Note that the unit-segment transmission time, T_1, is equal to the total time to deliver the object divided by 189.

To aid in interpreting the results, each graph in Figure 3 shows the estimated number of remote server channels and regional server channels that should be provided for the most popular object (object 0) and least popular object (object 99) as a function of the fraction of the object that is cached at the regional server. The x-axis in each graph ranges from 0 to 66% because this represents the range of 0 to 11 segments of the object when $K = 12$ and $W = 64$. Key observations from these figures are:

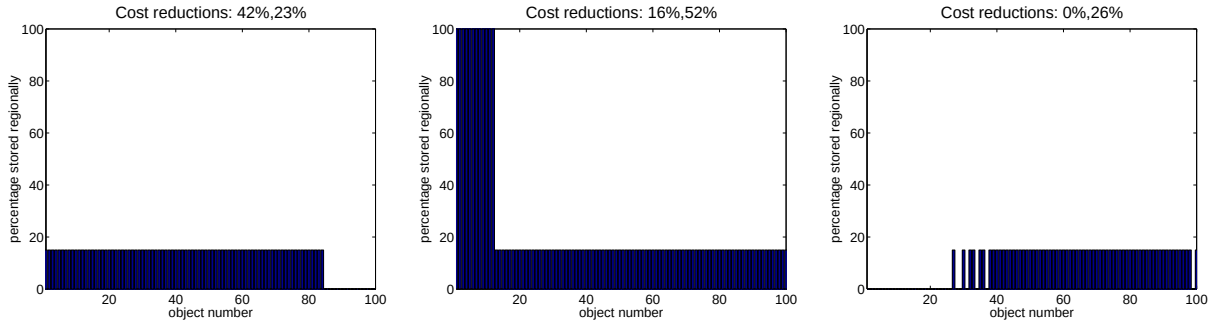
- Caching the initial segments of each object leads to greater decrease in remote bandwidth per unit of storage than caching the later segments.
- The most popular object has the greatest decrease in remote server bandwidth and greatest increase in regional server bandwidth as a function of the fraction cached.
- As client request rate (λ') increases, the differential in remote server channels required for object 0 versus object 99 decreases. If P is fixed and greater than one, the differential in required regional server channels also decreases, but more gradually.
- When $P = 1$, the total number of channels required is minimum for some $k > 0$. Thus, the new transmission mini-clusters proposed in this paper improve the cost/performance ratio even when used within a single server.

We have computed the minimum-cost regional cache contents for each experiment below for $k = 0$, 3, 5, 7, and 9. The caching strategy as a function of the other system parameters is similar for each k, and since $k = 7$ achieves the lowest delivery cost in almost every case, we provide the results for $k = 7$. Note that the first seven segments represents 29/189 or about 15% of the total object length.

For each experiment, we also express the regional server storage capacity and bandwidth constraints in units of a "baseline server capability", where one baseline unit is $N_{channels} = 192$ and $N_{segments} = 2436$[‡‡]. Note that 2436

[††]As noted in section 4, substitution of $\lambda = \lambda'/T_1$ in the model equations removes the dependence of the cost on the value of T_1.

[‡‡]These baseline constraints might represent, for example, that the server contains four commodity disks each with capacity to store 4.35 gigabytes of data, the objects have total size equal to 1.35 gigabytes (e.g., two-hour MPEG-1 encoded videos), and delivery rate is 1.5 million bits per second. The constraints are also valid, for example, if the server contains three times as many disks and the objects are two-hour MPEG-2 encoded videos.



(a) $\lambda' = 10$, $P = 10$, baseline (cap) (b) $\lambda' = 10^2$, $P = 10$, 2×base (cap) (c) $\lambda' = 10^3$, $P = 10$, 2×base (b/w)

Figure 4. Regional Cache Content that Minimize C_{remote} ($\beta = 0$, $k = 7$)
baseline: $N_{channels} = 192$, $N_{segments} = 2436$
(cap) indicates the regional storage capacity constraint is active at the solution;
(b/w) indicates the regional bandwidth constraint is active at the solution.

