Service Aggregation Through Rate Adaptation Using a Single Storage Format[1]

R. Krishnan and T.D.C. Little
Multimedia Communications Laboratory
Department of Electrical and Computer Engineering
Boston University, Boston, Massachusetts 02215, USA
(617) 353-9877, (617) 353-6440 fax
{*krash,tdcl*}*@bu.edu*

MCL Technical Report No. 05-10-1997

Abstract–Service aggregation schemes like batching, caching and rate adaptation can enable large scale interactive multimedia services, such as movies-on-demand and popular television programming. These applications are characterized by a large user population and a high density of access. Rate Adaptation is a dynamic service aggregation technique that can merge streams and reclaim resources at run-time. Existing schemes for rate adaptation are expensive since they require multiple content storage formats or special hardware, therefore, this technique has not been exploited so far. We propose a novel data placement and disk scheduling strategy which supports two different content progression rates originating from a single storage format, and that is applicable to single-disk architectures as well as to multi-disk architectures with large or small stripe-sets. The proposal leads to a feasible solution for dynamic service aggregation via rate adaptation, thereby making large scale multimedia services more economical.

Keywords: Service aggregation, rate adaptation, content progression rate, interactive video-on-demand, video servers, batching, caching, disk striping, service scaling.

[1]In *Proc. 7th Intl. Workshop on Network and Operating System Support for Digital Audio and Video*, St. Louis, MO, May 1997. This work is supported in part by the National Science Foundation under Grant No. NCR-9523958, EMC Corporation, and Hewlett-Packard.

1 Introduction

Interactive multimedia services such as movies-on-demand and interactive television have the characteristics of long playout times, low to moderate interactivity and high density of access. By high density of access, we mean that a small set of programs are accessed much more frequently than others within a close temporal locality. During periods of peak demand, several users would be accessing the same programs.

Let us consider a true video-on-demand (T-VOD) application, in which a user can choose a movie and have VCR-like control over the playout. Typical settings include polychannel architectures as described by Gifford [8] or multicast architectures as proposed by Almeroth and Ammar [1] or CATV [10]. Movies are multicast (or simulcast) on high bandwidth channels and a duplex "back channel" can be used to convey user interaction requests to the server. With today's technology, the cost of dedicating one channel per user is prohibitively high. For example, a server with an ATM connection at 622 Mb/s can support only about 415 streams at MPEG-1 bandwidths. Digital satellite systems provide about 200 channels while Hybrid Fiber/Coax (HFC) networking can support only up to 464 digital video channels. A large number of streams also implies enormous resource requirement at the server.

At the other end of the spectrum, there is a significant demand for services providing interactive video delivery over the Internet. As network bandwidths continue to be a bottleneck and the number of users are steadily increasing, service aggregation is inevitable.

The economic infeasibility of providing such services at present motivates research into resource sharing and service scaling techniques that can reduce per-user cost of usage. We perceive that service aggregation techniques would enable the provision of low-cost interactive services in the future. An approach that reduces the number of streams for a given user demand is preferable since it automatically reduces resource requirements in the server. The high access density characteristics can be exploited to combine multiple users into groups served by single streams.

Rate adaptation is a technique that tries to merge streams by varying their content progression rates. Merging reduces resource requirements both in the server and the network. Thus rate adaptive merging is an end-to-end aggregation solution. Other techniques like batching and caching mainly address server issues only.

Existing rate adaptation schemes are expensive since they entail enhanced storage or hardware requirements. The use of on-line rate adaptation hardware to alter content pro-

gression rates is prohibitively expensive. For n channels in our system, we require $\mathcal{O}(n)$ instances of such hardware. Another solution is to store content in different formats at the expense of extra storage. It has been suggested that merging can be limited to a given length of the movie to reduce storage requirements [9]. Consider a large storage server storing about 1000 movies in MPEG-1 format. This would require a terabyte of storage. If we limit merging of streams to only the first half, this would still translate to an additional requirement of 500 gigabytes to support rate adaptation.

In this paper, we propose a scheme that enables us to support rate adaptation using a single content format. Applications to striped and non-striped architectures are considered. The work is novel in that QoS scaling and disk-striping are used in an entirely new way. It is also significant since it provides a practical solution for rate adaptive merging – an aggregation scheme which is currently not popular due to the high cost of existing implementations. We expect that this technique would enable the provision of large scale interactive services. The rest of the paper is organized as follows. Section 2 provides the background and establishes the basis for our work by surveying existing aggregation techniques of batching, caching and rate adaptation and their limitations. Section 3 explains the proposed scheme. A discussion on the merits and demerits of the proposed scheme is provided in Section 4 followed by a concluding summary in Section 5.

2 Service Aggregation Techniques – a Survey

Let us return to the context of our VOD setting, in which a large number of users in a neighborhood share the same pool of channels. Suppose there are two users watching the same movie stream displaced by a small offset in time. If we can bridge the temporal gap between them, we can serve both users with a single channel and free the resources associated with one channel. This is the basic idea of service aggregation. Using this concept, we can reduce the number of channels required to serve a given number of users. There is a corresponding reduction in the number of streams to be retrieved, which translates to reduced bandwidth and buffering requirements in the server. It is intuitively evident that aggregation schemes can enable us to support a larger number of users for a given amount of resources, subsidizing the cost per user. That this is indeed the case has been demonstrated by Almeroth and Ammar [1].

