

A Scalable Video-on-Demand Server for a Dynamic Heterogeneous Environment^{*}

Maria Papadopouli^{**} and Leana Golubchik^{***}

Abstract. In this paper, we consider a Video-on-Demand storage server which operates in a *heterogeneous* environment and faces (a) fluctuations in workload, (b) network congestion, and (c) failure of server and/or network components. We present scheduling of data retrieval techniques which are capable of dynamically resolving congestion which occurs on the storage subsystem due to expected and unexpected changes in I/O bandwidth demand for a *scalable* video server in a *multi-disk* environment.

1 Introduction

Recent technological advances in digital signal processing, data compression techniques, and high speed communication networks have made Video-on-Demand (VOD) servers feasible. The design of such large scale systems involves several challenging tasks, including the satisfaction of (a) real-time constraints of continuous delivery of video objects as well as (b) quality of service (QoS) requirements. A great deal of work has been performed in the last few years in the area of VOD server design (see [7] for details).

In this work, we consider the retrieval of variable bit rate (VBR) video streams from a VOD storage-server in a *multi-disk environment*. An attractive feature of the use of VBR video (as opposed to constant bit rate (CBR) video) is the constant quality of video that can be provided throughout the duration of an object's display. However, VBR video exhibits significant variability in required video display rates. This variability can affect resource utilization in the system and complicate scheduling of both transmission of objects over a communication network as well as their retrieval from the storage subsystem. There are several works dealing with the support of VBR video for networking applications [6] some of which can be extended for use in the storage-server domain; however, an essential difference in this case, is that a storage server has *a priori* knowledge of the video traces which can aid in *further* improvements of the design.

In many multimedia systems different levels of quality of service (QoS) can be supported through the use of the multiresolution property of video (and/or images). In this work, we present a *scalable* VOD server for a *heterogeneous*

^{*} This work was supported in part by the NSF CAREER grant CCR-96-25013.

^{**} Department of Computer Science, Columbia University.

^{***} Department of Computer Science, University of Maryland at College Park. This work was partly done while the author was with the Department of Computer Science at Columbia University.

environment that provides statistical service guarantees and propose scheduling techniques for retrieval of VBR video that exploit the multiresolution property of compressed video streams. An example of such a system includes a video server delivering videos over the Internet, with users (potentially) requesting service using different types of hosts. In this work, we define the *scalability* characteristic, as it applies to a VOD system, as the capability to adjust to changes in (a) workload and (b) availability of storage and network resources as well as the capability to deliver video data at different levels of QoS guarantees.

In general, the video server faces (a) fluctuations in workload, (b) network congestion, and (c) failure of server and/or network components. At the same time, the users, while being served, are subject to changes in their sustained bandwidth requirements due to either (a) network congestion, (b) failure of network components, or (c) moving to a different environment (i.e., network). We define the *sustained bandwidth* of a user, in a certain time period, as the rate at which he/she is “expected” to effectively receive the data in that time period and which corresponds to a certain video quality profile ¹.

The flexibility that scalable compression techniques provide, in adjusting the resolution (or rate) of a video stream at any point **after** the compressed video object has been generated can be of great use in designing scalable video servers. Various video compression schemes, such as subband coding and MPEG-2, provide such a multiresolution property. The multiresolution property has been utilized, for instance, in previous work on dynamic adjustment of resolution of video streams being transmitted through a *communication network* [5] based on available network resources; although this approach has produced good results in utilization of available communication network bandwidth, there are several difficulties with adapting it to solving similar problems with utilization of disk bandwidth resources in a *storage* subsystem. In [16] a framework is proposed for a layered substream abstraction with highly scalable compression algorithms to support this abstraction. Finally, simulations in [3] show that the use of scalable video with Laplacian and Pyramidal coding can greatly increase the I/O bandwidth demand that can be sustained and decrease the waiting time for start of new requests for video objects, as compared to the use of full-resolution non-scalable video.

The focus of this paper is on the scheduling of data retrieval and approaches to resolving disk bandwidth congestion under expected and unexpected changes in I/O bandwidth demand. The work presented here differs from other related works in that the scheduling of the retrieval *adapts* to the availability of the *storage* and *network* resources as well as the *user’s* bandwidth requirements (that may change dynamically). It is introduced in a dynamic rate-distortion context for a *multi-disk environment*.