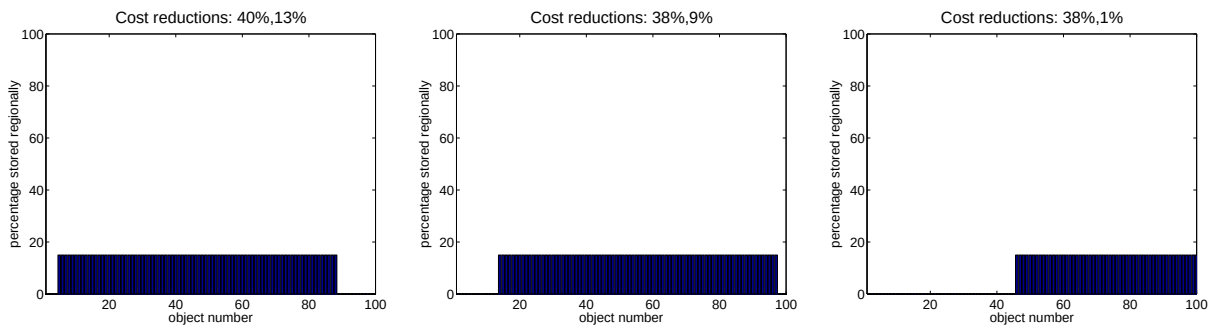
segments is about 13% of the total number of segments for the 100 objects in the system.

5.1. Minimizing Use of the Remote Server

In this section we solve the model for $\beta = 0$ to compute the partial and full objects that should be cached at the regional servers to minimize the use of remote server bandwidth. The optimal cache contents for particular values of client request arrival rate, number of regional servers, and server capacity and bandwidth, are shown in Figure 4. Above each graph, we give the percent reduction in required remote server bandwidth for two improvements suggested in this paper: (1) the percent reduction when the regional server optimally caches only whole objects ($\theta_i^{cr} = 0$) and each server uses transmission mini-clusters (of size $k = 7$) compared with optimally caching only whole objects and not using mini-clusters ($k = 0$), and (2) the additional cost savings when the regional servers are able to cache partial objects. The key insights from these results are:

- There is a high tendency to cache partial objects.
- The use of transmission mini-clusters greatly reduces delivery cost unless the client request rate is high enough that full transmission clusters for the leading segment set are always well utilized.
- The ability to cache partial objects can lead to a substantial (up to 52%) reduction in delivery cost.
- When server storage capacity is the active constraint, but there is sufficient storage to cache more than the initial k segments of each object, then the optimal solution also fully stores as many of the most popular objects as will fit (for $\beta = 0$).
- When the regional server bandwidth becomes the active constraint, less popular object segments are cached; however the precise form of the optimal cache content is specific to the given set of parameters that characterize the system, and thus the optimal content can only be determined by solving the model.

Increasing the number of regional servers reduces the regional client request rate, which can alleviate the regional server bandwidth constraint. For example, the server capacity is the active constraint for $\lambda' = 1000$, $P = 100$, and 2× baseline, and the optimal solution in this case is identical to the solution in Figure 4(b).



(a) $\beta = 0.3$ (cap) (b) $\beta = 0.5$ (cap) (c) $\beta = 0.8$

Figure 5. Impact of Relative Regional Server Bandwidth Cost
$k = 7$, $\lambda' = 10$, $P = 10$, $N_{channels} = 192$, $N_{segments} = 2436$.
(cap) indicates the capacity constraint is active.
The bandwidth constraint is never active.

5.2. Minimizing the Total Cost of Delivery

The experiments in this section investigate the regional cache content that minimizes total delivery cost over remote *and regional* channels, for a fixed regional server capacity and bandwidth. The results provide insight into how the form of the optimal assignment of full and partial objects changes as the relative cost of the regional channels, β, increases.

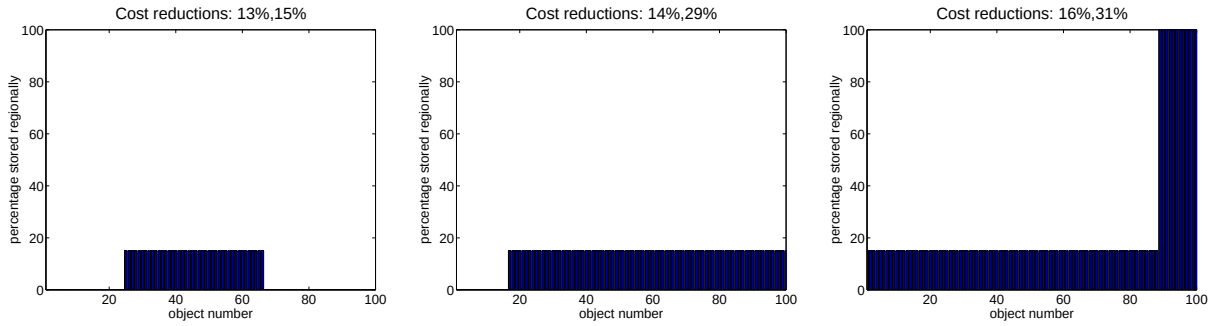
To facilitate comparison with the previous set of experiments for $\beta = 0$, we use the same system parameters as in Figure 4(a), except that we increase the value of β. The results, given in Figure 5, include the same cost reductions as in the previous set of experiments, and lead to the following observations:

- As β increases, lower-popularity objects are cached at the regional server. This is due to the earlier observation (from Figure 3(a)) that more popular objects have a greater increase in regional server bandwidth per unit of storage than the less popular objects.
- When β is sufficiently high, as in Figure 5(c), the minimum cost solution does not fully utilize either the available regional storage capacity or the available regional bandwidth. This is principally due to lower cost sharing of the regional multicasts.
- As in the experiments for $\beta = 0$, the use of transmission mini-clusters greatly reduces minimum delivery cost (for the given client arrival rate).
- The ability to cache partial objects yields a smaller reduction in minimum delivery cost when fewer segments are stored.

5.3. Impact of Regional Server Resource Constraints

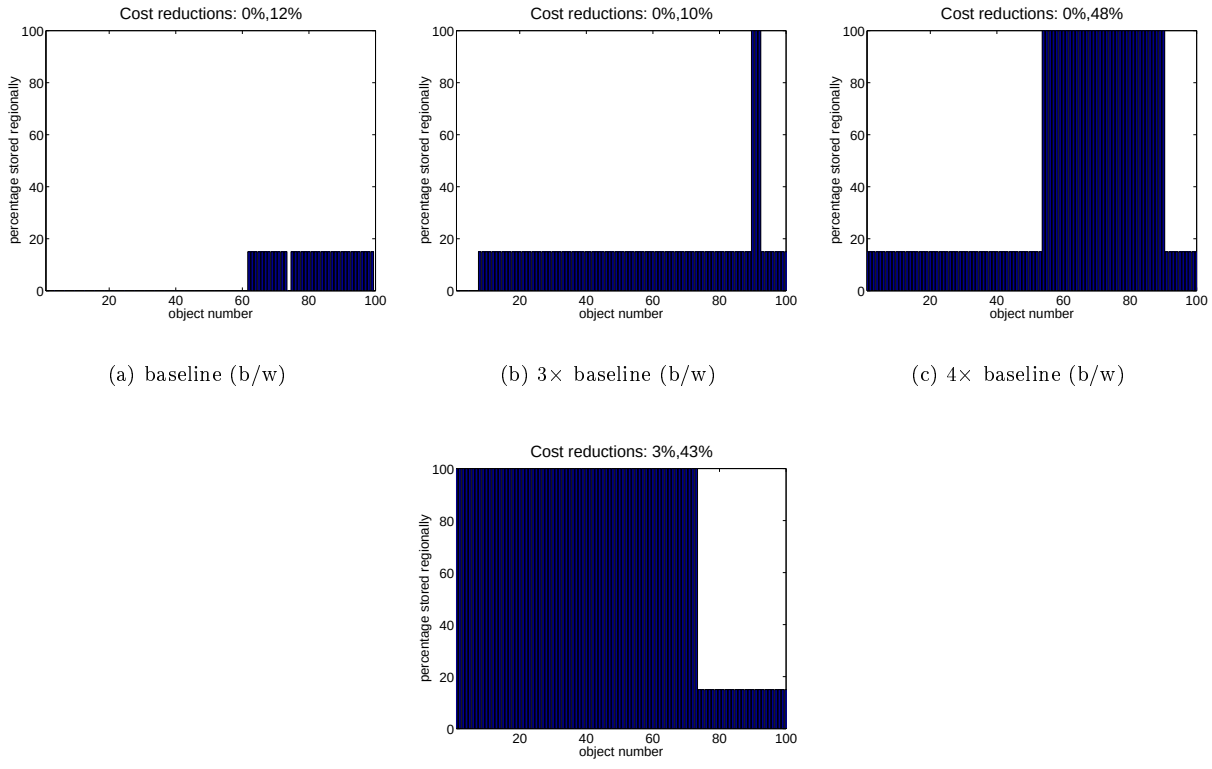
In this section we investigate the impact of varying the regional server storage capacity and bandwidth (in units of the baseline server capability) on delivery cost and the form of the optimal object assignments.