Though streams may be merged through aggregation, user interactions like fast-forward will result in users breaking away from their groups. Thus promotion [14] of users is a

complementary operation to aggregation. As promotion increases resource requirements, the aggregation mechanism must dynamically try to compensate for such increases by reclaiming resources. Thus service aggregation schemes can be static or dynamic. Static aggregation schemes cluster requests and allocate resources to groups of users conservatively. Dynamic aggregation schemes are more attractive because they can reclaim resources at run-time, thereby supporting a larger user population and increased levels of interactivity. In the polychannel architecture setting, the duplex "back-channel" can be used by the server to communicate channel switching commands to users when they are aggregated with a group or when they are promoted and given a separate channel when they interact [14].

Dynamic service aggregation enables better system utilization and lowered costs through the provision of statistical rather than deterministic guarantees. There is a finite but small probability of a user interaction request being blocked, whereas, this would not be the case with dedicated channels. The goals for dynamic service aggregation schemes from a user perspective are minimized interaction latencies, low blocking probabilities, fairness and a low cost of usage. Initial aggregation schemes must minimize the reneging probability and the average waiting time [6]. It is conceivable that pricing policies would drive the operating point for these parameters. Service Aggregation techniques promote "sharing" of resources which is more desirable than "rationing," as the latter can result in significant degradation in quality of service to the user.

There are three main flavors of service aggregation techniques that are considered in the literature [6, 7, 9] viz., batching, caching and adaptive piggy-backing of streams. These schemes can be employed in isolation or in combination. As a background to our work, we survey the advantages and disadvantages of these techniques, and try to establish why rate adaptation is attractive and how existing schemes undermine its applicability.

2.1 Batching

The batching technique accumulates requests for playout until a threshold of demand is reached. This threshold can be a number of requests or a timeout or both. Batching imposes no additional hardware, storage or memory requirements, but can result in large waiting times and possible blocking on interactions.

2.1.1 Initial Batching

In this scheme, playout requests for new movies are clustered initially and channels are allocated to groups of users. This results in possible reneging of users and also on denial of requests on failure to meet certain popularity threshold criteria. Three different channel allocation policies – first-come-first-served, maximum-queue-length-first, and first-come-first-served-N (where a certain fraction of the server capacity is pre-allocated for batching the N hottest movies) are analyzed by Dan et al. [6]. This approach favors the first-come-first served policy on account of its fairness, while noting that the maximum-queue-length-first policy maximizes the number of users.

Initial batching is excellent as long as the users do not interact. However, VCR-like interactions cause users to break away from their groups, defeating the initial gains. If a group corresponding to the requested playout point exists then we can aggregate the user with this group. Since most of the interactions are likely to be of fractional duration of the batching interval, such a group is unlikely to exist. Such break-aways have to be handled by starting a new session (promoting) for this user by drawing from a pool of "contingency channels." A model for optimally allocating channels for batching, on-demand playback and contingency has been developed by Dan et al. [5].

When all contingency channels are consumed, VCR actions block. This is an event which can occur with a very small probability but nonetheless with a non-zero probability. This cannot be altogether avoided due to the statistical nature of the allocation policy. Periods of high interactivity may deplete all the free channels available and no more users can be admitted nor user interaction be permitted until some streams exit. This is a serious disadvantage of initial batching.

An enhancement that somewhat alleviates this problem is proposed next. The number of promotions can also be reduced by using a caching strategy in conjunction with batching (Section 2.2).

2.1.2 Intermediate Batching

To preserve the gains made out of batching, we propose that the server batch the streams not only at entry but iteratively at suitable points during the playout. This releases more channels thereby lowering the resource requirements on the average. This idea can be extended to handle VCR action blocking as discussed later. During intermediate batching, the

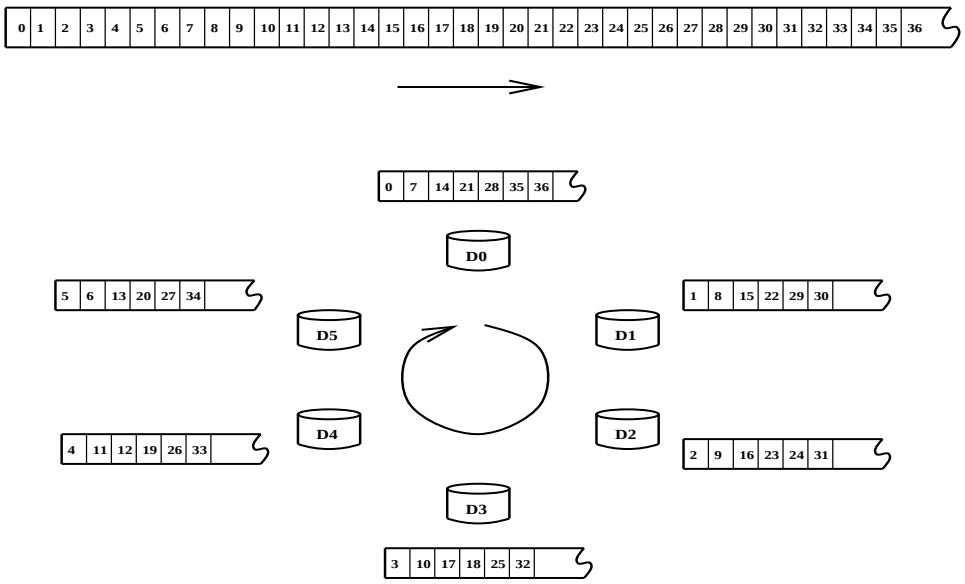

Figure 1: Placement Scheme for Multi-disk Architecture

batched users are fed with inserted content, possibly commercials. Initial batching schemes assume continuation of playout on blocking. Content insertion is a technique to extend the intermediate scheme to handle VCR action blocks.