Specifically, we consider techniques which dynamically adjust resolution of video streams in progress, in order to adjust to fluctuations in workload and

¹ The changes in the sustained bandwidth of a user might be due, for instance to network congestion. Some of the work on determining the sustained bandwidth is discussed in [14].

resource availability while satisfying given QoS constraints and utilizing system resources efficiently. We propose resolution adjustment and load balancing techniques which address: (a) different causes of fluctuations, which include VBR property of video objects, the use of VCR functionality, as well as failure of system components, (b) the extent and duration of “overflow” of disk bandwidth demand beyond the available resources, (c) predictability of future I/O bandwidth demand, and (d) variations in QoS requirements. The resolution adjustment algorithm deals with short term fluctuations in the workload and takes advantage of the multiresolution property of video by adjusting the resolution of video streams in progress on a per single disk basis. The load balancing algorithm, on the other hand, deals with long term fluctuations in workload as well as extensive overflow of I/O bandwidth demand and “shifts” the load between multiple disks in the system; this algorithm takes advantage of both the multiresolution property of video and replication techniques. Furthermore, it also deals with changes in users’ sustained bandwidth requirements ².

2 System Description and Background

In this section we first briefly review the notion of scheduling of video requests in cycles or groups and then discuss data layout issues. Throughout the paper we use the term *stream* to refer to the delivery of a given video object at a given time.

<i>System Parameters</i>	
Notation	Description
L_d	set streams whose data blocks are stored on disk d
B	effective bandwidth of a single disk
τ_w	size of an interval (in units of time)
T_{cycle}	size of a cycle (in units of time)
$X_{i,d}(c)$	r.v. bandwidth requirement (for full resolution) of stream S_i on disk d during cycle c .
$X_{i,d}^k(c)$	r.v. bandwidth requirement of the first k layers of stream S_i on disk d during interval c .
$fb_{i,d}(w)$	sustained bandwidth of stream S_i on disk d during interval w .
μ	mean bandwidth requirement (for full resolution) of a stream
μ^L	mean bandwidth requirement that corresponds to retrieval of L multiresolution layers

Table 1. Notation

2.1 Data Retrieval and Corresponding Disk Model

To achieve efficient use of available disk bandwidth, it is common to organize the scheduling of streams into (time) cycles or groups, e.g., as in [2]. In their simplest form, cycle-based schemes deliver in each cycle the data that is read in the previous cycle. During each time period, data for each active stream is read from the disks into main memory while, concurrently, the data read

² We would like to emphasize that the success of resolution adjustment techniques, in general, depends on the number of resolutions available, the workload on the system, and the variability of the VBR video streams. For instance, in the case of subband coding, a large number of layers can be supported, whereas in MPEG-2, the number of layers is restricted to a maximum of 3.

during the previous cycle is transmitted over the network to display stations. The motivation for this organization is to provide opportunities for seek optimization (see [2] for details)³.

In the remainder of the paper, we will consider scheduling of data retrieval and overflow management in time intervals composed of an integral number of cycles. This slight generalization will allow us to control the scale on which bandwidth re-allocation is performed (as explained in more detail in Section 2.3). Thus, an interval w starting in cycle c , i.e., $[c, \tau_w + c]$, whose size is τ_w (in *number of cycles*) is composed of consecutive, non-overlapping cycles ($c, c + 1, \dots, c + \tau_w$), where continuous playback of a video with quality Q_L can be guaranteed if all blocks corresponding to $\{X_{i,d}^L(j), \text{ where } j \text{ is a cycle and } j \in [c, c + \tau_w]\}$ have been retrieved (from disk d) by the end of that interval, where $X_{i,d}^L(j)$ is a random variable that indicates the bandwidth requirement of stream S_i on disk d , during cycle j corresponding to video quality Q_L . Lastly, we define the mean bandwidth requirement of stream S_i on disk d during interval w corresponding to video quality Q_L as follows:

$$\mu_i^L(w) = X_{i,d}^L(w) = \frac{\sum_{j=c}^{c+\tau_w} X_{i,d}^L(j)}{\tau_w} \quad (1)$$

Disk Model

We now introduce a simple disk model that we use in the remainder of this paper. During each cycle of size T_{cycle} each stream retrieves a variable number of blocks corresponding to its bandwidth requirement for that cycle. Given the cycle based scheduling scheme the amount of time needed to retrieve data blocks corresponding to x streams scheduled during that cycle includes (a) a maximum seek τ_{seek} , which is the time to move the disk head between the extreme inner and outer cylinders of a disk, (b) rotational latency τ_{rot}^{avg} , and (c) the transfer time for each of the x streams that is attributable to reading the necessary number of data blocks, τ_{rot} . Note that the amount of data that has to be retrieved per stream in order to maintain continuity in the data delivery is $T_{cycle} * \mu$, where μ is the *mean* bandwidth requirement of each stream. Then, the time to read this data is $\frac{T_{cycle} * \mu}{B_{track}} \tau_{rot} + \tau_{rot}^{avg}$, where B_{track} is the number of bytes per track, and the constraint that there must be time in a cycle to read that much data for x streams is $\tau_{seek} + x(\frac{T_{cycle} * \mu}{B_{track}} \tau_{rot} + \tau_{rot}^{avg}) \leq T_{cycle}$. In the remainder of the paper we assume that $\tau_{rot}^{avg} = \frac{\tau_{rot}}{2}$, which then gives us a bound on the number of streams that can be serviced in one cycle on one disk⁴: $x \leq \frac{T_{cycle} - \tau_{seek}}{0.5 + \frac{\tau_{rot}}{B_{track}}} \frac{1}{\tau_{rot}}$.