Figure 6 gives the results for the system parameters that were used in Figure 4(b), except that $\beta = 0.1$. For this set of system parameters, the cache content is constrained by the regional server storage capacity. As storage capacity increases, an increasing number of objects are partially cached. Once capacity is sufficient to cache more than the first seven segments (15%) of each object, the optimal solution is to also fully cache some of the objects. In this case, the *least popular* objects are fully cached (for $\beta = 0.1$), since the last five segments of these objects have similar bandwidth requirement for remote multicasts, but lower bandwidth requirement for regional multicasts, as compared with the last five segments of the most popular objects (see Figure 3(b)). As in the previous set of



(a) 1/2 baseline (cap) (b) baseline (cap) (c) 2× baseline (cap)

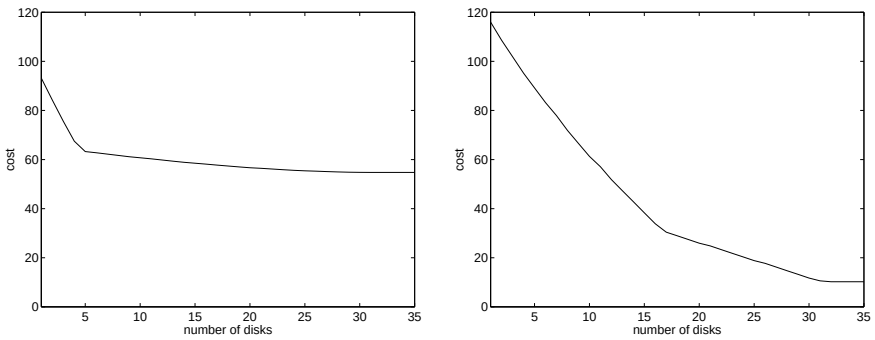
Figure 6. Impact of Regional Server Storage and Bandwidth Constraints
$k = 7$, $\lambda' = 100$, $P = 10$, $\beta = 0.1$



(a) baseline (b/w) (b) 3× baseline (b/w) (c) 4× baseline (b/w)

(d) 6 × baseline (cap)

Figure 7. Impact of Regional Server Storage and Bandwidth Constraints
$k = 7$ $\lambda' = 1000$, $P = 10$, $\beta = 0.01$



(a) λ', P, β, k from Figure 6 (b) λ', P, β, k from Figure 7

Figure 8. Delivery Cost vs Regional Server Storage and Bandwidth

experiments, $\beta > 0$ implies a tendency to store less popular object segments. The partitioned skyscraper architecture and the ability to cache partial objects each provide moderate reduction in delivery cost; altogether the reduction is substantial.

Figure 7 provides results for the case that the client request arrival rate is increased ten-fold and the relative cost of the regional channels, β, is decreased by a factor of ten. These are the system parameters that were used in Figure 4(c), except that $\beta = 0.01$. In this case, the optimal assignment is constrained by the regional server bandwidth at up to four times the baseline resource constraints, and for those configurations it is optimal to store lower-popularity objects. On the other hand, when the regional server resources are increased to six times the baseline, the capacity constraint becomes active. In this case, β is small enough that it is optimal to fully store the most popular objects completely at the regional server (in addition to storing the first seven segments of all other objects). Due to the high client request rate, the partitioned skyscraper architecture does not provide a cost savings in these cases. However, because the request rate is high, the ability to cache partial objects substantially reduces delivery cost when the regional server has reasonable capacity and bandwidth.

Figure 8 gives the minimum delivery cost ($C_{remote} + P\beta C_{regional}$) as a function of the regional server resource constraints, with four disks being equal to the baseline constraints, for the system configurations in Figures 6 and 7. Note that greater reduction in cost can be achieved by increasing the regional server resources when β is smaller, as would be expected.

6. CONCLUSIONS

This paper has developed a partitioned dynamic skyscraper delivery architecture that (1) provides more efficient use of system bandwidth by increasing the size of the window in which clients can join in on-going multicasts, and (2) allows regional servers to cache partial objects. A similar partitioned architecture that uses decoupled transmission *mini*-clusters might be applied to other segmented multicast delivery techniques.

We have also developed a simple optimization model that can be solved to determine the form of the optimal regional caching strategy in a system with homogeneous regional caches. The model computes the cache content that minimizes delivery cost, constrained by the storage capacity and bandwidth available at the regional servers.

Initial results of the model show the impact of various system parameters on the form of the optimal allocation policies. Key parameters include the regional client request arrival rate, the relative constraints on disk bandwidth and storage capacity at the regional servers, and the cost of the regional server bandwidth relative to the cost of the remote server and network bandwidth. The results show a very strong tendency to store the initial segments of many objects, rather than the entire data for fewer objects, at the regional server. The results also show that the partitioned skyscraper architecture and the ability to cache partial objects can greatly reduce delivery cost.

