VLDB Journal manuscript No.
(will be inserted by the editor)

Efficient Support for Interactive Service in Multi-resolution VOD Systems

Kelvin K.W. Law[1], **John C.S. Lui**[1*], **Leana Golubchik**[2**]

[1] Department of Computer Science & Engineering,
The Chinese University of Hong Kong,
Shatin, NT, Hong Kong
[2] Department of Computer Science
University of Maryland at College Park
USA

Received: date / Revised version: date

Abstract *Advances in high speed networks and multimedia technologies have made it feasible to provide video-on-demand (VOD) services to users. However, it is still a challenging task to design a cost effective video-on-demand system that can support a large number of clients (who may have different quality of service (QoS) requirements) and, at the same time, provide different types of VCR functionalities. Although it has been recognized that VCR operations are important functionalities in providing VOD service, techniques proposed in the past for providing VCR operations may require additional system resources, such as extra disk I/O, additional buffer space as well as network bandwidth. In this paper, we consider the design of a VOD storage server that has the following features: (1) provision of different levels of display resolutions to users who have different QoS requirements, (2) provision of different types of VCR functionalities, such as fast forward and rewind, without imposing additional demand on the system buffer space, I/O bandwidth, and network bandwidth and, (3) guarantees of the load-balancing property across all disks during normal and VCR display periods. The above mentioned features are especially important because they simplify the design of the buffer space, I/O, and network resource allocation policies of the VOD storage system. The load balancing property also ensures that no single disk will be the bottleneck of the system. In this paper, we propose data block placement, admission control, and I/O scheduling algorithms as well as determine the corresponding buffer space requirements of the proposed VOD storage system. We show that the proposed VOD system can provide VCR and multi-resolution services to the viewing clients and at the same time maintain the load balancing property.*

[*] This work was supported in part by the RGC Earmarked Grant.
[**] This work was supported in part by the NSF CAREER grant CCR-98-96232.

1 Introduction

Advances in distributed multimedia systems and networking technologies have made it feasible to provide interactive digital services such as home shopping and video-on-demand (VOD) services [4]. Unlike a traditional television broadcasting service in which subscribers have no control during the program delivery sessions, users subscribing to VOD services will have more control capabilities. For instance, they can tailor their viewing preferences such as viewing dimensions and quality of video playback during a video session (e.g., such as video resolution and video frame rate).

Designing and implementing a cost-effective VOD server is a challenging task due to some fundamental characteristics of digital video. Firstly, the delivery of digital video must meet real-time requirements. Secondly, digital videos in general have high transfer bandwidth (both in terms of I/O bandwidth and network bandwidth) and high storage requirements. Although it has been recognized that provision of VCR functions is a major requirement for VOD service, implementing various VCR operations is not trivial. The reason being that during the VCR display period (e.g, during the period of fast forward or fast rewind operations), the VOD storage server may need to retrieve more information from the disks at a bandwidth which is several times higher than the I/O bandwidth that was allocated for normal display. This implies that VCR operations may introduce additional workload on the VOD storage server and on the communication network. This additional workload adds complexity to the design of the VOD system as well as to the network management algorithms. Thirdly, it is desirable to provide multi-resolution viewing configurations to different viewing clients. For example, if a client has a high-end display station or if the network connection bandwidth is not a constraint, then a client may want to watch the video at the highest display resolution. Since a digital video which is encoded using a non-scalable compression technique cannot support multi-resolution viewing feature, the VOD server would have to replicate the same set of videos with various resolutions and store the information on the storage system. Upon receiving a request from a user for a particular resolution, the corresponding video data would then be retrieved from the VOD storage system. It is clear that this type of an implementation wastes a great deal of storage space and adds to scheduling complexity, therefore such a VOD storage architecture is not considered to be cost-effective.

Based on the above considerations, we propose a VOD storage architecture that has the following features:

- provision of different levels of resolution to users who may have different quality of service requirements,
- provision of different types of VCR functionalities, such as fast forward and rewind, without imposing additional demands on the system buffers, I/O, and network transmission bandwidth,
- guarantees of a load-balancing property across the disk sub-system during the normal display and the VCR display periods. This property reduces the probability that a single disk becomes the system's bottleneck and thereby allows the system to fully utilize its resources and thus support a greater number of viewing customers simultaneously.