2.2 Caching

The use of buffering in the client or server can reduce server disk bandwidth requirements or the number of channels for interactive video delivery. The usual rationale given for such schemes is the falling cost of memory, but as of now such solutions are expensive. Alternative schemes like the rate adaptation scheme proposed later in this paper can offer low-cost solutions. It is also possible to reduce buffer costs by using caching in combination with rate adaptation schemes.

2.2.1 Server Caching

Server caching techniques involve buffering the retrieved data in the server and feed subsequent streams from the buffer. Sincoskie [11] introduces the concept of a "start-stop buffer" to handle user interaction and "phase differences" between users. Interval caching is a buffering-based scheme proposed by Dan et al. [7] for handling VCR actions. The idea is to

cache a batching interval's worth of data to feed succeeding streams. Assuming that there are streams in every batching interval, the breakaway streams fall into one of these intervals. These breakaway streams can be served using the data in the cache without consuming storage bandwidth. The look-ahead scheduling scheme to support pause-resume interactions relies on server caching [16].

Buffering allows a larger number of streams to be supported by lowering the aggregate disk bandwidth requirements at the expense of buffer costs on the server. However, server buffering is not very useful when the bottleneck is the number of delivery channels.

2.2.2 Client Caching

Use of buffering in the client (bridging) to handle VCR actions can be a limited alternative to promotion [1]. If the length of the stream corresponding to the length of interaction can be buffered in the client, then it can be handled without a promotion. This technique can be very effective in a near-VOD [13] scenario where there are streams immediately preceding or succeeding the current stream. The client can switch to these on buffer overflow [1].

Bridging requires a considerable amount of buffering at the client, thereby increasing the cost of the customer premises equipment. However bridging is not a complete solution, but can be used in conjunction with other techniques to reduce the number of the channels required in scenarios where the duration of interaction is short.

2.3 Rate Adaptation

Rate adaptation (or adaptive piggy-backing) is a technique that tries to merge streams by varying their display rates. Golubchik et al. [9] observe that rate changes of $2-3\%$ by interpolation of frames and expansion or contraction of the total length of the movie by up to 8% are acceptable in commercial video playback. This leads to a notion of content progression rates, distinct from data delivery rate or frame rates. An accelerated content progression rate implies that the total duration of the video will be reduced and any given scene would occur earlier in time. However, this is different from fast-forward which involves an increased frame rate.

Golubchik et al. [9] analyze the three merging policies for adaptive piggy-backing – viz., odd-even reduction, simple merging and a greedy policy. In odd-even batching, consecutive arrivals are paired up if the second arrival occurs while the first is still within a catch-up

window. At most 50% of arrivals can be merged by this policy. The simple merging strategy assigns streams to merging groups. Each group is initiated by a leader and all succeeding arrivals occurring while the leader is within the catch-up window are assigned to the same group. The leader moves at a normal rate while the rest of the group tries to catch up by moving at faster rates. Both these policies suffer from the same drawbacks as that of initial batching described in Section 2.1.1.

The greedy policy attempts to merge streams as many times as possible – initially it works similar to the simple merging. However at each merge, a new "catch-up" window is computed. According to whether there are other streams within that window or not, the newly merged stream is assigned to a group or becomes a leader. The greedy policy results in the highest reduction in I/O bandwidth. We can also see that we need to support two different rates with any of these merging policies. In order to have the largest "catch-up" window possible, we need the difference between the playout rates to be as large as possible. This difference would however be limited by the maximum rate distortion acceptable to the user. With high request arrival rates in the order of 2 per minute, reduction in bandwidth of up to 80% has been reported by Golubchik et al., [9].

Rate adaptation is a means to truly support dynamic service aggregation. We argue that rate adaptation is necessary for solving the end-to-end service aggregation problem. Merging of streams reduces resource requirements both at the server and in the network. Batching can also be used in conjunction with adaptive piggy-backing for increased gains [9].

While conceptually rate adaptation is desirable, there are some difficulties in implementing rate adaptation. It is conceivable that on-line rate adaptation hardware could be used to generate content at multiple progression rates. As discussed earlier, this is an expensive proposition and is not a feasible alternative.

Another solution to content delivery at multiple rates is to store content in different formats. The extra storage necessary for this scheme makes it an expensive scheme. It is suggested that we limit merging to a given length of the movie to reduce storage requirements [9]. As discussed earlier, this method will still lead to a large storage overhead. Further, storage of multiple content formats poses additional problems when dynamic segment replication schemes are in effect.

From the foregoing discussion, we see that a practical rate adaptation technique should require only a single content format and no additional hardware. We propose a solution to this effect, in the next section.

3 Proposed Scheme for Rate Adaptation Using a Single Format

In this section, we describe the proposed storage layout and scheduling policy that permits us to generate two different content progression rates off of content stored in a single format. We achieve rate adaptation through scaling the quality of service – by dropping or intrapolation of frames to alter the content playout rate. In the following sub-sections, we explain the components strategies of our scheme. For the sake of clarity, we illustrate our scheme at the level of a single stream. Rate adaptation for each stream is implemented in an identical fashion.