The effective bandwidth of a disk is then $B = \mu * x$.

³ Note that, our bandwidth reallocation techniques, presented later in the paper, are not restricted to cycle-based scheduling, but are presented in that context for simplicity of exposition.

⁴ We have assumed that the data blocks of the same video object that correspond to one retrieval unit are stored contiguously on the disk. Therefore, the rotational latency corresponding to retrieval of the last fraction of the data block, i.e., the one that does not (necessarily) make up an entire track is on the average $\frac{\tau_{rot}}{2}$. We have also assumed that zero latency reads [13] are possible for the full size tracks and that B_{track} is a constant (i.e., we do not consider zones here).

2.2 Data Layout and Partial Replication

In this paper we extend the notion of a traditional replication scheme [1] for use in a multiresolution video server environment to instead support backup copies of videos in a *lower* resolution than the primary copy. That is, a backup copy of a video object is not necessarily identical to its primary copy; however, all resolution layers which are replicated, are identical to their primary copy counterparts and correspond to a predetermined video quality level Q_L ; thus, the backup copy is composed of the first L layers of each video segment ($L = 1, \dots, max$). In the remainder of the paper, we refer to this form of replication as *partial* replication.

In general, replication provides opportunities for better disk bandwidth utilization (as will become more apparent in Section 3), since it provides: (a) flexibility to service a stream by partial simultaneous retrieval of data from multiple disks in the system, when there is not sufficient bandwidth to serve that stream from any one disk, and (b) load-balancing opportunities, i.e., opportunities for “shifting” some of the load from one disk to another. Although much of the discussion on scheduling which follows is not restricted to a particular replication scheme, in Section 4 we present a scheme which is specific to *chained declustering* [9, 8]; thus, we describe this particular replication scheme next.

In the traditional chained declustering layout, at any point in time, two physical copies of a data fragment, termed the primary and the backup copy are maintained. If the primary copy of a fragment resides on disk D_i , then the backup copy of that fragment resides on disk $D_{i+1} \pmod{N_d}$, where N_d is the number of disks in the system). Under dynamic scheduling, a server can choose to retrieve a data block from either of the disks containing a replica of that data block. Thus, in our system, by (re)scheduling some data retrieval from disk D_i to disk D_{i+1} , the server can “create” available bandwidth space on disk D_i , which can be used for retrieval of other blocks corresponding to streams that were originally scheduled for service on either disk D_i or disk D_{i-1} (and consequently can be transferred for service to disk D_i). The provision of a backup copy in a *lower* resolution can also be applied to the chained declustering scheme. Figure 1 illustrates an example system using chained declustering and partial replication.

2.3 Interval-based Retrieval and Admission Control

Many different approaches to allocating (or reserving) bandwidth, at admission control time, for servicing video streams are possible. And, the scheduling framework described in Sections 3 and 4 are not limited to a specific approach; however, in order to focus the discussion better, we assume the following bandwidth reservation scheme (performed at admission control time) in the remainder of the paper. For each interval, the bandwidth reservation is performed by approximating the bandwidth requirements of the video (during that interval) with its *average* (over that interval), i.e., $\mu_i^L(w)$. The average bit rate reservation can

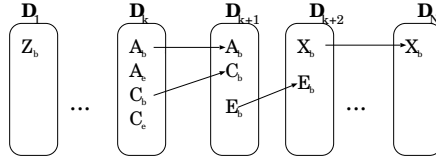


Fig. 1. Chained declustering with partial replication: the segments of the form S_b correspond to the first b resolution layers of a video object and have been replicated. For the S_e segments which compose the remaining or enhanced resolution layers only a single copy is kept. The arrows indicate a potential shift of the retrieval of data blocks from one disk to the next.

utilize resources more efficiently than the peak rate reservation, but leads to potential congestions (due to the variability of the stream bandwidth demand)⁵. One of the foci of this paper is a technique for alleviating these congestions without (significantly) affecting the quality of service provided to the users. In order to limit such congestion (and be able to provide an “acceptable” QoS level), it is necessary to perform some form of admission control.

During *admission control*, it is determined whether there are sufficient available resources for retrieval of a newly requested video stream; this is done by considering $\mu_i^L(w) \forall w$ and determining, on per interval basis, whether there is sufficient bandwidth available in the system for support of this new stream. The reservation of bandwidth resources made during the admission control phase, provides *statistical* guarantees for the retrieval of the streams and only “approximates” the actual retrieval schedule, without preventing potential congestion. Hence, a bandwidth re-allocation mechanism that dynamically alleviates congestions is required. The scheduling techniques presented here are not limited to a specific admission control scheme; thus, in the remainder of this paper, we only make the assumption that the admission control policy results in statistical QoS guarantees as follows — *a server can guarantee, to at most n_L users, a video quality of Q_L , with probability P_L* , where various methods for determining n_L and P_L can be constructed but are outside the scope of this paper⁶.