Our VOD storage system is based on a scalable video compression technique, which is known as *subband video coding* [15]. The appealing feature offered by subband video coding is that it provides multi-resolutions as well as multi-rate properties. These properties are very useful in providing a cost-effective VCR display service in that during the VCR service period, the system only needs to extract a subset of video frames and hence no additional system resources are needed[1]. Given that the video file is encoded using the subband coding technique, we present a data placement algorithm that can ensure a *load balancing* property for all disks during the normal as well as the VCR display periods. Due to this load balancing property, the VOD system can accept more viewing users, thus making the proposed VOD storage server more cost-effective. We present the I/O scheduling and viewer admission control algorithms as well as quantify the buffer space requirements of each viewer in the VOD system.

Let us begin with a brief survey of the published literature on VOD storage systems. In [2], the authors propose two methods, termed segment sampling and segment placement, so as to support variable bit rate browsing for MPEG-like video streams. In segment sampling, one out of m segments (a retrieval unit) is selected for display when a fast forward (or fast rewind) VCR function is desired and the fast forwarding speed is m times the normal display speed. In contrast, the segment placement scheme allocates segments to disks judiciously such that a completely uniform segment sampling across the disk array occurs. The drawback of this approach is that the quality of display during these VCR functions may not be acceptable. For example, viewers may experience jitter during video display and a large change of video scenery during the VCR display period. Also, that work only supports a single VCR speed while our approach can support multiple VCR speeds, depending on the QoS requirements of the viewers. In [3,8], the authors discuss implementation of I/O scheduling for VOD systems. They point out that if I/O scheduling is not carefully designed, starvation of data blocks might occur; they propose two algorithms where the main idea is to maintain a safe state transition (from normal display to a VCR function) and to avoid starvation. It should be noted that under these two schemes: 1) additional buffer resources are required and 2) the communication network experiences a sudden surge of traffic during the VCR display period. Consequently, data blocks might be dropped by the communication network because the data transmission bandwidth is higher than the initially allocated transmission bandwidth, which was allocated based on the effective bandwidth of the normal display periods only. In [12], authors propose using several buffer management techniques for providing constant viewing intervals as well as a limited form of VCR functionality. As long as the data blocks are pre-fetched and available in the memory buffers, no additional I/O resources are needed. However, the VOD storage server needs to pre-allocate a large amount of memory buffers so as to implement the proposed scheme. At the same time, it is possible to have a sudden surge of traffic on the network and thereby cause data packets to be dropped. Other works based on exploiting the characteristics of the MPEG coding technique are presented in [6]; however, all these works require additional I/O resources, such as buffer space as well as have the potential of introducing traffic workload fluctuations into the network. Recently, the subband coding techniques were proposed for implementing a VOD system that can support multiple video resolutions [1,3,14]. In [7], the authors studied two video layout strategies on disk arrays for achieving different degrees of parallelism and concurrency. However, to the best of our knowledge, there are no research results on how to provide VCR and load balancing features for a VOD storage server based on this multi-resolution property.

The organization of the remainder of this paper is as follows. In Section 2, we present the system architecture

[1] Exactly how this can be accomplished will be illustrated in the following sections.

as well as the fundamental concepts of subband coding. In Section 3, we present the proposed data placement algorithm of our VOD storage server and illustrate the load balancing property. In Section 4, the admission control and the disk scheduling algorithms are presented. In Section 5, we determine the buffer space requirements for each viewing client in the system, and in Section 6, we extend our data placement algorithm to a parallel disk system that has an arbitrary number of disks. Our conclusions are given in Section 7.