The optimization model developed in this paper assumes, for simplicity, that each object is of equal size and is either completely stored at the remote server, completely stored at the regional server, or has exactly a fixed value

k, $0 < k < K$, segments stored at the regional server. The model further assumes that all regional servers are *homogeneous*; that is, they each have the same overall client request arrival rate and object selection frequencies. These assumptions are desirable for initial exploration of the system design space. On-going research includes generalizing the model to investigate optimal assignments for heterogeneous regional servers, heterogeneous object lengths, and for allowing different numbers of segments to be stored regionally for different objects.

Acknowledgements:
The authors wish to thank Dan Sorin, John Zahorjan, and the anonymous referees for helpful comments on this work and its presentation.

REFERENCES

1. C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A Permutation Based Pyramid Broadcasting Scheme for Video-on-Demand Systems", *Proceeding of the IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, June 1996.
2. C. C. Bisdikian and B. V. Patel, "Cost-Based Program Allocation for Distributed Multimedia-on-Demand Systems", *IEEE MultiMedia 3*, 3 (Fall 1996), pp. 62-72.
3. A. Brooke, D. Kendrick, and A. Meeraus. *GAMS: A User's Guide*. The Scientific Press, South San Francisco, CA, 1988.
4. D. W. Brubeck and L. W. Rowe, "Hierarchical Storage Management in a Distributed VOD System", *IEEE MultiMedia 3*, 3 (Fall 1996), pp. 37-47.
5. P. Cao and S. Irani, "Cost-Aware WWW Proxy Caching Algorithms", *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, Monterey, CA, December 1997, pp. 193-206.
6. A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A Hierarchical Internet Object Cache", *Proceedings of the 1996 USENIX Technical Conference*, January 1996.
7. A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-demand Video Server with Batching", *Proceedings of the ACM Multimedia Conference*, San Francisco, CA, October 1994, pp. 15-23.
8. G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, New Jersey, 1963.
9. Dash Associates, Blisworth House, Blisworth, Northants, UK. *XPRESS-MP User Guide*, 1997.
10. D. L. Eager and M. K. Vernon, "Dynamic Skyscraper Broadcasts for Video-on-Demand", *Proceedings of the 4th International Workshop on Multimedia Information Systems (MIS '98)*, Istanbul, Turkey, September 1998, pp. 18-32.
11. M. C. Ferris, MATLAB and GAMS: Interfacing optimization and visualization software, Technical report, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, 1998.
12. L. Gao, J. Kurose, and D. Towsley, "Efficient Schemes for Broadcasting Popular Videos", *Proceedings of the 8th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 98)*, Cambridge, UK, July 1998.
13. K. A. Hua and S. Sheu, "Skyscraper Broadcasting: A New Broadcasting Scheme for Metropolitan Video-on-Demand Systems", *Proceedings of the ACM SIGCOMM'97 Conference*, Cannes, France, September 1997, pp. 89-100.
14. E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance*, Prentice Hall, Englewood Cliffs, New Jersey, 1984.
15. G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York, NY, 1988.
16. J.-P. Nussbaumer, B. V. Patel, F. Schaffa, and J. P. G. Sterbenz, "Networking Requirements for Interactive Video on Demand", *IEEE Journal on Selected Areas in Communications 13*, 5 (June 1995), pp. 779-787.
17. S. Sen, J. Rexford, and D. Towsley, "Proxy Prefix Caching for Multimedia Streams", Technical Report 98-27, Department of Computer Science, University of Massachusetts, July 1998.
18. A. K. Tsiolis and M. K. Vernon, "Group-Guaranteed Channel Capacity in Multimedia Storage Servers", *Proceedings of the 1997 ACM SIGMETRICS Conference on Measurement and Modelling of Computer Systems*, Seattle, WA, June 1997, pp. 285-297.
19. S. Viswanathan and T. Imielinski, "Pyramid Broadcasting for Video-on-Demand Service", *Proceedings of the SPIE Multimedia Computing and Networking Conference*, Vol. 2417, San Jose, CA, February 1995, pp. 66-77.
20. S. Viswanathan and T. Imielinski, "Metropolitan Area Video-on-Demand Service using Pyramid Broadcasting", *Multimedia Systems 4*, 4 (August 1996), pp. 197-208.
21. S. Williams, M. Abrams, C. R. Standridge, G. Abdulla, and E. A. Fox, "Removal Policies in Network Caches for World-Wide Web Documents", *Proceedings of the ACM SIGCOMM'96 Conference*, Stanford, CA, August 1996, pp. 293-305.