In this paper, for the sake of simplicity, we assume equal size intervals which are the same for all streams and are equal to an integral number of cycles. In general, we will consider intervals on the order of a few seconds. Of course, in order for continuous playback guarantees to be satisfied, we also need to assume that the user provides a buffer of size $2 * b_{max} * \tau_w * T_{cycle}$, at the user’s site, where T_{cycle} is the duration of one cycle, b_{max} is based on the user’s video quality requirement and is basically the maximum bandwidth requirement

⁵ The significance of underutilization of resources due to peak rate reservation in a piecewise (i.e., interval-based) manner, as opposed to average rate reservation with possibility of overflow, depends on the characteristics of the video trace and the size of the intervals.

⁶ The discussion on the choice of interval size and the guarantees of the reservation mechanism can be found in [12].

corresponding to the quality of service he/she requests⁷. This buffer space will “mask” the jitter that might otherwise result from altering the data retrieval to be on a per-interval, rather than per-cycle, basis.

3 Scheduling of Data Retrieval

It is the main task of the scheduler to determine the amount of data to be retrieved per stream per cycle and schedule the retrieval for the appropriate cycle and disk. Due to the guarantees made during the admission control phase, the scheduler is ensured that there is available bandwidth somewhere in the system to serve all admitted streams with video quality Q_L with probability P_L (see Section 2.3). However, the scheduler might be required to re-allocated (or shift) part of the workload, due to congestion, across the disks of the system. Recall that, due to replication, some of the data can be retrieved from one of two disks. Furthermore, transferring of some load from one disk to the next one might result in a number of shifts of load between consecutive disks. That is, a video block of stream S_i , of size b_i can be “shifted” from a congested disk D during some time interval T , to disk D’ if: (a) a replica of the video block is also stored on disk D’ **and** (b) the available bandwidth (that exists or could be “made available”) on disk D’ during time T *at least* corresponds to the amount needed to retrieve a block of size b_i . Below, we present a general framework for re-allocating bandwidth to streams in cases of fluctuations in order to dynamically exploit the available bandwidth of the system.

3.1 Resolution Adjustment on Multiple Disks

An optimal assignment of data block retrievals to disks, i.e., one that minimizes the amount of data that can *not* be retrieved (due to overflow), can be determined using a “max flow” algorithm [4]⁸.

Graph Generation

We begin with the needed generation of an appropriate graph. The construction of this graph and the computation of the corresponding optimum schedule under given constraints requires the knowledge of the amount of bandwidth that was *reserved* at admission control time for retrieval of data blocks corresponding to active streams as well as the amount of bandwidth that is *required* by these streams during the time interval in question. The max flow algorithm is specified on a per-interval basis, i.e., we can determine, on a per-interval basis, whether overflow will occur in that interval and run the max flow algorithm in order to determine a new assignment of streams to disks and the corresponding retrieval schedule which alleviates overflow in an optimal manner (given the optimality criteria stated above), where τ_w is a parameter of the scheduling algorithm. The

⁷ Here, we assume double buffering at the user’s site, for ease of exposition.

⁸ See [11] for extensions to use with lower bounds on edges.

max flow algorithm will be applied on a graph $G^L(w) = (V, E)$ (e.g., see Figure 2), which can be constructed as follows:

$$G^L(w) = (V, E), \quad V = \{D_1, D_2, \dots, D_{N_d}\} \cup \{V_1, \dots, V_n\} \cup \{s\},$$

$$E = \{(s, V_j), j = 1, \dots, n\} \cup \{(V_j, D_i), j \in L_{D_i}, i = 1, \dots, N_d\}$$

where L_{D_i} is the set of streams whose data blocks are stored on disk D_i and where each node D_i has capacity B , which is the effective bandwidth capacity of disk D_i (see Section 2.1). There are n number of nodes V_j , each corresponding to a currently active stream S_j , and each connected to the start node s (an artificial node) with an edge $(s, V_j) \in E$ whose capacity is bound by l_j and u_j , i.e., the flow on this edge, (s, V_j) , is $l_j \leq f_j \leq u_j$. If the required bandwidth of stream S_j during interval w , $fb_j(w)$, is less than $X_{j,d}^L(w)$, then $l_j = u_j = fb_j(w)$. Therefore, if the (adjusted) sustained bandwidth of a user is less than or equal to the mean bandwidth that the server has reserved for that user, then the retrieval corresponding to S_j will be at least of size $fb_j(w)$. If, on the other hand, the user's required bandwidth is higher than $X_{j,d}^L(w)$, then $l_j = \mu_j^L(w)$ (refer to Section 2.1) and $u_j = fb_j(w)$. Furthermore, if the first k layers of a stream S_i are stored on disk D_j , then there is an edge (V_i, D_j) , with its lower and upper capacity equal to 0 and $\mu_i^k(w)$, respectively. The max flow algorithm will assign a flow to each edge (s, V_i) which represents the bit rate that the server will deliver to S_i during interval w . Note that, a flow > 0 on edges (V_j, D_i) indicates that a fraction of stream S_j will be retrieved from disk D_i .