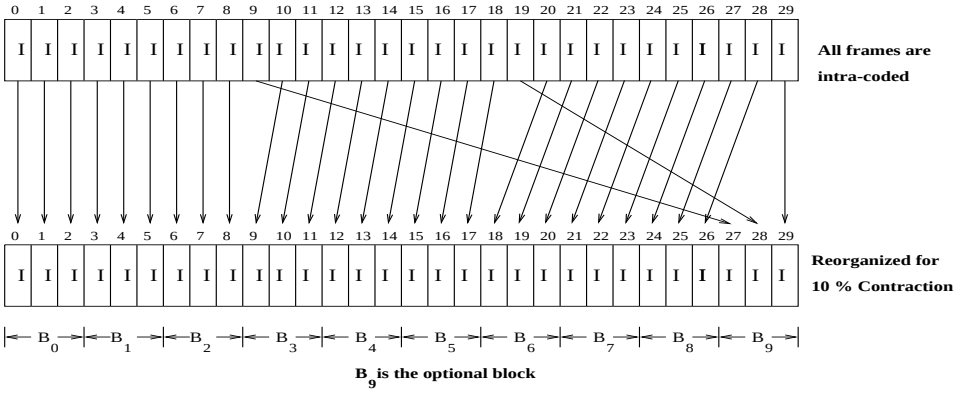

Figure 2: Content Reorganization for Intra-Coded Format

3.1 Basic Scheme for Large Stripe-sets

One of the central concepts of our rate adaptation scheme is our data placement policy. In this section, we introduce our scheme in the context of a multi-disk architecture. Disk striping is a common technique where contiguous data blocks are laid out on multiple disks in a cyclic fashion. The set of disks across which data is striped is called a stripe-set. Disk striping allows higher storage throughput to be achieved by reading the data blocks simultaneously from all disks of a given stripe-set.

Let i be the sequence number of the i^{th} media block to be played out for a given media stream. Let the stripe-set consist of N disks numbered $0, 1, 2, 3, ..., N-1$. Our placement policy is to place block i on disk number d, where

$$d = (i - \lfloor i/N \rfloor) \ \ mod \ \ N.$$

This placement policy is illustrated in Figure 1 for a six disk ($N = 6$) stripe-set. Note that blocks numbered 5 and 6 are placed on disk D_5.

The retrieval schedule for normal rate playout is as follows. The i^{th} media block in the reorganized format is retrieved in the ${s_i}^{th}$ scheduling interval where

$$s_i = \lfloor i/N \rfloor.$$

Thus N blocks ($N = 6$ in our example) are retrieved and played out in each interval and no blocks are dropped. Our scheme will result in the stripe blocks having different sizes.

Blocks satisfying

$$i \ \ mod \ \ N = N - 1$$

are *optional* blocks. In order to achieve a higher content progression rate, optional blocks are skipped and their next blocks retrieved instead. In our illustration, blocks numbered $5, 11, 17, 23, 29, 35, ...$ are optional and are dropped to achieve an accelerated content progression rate. In the ${s_i}^{th}$ scheduling interval, the optional block (to be skipped) occurs on the disk number

$$d = N - 1 - (s_i \ \ mod \ \ N),$$

which forms a cyclic sequence. In our example, blocks are skipped in the disk sequence $5, 4, 3, 2, 1, 0, 5, ...$

The block retrieval schedule is different for accelerated playout. The i^{th} media block, provided it is not an optional block, is retrieved in the ${s_i}^{th}$ scheduling interval where

$$s_i = \lfloor (i - \lfloor i/N \rfloor)/N \rfloor; \ \ (i \ \ mod \ \ N) \neq N - 1.$$

This means that with accelerated playout, block number 6 is read in the 0^{th} scheduling interval rather than in the 1^{st} scheduling interval. In either case, exactly one block of data is retrieved from each disk per scheduling interval and therefore the scheme is balanced. Data delivery occurs at the same frame rate at either content progression rate.

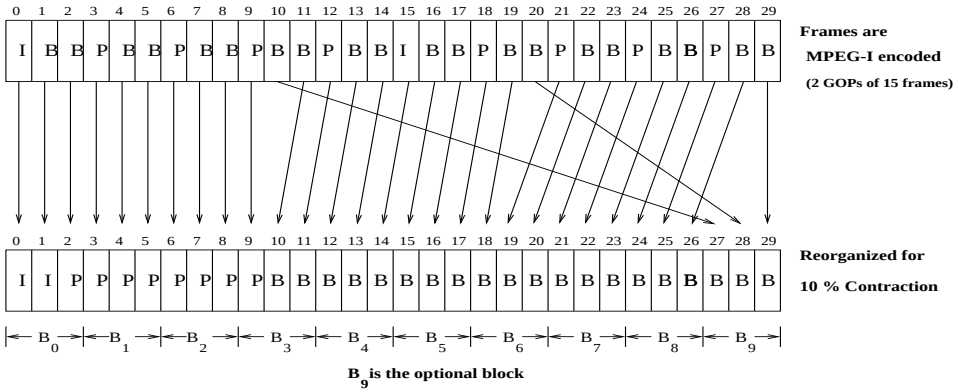

Figure 3: Content Reorganization for MPEG-1 Format

This basic placement policy is applicable when large stripe-sets can be used or when large contraction factors can be tolerated. A simple analysis reveals that the size of the stripe-set required for small contraction factors is very high. Suppose l is the normal length of the content in terms of number of frames. Let c be the content progression rate for normal playout in frames per second, and c' be the content progression rate for accelerated playout. The delivery rate in both cases is equal to c. c and c' are related by

$$c' = ((N+1) * c)/N.$$

If t is the scheduling interval, then the size of each block B is given by, $B = (c * t)/N$. The duration of normal playout is l/c and that of accelerated playout is l/c'. The reduction in duration or the contraction factor r is given by, $r = (c' - c)/c'$. For a given value of r, the maximum contraction factor permissible from quality requirements, the number of disks in the stripe-set, can be determined from $N = (1 - r)/r$. For a contraction factor of 5%, we require a 19-disk stripe-set with this scheme.