2 System Architecture and Subband Coding

In this paper, we consider a video on demand storage server which archives many objects of long duration, such as movies and educational training material. The storage server consists of a set of disks and a tertiary storage device, where the entire database resides on the tertiary device, and the more frequently accessed objects are cached on disks. When a request arrives to the system and the corresponding multimedia object is currently available on the disk storage sub-system, information retrieval can be made through the disks. On the other hand, if the multimedia object is not available on the disk storage sub-system, then it has to be retrieved from the tertiary storage sub-system to the secondary storage sub-system for display. This type of storage architecture is illustrated in Figure 1.

Subband coding techniques [9,11,13] are a class of scalable video compression algorithms capable of producing multiple resolutions of video bit streams [14]. In [10], the authors illustrate that the compression ratios of subband coding techniques are comparable to MPEG-2 compression. Subband coding techniques offer a high degree of scalability, allowing video streams to have a high number of display resolutions and different bit rates. This type of coding technique is based on multi-stage algorithms, where each stage uses the quadrature mirror filtering (QMF) technique to separate the input streams into two frequency spectrums, one for the high band and the other for the low band frequency spectrum. Spatial compression is achieved by processing the video signals through two stages of the compression algorithm, one for the x direction and the other for the y direction. Temporal compression is achieved by filtering and downsampling successive frames. By cascading several stages of filtering and downsampling, further compression of the video signal can be achieved [1,3].

Let us illustrate how the subband coding technique processes an input signal. The input signal is first downsampled into a set of streams, each of which is limited to a range of spatial frequencies. The filtered subband streams are further encoded by one or more coders for optimization. Reconstruction is achieved by first decoding the incoming encoded streams; the decoded subband streams are then added together to form an image. Figure 2 shows an example of subband compression with the two cascading filters scheme. On the encoding side, the original video signal is decomposed by an analysis filter bank, in which quadrature mirror filters (QMF) are implemented for alias free reconstruction into three subband streams. After passing through the first stage of the analysis filter, high resolution components (h) are filtered out. Medium resolution components (m) and low resolution components (l) are obtained by passing through the second analysis filter. These downsampled streams are coded by possibly multiple coders before being stored in the storage server. In this example, the data rate (or throughput) of the encoded high, medium, and low resolution components are 0.25 MB/s, 0.125 MB/s, and 0.125 MB/s, respectively. During data retrieval, the encoded streams are retrieved from the storage server and are decoded by the corresponding decoders. The low resolution video (l), which has a data rate of 0.125 MB/s, is obtained by simply using the decoded low resolution components. By adding the decoded low resolution and medium resolution components, the medium resolution video stream ($l + m$) now has a data rate of 0.25 MB/s. The high resolution video stream ($l + m + h$) is obtained by combining the low, medium, and high decoded resolution components, and this high resolution video stream has a data rate of 0.5 MB/s.

3 Data Placement for the Multiple Resolutions Video File

In this section, we propose a file layout scheme for video objects which were compressed using subband coding, which results in the following features:

- provision of multiple resolutions for users so that different levels of QoS can be maintained,
- provision of load balancing during normal and VCR display periods and,
- during the VCR display period, no additional system resources, such as I/O and network bandwidth, need to be allocated so as to deliver the video data.

Let us assume that our video server is a disk array storage system with d homogeneous[2] disks. Let $T^{max}_{seek}(\mathcal{U})$, $T^{max}_{rotational}(\mathcal{U})$, and $T^{max}_{transfer}(\mathcal{U})$ be the maximum seek time, rotational latency, and transfer time for $\mathcal{U}$ bytes of data from a disk, respectively. Therefore, the total latency of retrieving $\mathcal{U}$ bytes of data is :

$$T^{max}(\mathcal{U}) = T^{max}_{seek}(\mathcal{U}) + T^{max}_{rotational}(\mathcal{U}) + T^{max}_{transfer}(\mathcal{U}) \quad (1)$$

Suppose that our proposed VOD storage server can support $n > 0$ display resolutions. Furthermore, let us assume that a given video $\mathcal{V}$ has g segments such that:

$$\mathcal{V} = \{\mathcal{V}_0 \cup \mathcal{V}_1 \cup \cdots \mathcal{V}_i \cdots \cup \mathcal{V}_{g-1}\}$$

[2] The disks are homogeneous in the sense that they have the same data transfer characteristics.