The max flow algorithm produces one bandwidth allocation (from all possible ones) that retrieves the maximum total amount of data for the active streams with respect to the guarantees of certain quality of service and constraints on the playback rates of the users ⁹.

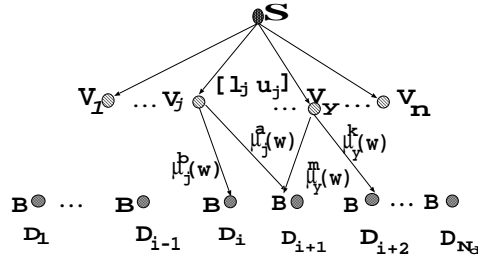


Fig. 2. Graph for the max flow algorithm

Lastly, the time complexity of the max flow algorithm is $O(|V||E|\log|V|)$, e.g., the algorithm by Sleator and Tarjan [15] can be used. Then, given that the

⁹ We should emphasize that the feasibility of a flow on the above graph will depend on the statistical guarantees the server makes at admission control time.

maximum total number of streams in the system is $n_f * N_d$, the time complexity becomes $O((N_d + N_d * n_f + 1) * (3 * n_f * N_d)(\log(N_d * n_f + N_d + 1)))$ or $O((n_f * N_d)^2(\log(\max(N_d, n_f))))$. The max flow algorithm can run at the beginning of an interval to compute the optimal bandwidth allocation for the next interval. Of course, the tradeoff here is between having sufficient time to run the max flow algorithm (i.e., making the intervals longer) and the amount of buffer space needed at the user site (which grows with the interval length).

3.2 Resolution Adjustment on Per-Disk Basis

In this section we consider the problem of “per-disk” re-adjustment of the data retrieval schedule. As already mentioned in Section 1, the motivation here is to be able to resolve short term overflow problems relatively quickly and on a per-disk basis.

One of the important requirements of bit rate re-adjustment is to minimize the distortion that the streams will experience as a result of the re-adjustment process. We define the distortion of a stream S_i during time interval t , due to a decrease of information retrieved from $b_i(t)$ to $b_i(t) - \beta_i$ as a non-decreasing (cost) function R_i : $R_i(\beta_i, t) = \tau_i * \Delta_i(\beta_i, t)$, where, $b_i(t)$ corresponds to a (feasible) retrieval schedule¹⁰ of the requested resolution, τ_i is a distortion tolerance factor¹¹, and $\Delta_i(\beta_i, t)$ is the distortion due to retrieving only $b_i(t) - \beta_i$ instead of $b_i(t)$ amount of data.

Each layer k is associated with a “distortion measure”, $\delta_i(b_i^k(t)) = r_k$, where $b_i^k(t)$ corresponds to the amount of data that needs to be retrieved for the k^{th} layer of stream i at time interval t ¹². Therefore, r_k indicates the resolution improvement, if in addition to the first $k - 1$ layers of video retrieval, we also retrieve the k -th layer. When β_i amount of data is not retrieved, the decoder will be able to decode only the first λ_i layers (instead of all λ_{max} layers that compose the requested resolution data blocks), and therefore, the distortion that the user will experience during interval t will be as follows:

$$\Delta_i(\beta_i, t) = \sum_{k=\lambda_i+1}^{\lambda_{max}} \delta_i(b_i^k(t)), \text{ where } \lambda_i = \max_k(\beta_i \leq \sum_{l=k}^{\lambda_{max}} b_i^l(t)). \quad (2)$$

Here, the cost function R_i quantifies the QoS of a stream S_i .

The server runs the Resolution Adjustment (RA) algorithm (given below) on a future time interval t after detecting a congestion in that interval in order to determine the proper bandwidth re-allocation. Let us assume that for that

¹⁰ A retrieval schedule is feasible when it does not violate the disk bandwidth constraints while it guarantees normal playback at the receiver without discontinuities.

¹¹ The distortion tolerance factor, τ_i , is a function of: (a) QoS requirements of the user, (b) feedback mechanism, and (c) subjective distortion measures on specific video segments, e.g., the service of streams in FF/REW mode, which can tolerate larger distortions.