This scheme is extended to small stripe-sets in Section 3.4. Our scheme involves dropping optional blocks. If an optional block consists of a contiguous sequence of frames, dropping it results in large cuts in the video, causing a drastic loss of quality. We address this issue and propose solutions in the next section.

3.2 Content Reorganization Within a Stream

The idea of content reorganization to achieve QoS scaling is inspired by Chen et al. [4]. We make a new application of this concept to provide multiple content progression rates. Dropping of content imposes a restriction that the media access units (frames) to be dropped must lie entirely within the optional block. These units must also be smoothly distributed in the playout in order to minimize the effect on the output quality. Therefore, we reorganize the content so that the access units for the optional block are chosen at uniform spacings in the original stream. A shuffling strategy for use with intra-coded video formats like MJPEG is illustrated in Figure 2.

When content is inter-coded, frame dependencies seriously limit our choice of frames which could be dropped independently of others. In MPEG-1 encoded video, B frames are not required in the decoding of any other frames. I and P frames do not have this property. So our choice of frames for the optional block has to be made from B frames only. We also require that these frames be chosen at uniform spacings over the data to be retrieved in a given scheduling interval. Figure 3 illustrates a possible reorganization for MPEG-1 encoded video content. In this case, block B_9 is optional and consists of only B frames chosen at regular intervals along the playout. Trade-offs in using content reorganization are discussed in Section 4.

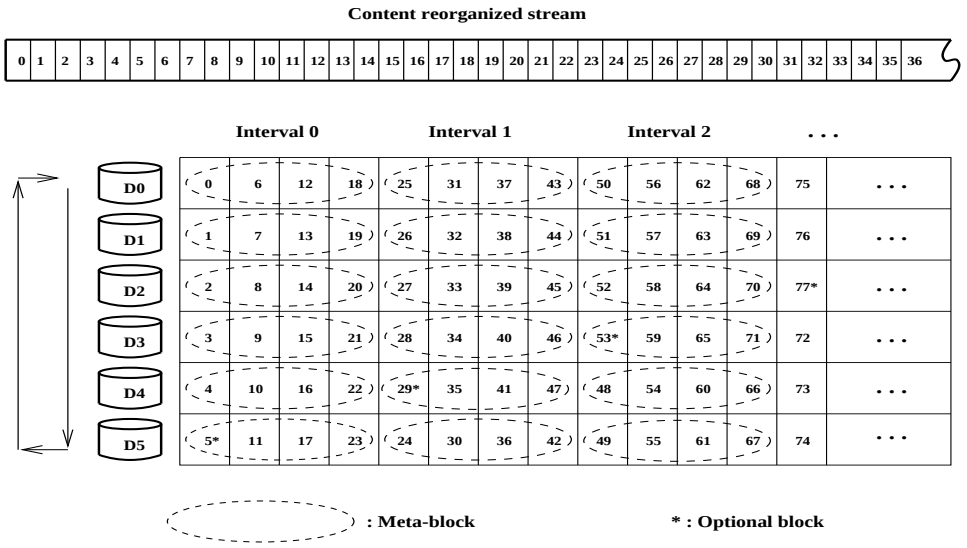

Figure 4: Placement Scheme and Normal Playout Schedule for Small Stripe-sets

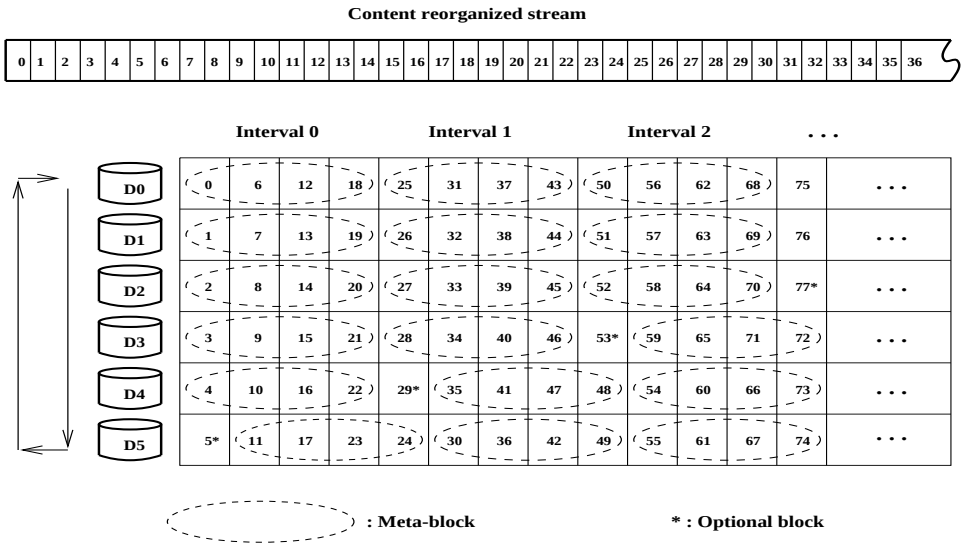

Figure 5: Accelerated Playout Schedule for Small Stripe-sets

3.3 Service Scaling During Peak Overload

Dynamic service aggregation is useful when peak demand coincides with high density access. There can be occasional high demand periods when the density of access is low. In other words, there are a large number of users who all request different content. The case for service scaling in such scenarios has been made by Chen et al. [2, 3].