¹² Distortion measures may vary from a simple mean squared error (MSE) estimate to more complicated perception-based functions.

interval t , B_{of} is the disk bandwidth overflow on the disk in question, b_i^{max} is the retrieval unit corresponding to requested resolution of the video, and b_i^{base} is the retrieval unit corresponding to some minimum acceptable resolution, as specified by the user¹³. Then, RA can be formulated as follows¹⁴:

- **Find** β_i $i = 1, \dots, n$
- **Minimize** $max_i\{R_i(\beta_i)\} - min_i\{R_i(\beta_i)\}$
- **such that** :
 1. $\sum_i \beta_i = B_{of}$
 2. $0 \leq \beta_i \leq b_i^{max} - b_i^{base}$, where β_i is a non-negative integer, $i = 1, 2, \dots, n$, where n is the number of active streams on the disk.

The RA problem is a “fair resource allocation” problem that can be solved using the FAIR algorithm, as given in [10], of complexity $O(n \log(max(n, B_{of})))$, where n is the number of active streams on a disk and B_{of} is the disk bandwidth overflow. If RA does not have a feasible solution, we can attempt to shift the overflow to the remaining disks in the system (as described above)¹⁵.

4 Application of Max Flow Formulation

In this section we consider sources of congestion due only to changes in sustained bandwidth of users and then evaluate the performance of the resulting server, termed $S_{scalable}$. This evaluation focuses on the retrieval scheduling/overflow management policy which was formulated in Section 3 as a max flow problem. For the purposes of comparison we use a baseline server, termed $S_{independent}$, which does not take advantage of replication (and here specifically chained declustering with partial replication). In $S_{independent}$ all disks are independent, i.e., a retrieval for a particular stream is always scheduled on a specific disk without the flexibility of replication, that exists in $S_{scalable}$, which allows “shifting” of the load across disks. The metric used in evaluating the overflow management policy is the percentage of (“newly available”) bandwidth that we are able to utilize (we explain this in more detail below). The results of this evaluation have been obtained through simulation, where we use the disk model given in Section 2.1. The parameters of that model used in this section are as follows: (a) $\tau_{seek} = 30$ msec, (b) $\tau_{rot} = 10$ ms, (c) $B_{track} = 100$ KB, and (d) $T_{cycle} = 530$ msec.

Below we illustrate that, through the use of chained declustering with partial replication, the server is not only capable of resolving overflow due to fluctuations, for instance, of VBR video compression, but is also sufficiently flexible and can take advantage of the available bandwidth that results from reductions

¹³ For instance, in the notation of the previous section, if the required bandwidth of stream S_i during interval t $fb_i(t)$, is higher than $X_{i,d}^L(t)$, then $b_i^{base} = \mu_i^L(t)$ and $b_i^{max} = fb_i(t)$; otherwise $b_i^{base} = b_i^{max} = fb_i(t)$.

¹⁴ In this formulation, we drop the “time dependence” in the notation in order to simplify it.

¹⁵ This approach can result in certain disadvantages, the description of which is left out due to lack of space; see [12] for details.

in the sustained bandwidth of some users. For the purposes of the following discussion, we term the users with decreases in sustained bandwidth requirements in an interval w as “degraded” users; similarly, we term the users with increases in sustained bandwidth requirements in an interval w as “upgraded” users. The simulations described below aim to evaluate the bandwidth re-allocation process of $S_{scalable}$ and $S_{independent}$ by computing the portion of bandwidth (that can be scheduled) that was freed by the “degraded” users that can (potentially) be re-assigned to the “upgraded” users in order to satisfy their requests for increases in bandwidth, in an interval w .

Let us consider some such interval w . In order to focus on the comparison between overflow management of the two servers, we fix the load on each disk of both server to be the same. Specifically, for our simulation we consider a cluster of 45 disks. The load on each disk, i.e., the total amount of bandwidth that is needed during interval w is determined based on a *uniform* distribution with values ranging in [38 Mbps, 60.8 Mbps].

As already mentioned, $S_{scalable}$ uses the max flow algorithm; Figure 3 illustrates the graph for this max flow algorithm that can be created as follows: (a) each disk corresponds to a node and each stream corresponds to an edge, (b) there is an artificial “start” node as well as a “sink” node, (c) for each “upgraded” stream, there is an edge which connects a node corresponding to a disk on which the stream is (at least partially) scheduled with an artificial “start” node, with capacity equal to the *increment* in the bandwidth requirement of that stream, (d) for each “degraded” stream, there is an edge which connects a node corresponding to a disk on which the stream is (at least partially) scheduled with an artificial “sink” node, with capacity equal to the *decrement* in the bandwidth requirement of that stream, (e) for each node that corresponds to a disk there is an edge that connects it to the node that corresponds to the disk on its right¹⁶ (for the last disk, its node is connected with the node of the first disk) — the capacity of this edge is equal to the amount of bandwidth that can be transferred during this interval w from the disk in question to the next disk on its right, which depends on the degree of replication used in the system as well as the bandwidth requirement of streams accessing the disks in that interval.

This graph is somewhat different from the one described in Section 3.1, which is due to the source of change in workload, which in this case is due *only* to the changes in sustained bandwidth of “degraded” and “upgraded” users. In addition, in this case we are considering a specific form of replication, namely that of chained declustering. Moreover, this simplification in the formulation of the max flow algorithm reduces the complexity of the solution to $O((n_f * (N_d)^2)(\log N_d))$. The max flow algorithm returns the maximum amount of bandwidth, as a fraction of the total amount of bandwidth that has been “released” by the “degraded” users, which can be re-allocated to the “upgraded” users. Note that, $S_{independent}$ is able to assign additional bandwidth to the “upgraded” users only if the disk on which they are served has some available bandwidth due (in this

¹⁶ Recall that in chained declustering, the disk logically to the right of disk i contains copies of data stored on disk i .

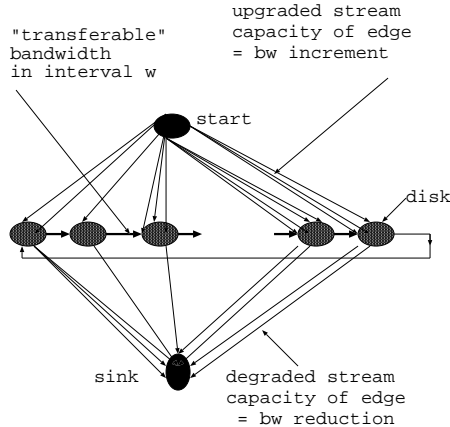


Fig. 3. Graph for the max flow algorithm.

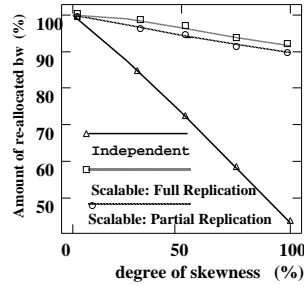


Fig. 4. Amount of bandwidth re-allocated to “upgraded” users.

case *only*) to the reduction in the bandwidth requirements of the “degraded” users. Both servers assign to the “degraded” users bandwidth equal to their new reduced bandwidth requirements.

We illustrate in Figure 4 the amount of bandwidth that was “released” by the “degraded” users, which can be re-allocated to the “upgraded” users, as a function of the “skewness” of the changes in users’ required bandwidth on the disks. Specifically, the total increment (reduction), $load_{increment}^d$ ($load_{reduction}^d$) in the bandwidth requirements on disk d is given by ¹⁷:

$$load_{increment}^d = \frac{C_{increment}}{\phi(d)\gamma}, \quad load_{reduction}^d = \frac{C_{reduction}}{f(d)\gamma} \quad (3)$$

where $C_{increment}$ and $C_{reduction}$ are constants, and $\phi()$ and $f()$ provide a “one-to-one” mapping of the (N_d) disks to the (N_d) different loads (i.e., in Eq. (3) d is in the range 1 to N_d), where N_d is the disk cluster size. As γ increases, the access pattern becomes increasingly skewed. In our simulations we consider the following values of γ : 0.0, 0.25, 0.5, 0.75, 1.0. Note that the *maximum* total increment (reduction) in bandwidth requirements on a disk is $C_{increment}$ ($C_{reduction}$). In this simulation we consider $C_{increment} = C_{reduction} = 35$ Mbps.

In Figure 4 we illustrate the performance of the two servers, $S_{independent}$ and $S_{scalable}$, with full (100%) and partial replication, where, for simplicity of illustration, we have assumed that the degree of partial replication corresponds to the amount of bandwidth reserved for a stream at admission control time.

For the case of partial replication we assume the degree of replication (i.e., percentage of the retrieval unit that is replicated) to be equal with the percentage

¹⁷ The increment/reduction in sustained bandwidth corresponds to the difference between the current sustained bandwidth requirements and the amount of bandwidth reserved at admission control time.

of bit rate that is reserved during the admission control per stream. As already mentioned, the performance metric is the percentage of “released” bandwidth (by the “degraded” users) which can be re-scheduled for use by “upgraded” users.

The performance of both servers degrades as the skewness increases. For example, in the case of $S_{scalable}$ and $\gamma = 0.25$ with 100% replication, 99% of the “released” bandwidth is re-allocated to “upgraded” users, whereas with partial replication 97% of the “released” bandwidth is re-allocated. However, in the case of $\gamma = 1.0$ with 100% replication, 91.6% of the “released” bandwidth is re-allocated, whereas in the case of partial replication 89.8% is re-allocated. Therefore, as expected, the less uniform the change in requested bandwidth, the more difficult it is to re-allocate the bandwidth.