Our proposal for rate adaptation works on the same philosophy of QoS scaling albeit in a different flavor. As a consequence, our scheme also supports limited service scaling during peak overloads. Service scaling is achieved as follows – during periods of overload, we do not read the optional blocks, rather, we scale down service by delivering content at a lowered frame rate. This means lesser network and disk bandwidth requirements, but at the same time a lowered QoS as well.

Our primary goal in this paper is to support rate adaptation for dynamic service aggregation. Service scaling is viewed only as a side benefit and therefore we make only this brief mention of the versatility of the proposed scheme.

3.4 Modified Scheme for Small Stripe-sets

The analysis in Section 3.1 showed that our scheme requires large stripe-sets to be effective. In this section, we propose a modification that permits us to employ our scheme with any stripe-set size greater than one.

We first introduce the notion of meta-blocks. A meta-block is a contiguous group of a fixed number of blocks that are retrieved from any disk in any given scheduling interval. We also define a parameter M, which is the number of blocks per meta-block. Let the stripe-set consist of N disks numbered $0, 1, 2, 3, ..., N-1$. Our modified placement policy is to place block i on disk number d, where

$$d = (i - \lfloor i/(N * M) \rfloor) \ \ mod \ \ N.$$

The data placement policy for this modified scheme with $(N = 6, M = 4)$ is shown in Figure 4. Note that blocks numbered 23 and 24 are placed on disk D_5.

The retrieval schedule for normal rate playout is as follows. The i^{th} media block is retrieved in the ${s_i}^{th}$ scheduling interval, where

$$s_i = \lfloor i/(N * M) \rfloor.$$

Thus N meta-blocks, or $N * M$ blocks are retrieved and played out in each interval and no blocks are dropped.

Optional blocks satisfy the condition,

$$i \ \ mod \ \ (N * M) = N - 1.$$

These are marked with an asterisk in Figure 4. We use the same content reorganization strategy explained in Section 3.2. The optional block numbered i consists of frames uniformly chosen from blocks numbered p to q, where

$$p = i - (i \ \ mod \ \ (N * M)) \text{ and } q = p - 1 + (N * M).$$

Optional blocks always occur at the beginning of the next meta-block. As before, the contents of the optional block have to be inserted in their respective positions before delivery. There

is an additional requirement that the blocks retrieved in the interval be arranged in the correct order before delivery.

In order to achieve a higher content progression rate, optional blocks are skipped and the meta-block beginning from the next block is retrieved. In our illustration in Figure 5, blocks numbered $5, 29, 53, 77, ...$ are optional and would be skipped to achieve the accelerated content progression rate. When block 5 is skipped, blocks numbered $11, 17, 23$ and 24 are retrieved as a single meta-block. In the ${s_i}^{th}$ scheduling interval, the optional block occurs on the disk number

$$d = N - 1 - (s_i \ \ mod \ \ N),$$

which forms a cyclic sequence. In our example, blocks would be skipped in the disk sequence $5, 4, 3, 2, 1, 0, 5, ...$

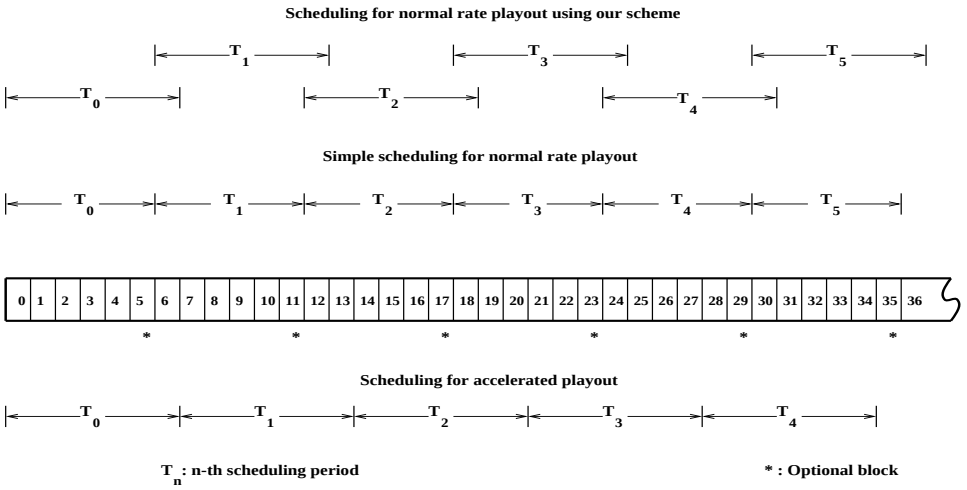

Figure 6: Placement Scheme for Single Disk Architecture

For accelerated rate playout, the ${i}^{th}$ media block, provided it is not an optional block, is retrieved in the ${s_i}^{th}$ scheduling interval where

$$s_i = \lfloor (i - \lfloor i/(N * M) \rfloor)/(N * M) \rfloor;$$
$$(i \ \ mod \ \ (N * M)) \neq N - 1.$$

In our example, during accelerated playout, blocks numbered 0 to 4 and 6 to 24 are read in the ${0}^{th}$ scheduling interval, while block 5 is skipped.