Furthermore, the gap in performance between $S_{scalable}$ and $S_{independent}$ increases as the skewness increases, i.e., even under high skews in changes in bandwidth requirements, $S_{scalable}$ is able to re-allocate a large percentage of the “released” bandwidth by exploiting replication (or specifically chained declustering in this case) and “shifting” the increase in load (due to “upgraded” users) from one disk to another (i.e., one that has the newly available bandwidth due to “degraded” users). For instance, under full replication both servers have the same performance when $\gamma = 0.0$, whereas when $\gamma = 1.0$, the gap in performance between $S_{scalable}$ and $S_{independent}$, is more than 48.5%. Note also that the performance of $S_{independent}$ decreases almost linearly from 100% to 43%, as the skewness increases. This is due to the fact that as the “skew” increases less “good matches” will occur on each disk. By a “good (or perfect) match” we mean the case where the total amount of additional bandwidth that is requested (by the “upgraded” users) to be retrieved from a single disk is approximately the same (or exactly the same) as the total amount of the reduction in requested bandwidth (due to the “degraded” users) on the same disk. In the case of $S_{scalable}$ the higher is the degree of replication, the less the probability of having a “good match” affects the performance of the server. For example, under full replication, the performance of $S_{scalable}$ is 100% (for $\gamma = 0.0$) and is reduced to 91.6% (for $\gamma = 1.0$), whereas under partial replication the performance of $S_{scalable}$ is reduced from 100% (for $\gamma = 0.0$) to 89.8% (for $\gamma = 1.0$). Thus, based on the results depicted in Figure 4, we can conclude that skewness has a greater effect on the performance of $S_{independent}$ than on the performance of $S_{scalable}$.

In summary, due to proper utilization of replicated data, $S_{scalable}$ is more effective at taking advantage of the bandwidth that has been “freed” in an interval w due to “degraded” users and assigning it to the “upgraded” users, and therefore it adapts more effectively to changes in bandwidth requirements or availability of resources. This becomes more critical under higher degrees of skewness, that is, there is a larger improvement in the percentage bandwidth that can be “re-allocated” by $S_{scalable}$ as compared to $S_{independent}$. Finally, the higher is the degree of replication the less skewness affects the performance of $S_{scalable}$. This reduced sensitivity to skewness, of course, is achieved at the cost of additional storage space.

5 Conclusions

In summary, in this paper we consider the problem of delivery of VBR video streams in a VOD server under provisions of statistical quality of service guarantees. We present several techniques for re-scheduling the video streams and adjusting bandwidth allocations for streams in progress when fluctuations in workload as well as changes in availability of resources occur while satisfying QoS constraints and utilizing system resources efficiently.

References

1. D. Bitton and J. Gray. Disk Shadowing. *VLDB*, pages 331–338, 1988.
2. M. Chen, D. Kandlur, and P. Yu. Optimization of the Grouped Sweeping Scheduling (GSS) with Heterogeneous Multimedia Streams. *ACM Multimedia '93*, pages 235–242, 1993.
3. T.C. Chiueh and R. Katz. Multi-resolution video representation for parallel disk arrays. In *Proceedings of ACM Multimedia*, pages 401–409, 1993.
4. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. The MIT Press, 1989.
5. A. Eleftheriadis. *Dynamic Rate Shaping of Compressed Digital Video*. Ph.D. dissertation, Columbia University, 1995. Technical Report CU/CTR/TR 419-95-25.
6. A. Elwalid, D. Heyman, T.V. Lakshman, and D. Mitra. Fundamental Bounds and Approximations for ATM Multiplexers with Applications to Video Teleconferencing. *IEEE Journal on Selected Areas in Communications*, 13(6), August 1995.
7. J. Gemmell, H.M. Vin, D.D. Kandlur, and P.V. Rangan. Multimedia storage servers: A tutorial. In *IEEE Computer*, pages 40–49, May 1995.
8. L. Golubchik, J.C.S. Lui, and R.R. Muntz. Chained Declustering: Load Balancing and Robustness to Skew and Failure. *RIDE-TQP Workshop*, February 1992.
9. H. Hsiao and D. J. DeWitt. Chained Declustering: A New Availability Strategy for Multiprocessor Database Machines. In *Proceedings of Data Engineering*, pages 456–465, 1990.
10. T. Ibaraki and N. Katoh. *Resource Allocation Problems*. The MIT Press, 1988.
11. C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., 1982.
12. M. Papadopouli and L. Golubchik. A scalable video-on-demand server for a dynamic heterogeneous environment. Technical Report CUUCS-013-98, CS dept., Columbia University, 1998.
13. C. Ruemmler and J. Wilkes. An Introduction to Disk Drive Modeling. *IEEE Computer Magazine*, pages 17–28, March 1994.
14. S. Seshan, M. Stemm, and R.H. Katz. Spand: Shared passive network performance discovery. *Proc 1st Usenix Symposium on Internet Technologies and Systems*, December 1997.
15. R.E. Tarjan. *Data Structures and Network Algorithms*. CBMS-NSF Regional Conference Series in Applied Mathematics, 1983.
16. D. Taubman and A. Zakhor. A common framework for rate and distortion based scaling of highly scalable compressed video. *IEEE Transaction on Circuits and Systems for Video Technology*, pages 329–354, 1996.