We have flexibility in choosing M, the number of blocks per meta-block. This allows us to work with an arbitrary stripe-set size N. Given N and r, the maximum contraction rate permissible, we can calculate M from the formula, $M = (1 - r)/(r * N)$. Our example in Figure 5 has a contraction factor of 4% and uses a 6-disk stripe-set, which leads to 4 sub-blocks per block. Thus the modified scheme allows us to use an arbitrary stripe-set size while retaining all the benefits of the original scheme.

3.5 Placement for Single Disks

In earlier sections, we detailed schemes for supporting multiple content rates from a single storage format for multi-disk architectures. There is some relevance for VOD servers that are designed around single disks (i.e., without disk striping) [4]. With this in mind, we present a single disk solution for the rate adaptation problem. In single-disk architectures, continuous media data should be placed contiguously to minimize seek overhead [4]. This is even more significant with modern disk drives since they cache entire tracks during a read to reduce rotational latencies. A detailed treatment of effects of disk parameters in continuous media scheduling can be found in Chen [4].

The data placement policy for single-disk architectures also employs the concept of optional blocks. However, the same scheduling policy cannot be used. With single-disk architectures, skipping optional blocks would result in additional seeks within each scheduling period and the overheads would seriously affect disk throughput. Consider, for example, that 0.5 seconds worth of data has to be read in each scheduling period. At MPEG-1 rates, this is about 96 KB. If the effective disk throughput is about 6 MB/s, then it takes approximately 31 ms to read the data for one stream. Typical seek latencies are about 8 ms and therefore, naive application of our rate adaptation scheme would result in a loss of throughput of up to 25%. Also, with the naive scheme, the bandwidths required for normal playout and accelerated playout would be different. This imbalance is undesirable from a disk scheduling perspective – the number of streams that can be supported would be a function of not only the rates, but also the number of streams playing at each rate.

The solution that we propose for the single-disk architecture is to use a larger scheduling budget that would enable an extra block to be read in each scheduling interval. With this enhanced scheduling budget, we will read one block beyond what is required to be delivered in the given scheduling period. We would discard the optional block or the extra block depending on whether we are delivering content at a normal rate or at an accelerated rate.

If we are playing out at normal rate, we re-read the extra block in the next period. Otherwise, we start reading from the next block. It is easy to see that the excess bandwidth required this way is about the same as the contraction factor which would be in the range of $5-8\%$. By this gambit, we get the benefit of bandwidth savings through aggregation. Also, the scheduling budget is the same for either content rate. Disk seeks for a given stream occur only between subsequent scheduling intervals. These seeks are inevitable whenever multiple streams are being sourced from the same disk and is not specific to our scheme.

We give an illustration of our scheduling scheme for a single-disk architecture in Figure 6. In our example, 6 blocks have to be read and delivered in every scheduling interval T_i. If no rate adaptation is done, the schedule shown in the middle would be used to achieve a normal playout. In our proposed scheme, we increase the scheduling budget and read 7 blocks in each interval, regardless of whether we deliver content at a normal rate or at an accelerated rate. Blocks $0-6$ are read in the scheduling interval T_0. To deliver content at an accelerated rate, we discard the optional block 5, before delivery. In the next interval T_1, we read blocks $7-13$ and discard block 11 and so on. For achieving a normal rate playout, we discard block 6 in T_0 and read blocks $6-12$ in T_1 and so on.

4 Discussion

In this section, we consider issues and trade-offs which can arise in practice and examine how they affect the applicability of the proposed scheme. The assumptions made in this work are also presented here.

The proposed scheme has several advantages. It leads to balanced placement of data on the disk and to a balanced loading of the disks and can easily integrate with dynamic segment replication policies since content is stored in a single format. No extra hardware or storage capacity is needed. During periods of overload, we can skip reading the optional blocks and scale down service by delivering content at a lowered frame rate.

In the paper we have considered that the stored content is of normal duration and an accelerated content progression rate is achieved by dropping frames. This is good for pay-per-view scenarios where there is an incentive to induce streams to complete quickly and release resources. This can be of significant advantage during loaded conditions with low density access – all streams can be played at an accelerated rate thereby releasing resources earlier. The same techniques can also be applied to an inverted scenario where content is

stored in an elongated format and acceleration through dropping gives normal playout. This may be a favorable model if users pay by the minute.

Previous work with video by frame dropping indicates that up to an 8% change in the content progression rate can be achieved [9]. For audio data, human perception is highly sensitive to drops. The audio content progression rate can be altered by compressing silent periods in the audio between talk-spurts or by manipulating the length of the cuts at scene transitions. Another technique is to use pitch-shifting with audio rate adaptation. Loss of lip sync within a scheduling period due to rate distortion can be maintained within limits. With our scheme any loss of sync will not be carried across scheduling periods. Except for some demanding applications like classical music, the approach will work for most commercial applications. In order to evaluate degradation in perceived quality due to rate adaptation we performed some experiments. We captured and encoded a video sequence using hardware in MPEG-1 format at 1.5 Mb/s. Rate acceleration was done in software. An acceleration of 6.67% was achieved by dropping 1 frame per GOP (of 15 frames) and 1 out of every 15 audio frames. Then the timestamps are also altered to reflect this change. Our results showed that the drops were difficult to distinguish from the normal MPEG-1 encoding/decoding anomalies.

A disadvantage is the need for off-line data placement due to content reorganization. Therefore, our scheme cannot be used with live video. This is not a limitation in interactive VOD scenarios and the cost of reorganization would be amortized by gains from service aggregation. Since our scheme stores content in a custom format, some on-line reorganization is also involved at the time of delivery. To enable quick on-line reorganization, meta-data must be stored resulting in a storage overhead of about 1 – 2%.

The business model for application of this technique is VOD in which it is highly likely that only one video format is used. So the fact that we require different schemes for different formats is not an issue. We have worked with MPEG system streams but the same concept is applicable to other formats as well. Multicast of video streams can be combined with service aggregation using the single disk approach in this paper. The single disk approach is provided for completeness of the solution and is applicable to small Web servers.

A final issue concerns "channel-switching." To implement rate adaptive merging, receivers must have the capability to switch channels smoothly under server control. Merging involves a channel switch when the movie is in progress. This must be done seamlessly and requires some buffering. No end-to-end negotiation is required regarding missing frames. The client or the transport end-point, whichever may be the case, always receives a system

stream.

5 Summary

In this paper, a scheme to support rate adaptation based on a novel content placement and scheduling strategy was proposed. This scheme requires only a single content format and is therefore economical in that it does not impose additional hardware or storage requirements. Furthermore, the scheme is balanced with respect to content placement and disk loading. This scheme makes rate adaptation a viable approach to dynamic service aggregation.

Future studies are proposed to determine the extent to which content can be contracted while keeping the degradation in QoS tolerable. This information will be useful in determining exact operating ranges and aggregation windows for our scheme. Future work includes an implementation of the proposed scheme on a media server to evaluate the efficacy of our scheme in practice.

References

[1] K. C. Almeroth and M. H. Ammar, "On the Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service," *IEEE Journal on Selected Areas in Communication,* Vol. 14, No. 6, pp. 1110-1122, Aug. 1996.

[2] H-. J. Chen, T. D. C. Little, D. Venkatesh, "A Storage and Retrieval Technique for the Provision of Scalable Delivery of MPEG-Encoded Video," *Journal of Parallel and Distributed Computing (Special Issue on Multimedia Processing and Technology),* Vol. 30, No. 2, pp. 180-189, Nov. 1995.

[3] H-. J. Chen, A. Krishnamurthy, T. D. C. Little, D. Venkatesh, "A Scalable Video-On-Demand Service for the Provision of VCR-Like Functions," *Proc. 2nd Intl. Conf. on Multimedia Computing and Systems, (ICMCS'95),* Washington DC, USA, pp. 65-72, May 1995.

[4] H-. J. Chen, "A Disk Scheduling Scheme and MPEG Data Layout Policy for Interactive Access from a Single Disk Storage Device," *Ph. D. Dissertation,* Dept. of Electrical, Computer and Systems Engineering, Boston University, Aug. 1995.

[5] A. Dan, P. Shahabuddin, D. Sitaram, D. Towsley, "Channel Allocation under Batching and VCR Control in Video-On-Demand Systems," *Journal of Parallel and Distributed Computing (Special Issue on Multimedia Processing and Technology),* Vol. 30, No. 2, pp. 168-179, Nov. 1995.

[6] A. Dan, D. Sitaram and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," *Proc. ACM Multimedia '94,* San Francisco, CA, USA, pp. 15-23, Oct. 1994.

[7] A. Dan, D. M. Dias, R. Mukherjee, D. Sitaram, R. Tewari, " Buffering and Caching in Large-Scale Video Servers," *Compcon '95*, San Francisco, CA, USA, Mar. 1995.

[8] D. K. Gifford, "Polychannel Systems for Mass Digital Communication," *Communications of the ACM,* Vol. 33, No. 2, pp. 141-151, Feb 1990.

[9] L. Golubchik, J. C. S. Lui and R. R. Muntz, "Adaptive Piggybacking: A Novel Technique for Data Sharing in Video-On-Demand Storage Servers," *Multimedia Systems,* ACM/Springer-Verlag, Vol. 4, pp. 140-155, 1996.

[10] J-. P. Nussbaumer, F. Schaffa, "Impact of Channel Allocation Policies on Quality of Service of Video on Demand over CATV," *Multimedia Tools and Applications,* Kluwer Academic Publishers, Vol. 2, pp. 111-131, 1996.

[11] W. D. Sincoskie, "System Architecture for a Large Scale Video on Demand Service," *Computer Networks and ISDN systems,* Vol. 22, pp. 155-162, 1991.

[12] R. Steinmetz, "Human Perception of Jitter and Media Synchronization," *IEEE Journal on Selected Areas in Communication,* Vol. 14, No. 1, pp. 61-72, Jan. 1996.

[13] D. Venkatesh and T. D. C. Little, "Prospects for Interactive Video-on-Demand," *IEEE Multimedia,* Vol. 1, No. 3, pp. 14-24, Fall 1994.

[14] D. Venkatesh and T. D. C. Little, "Dynamic Service Aggregation for Efficient Use of Resources in Interactive Video Delivery," *Lecture Notes in Computer Science, Vol. 1018, (Proc. of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video),* T. D. C. Little, R. Gusella, Eds., Springer-Verlag, pp. 113-116, Nov. 1995.

[15] H. Woo and C-. K. Kim, "Multicast Scheduling for VOD Services," *Multimedia Tools and Applications,* Kluwer Academic Publishers, Vol. 2, pp. 157-171, 1996.

[16] P. S. Yu, J. L. Wolf, H. Shachnai, "Design and Analysis of a look-ahead scheduling scheme to support pause-resume for video-on-demand applications," *Multimedia Systems,* ACM/Springer-Verlag, Vol. 3, pp. 137-149, 1995.