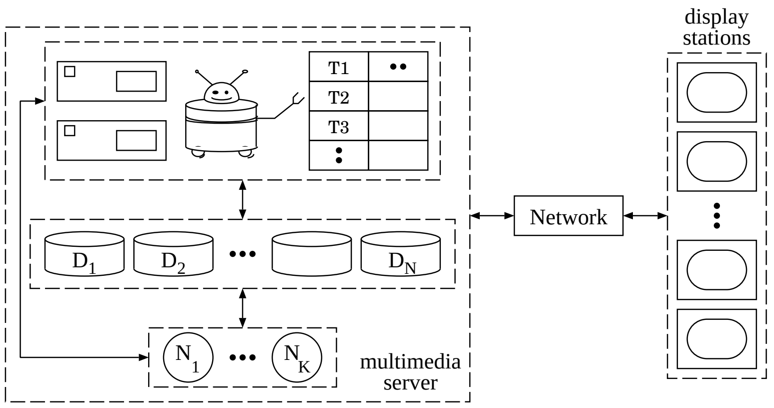

Fig. 1 Multimedia Storage Server Architecture.

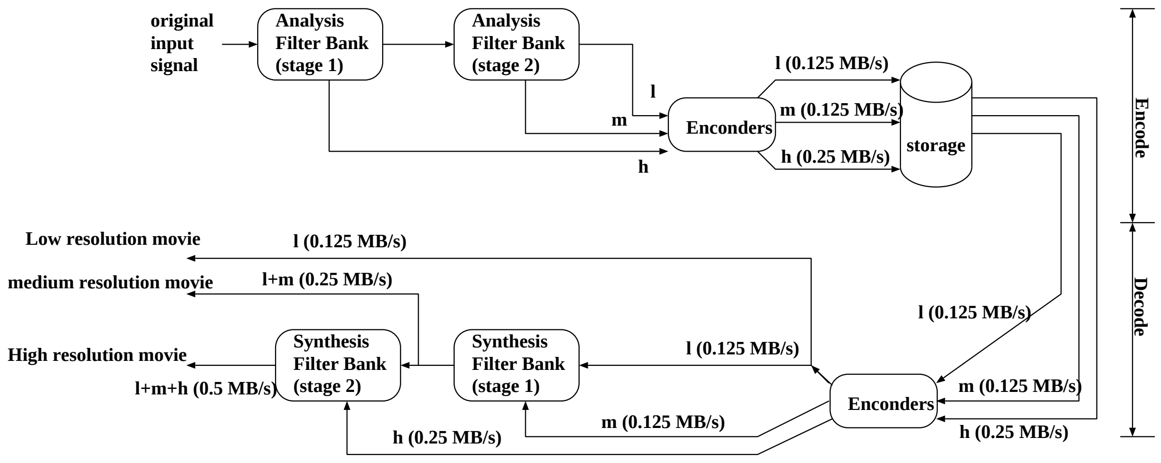

Fig. 2 Illustration for subband coding technique for three resolution components

where $\mathcal{V}_i$ is the ith segment of video $\mathcal{V}$ and $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$. Each segment $\mathcal{V}_i$ has up to n display resolutions of data:

$$\mathcal{V}_i = \{\mathcal{V}_i^0 \cup \mathcal{V}_i^1 \cup \cdots \cup \mathcal{V}_i^{n-1}\}$$

where $\mathcal{V}_i^j$ represents the *additional* data blocks needed to display the jth resolution of the $\mathcal{V}_i$ segment (e.g., to view the jth display resolution of the ith segment, we need to retrieve $\{V_i^0 \cup \cdots \cup V_i^j\}$). We refer to these additional data blocks as the jth *enhancement* for the ith segment. In general, the enhancement $\mathcal{V}_i^{j+1}$ contains data blocks required to form a higher display resolution than the enhancement $\mathcal{V}_i^j$ for $0 \le j < n-1$ at the ith segment. Each $\mathcal{V}_i^j$ is composed of b_j disk blocks[3] such that:

$$\mathcal{V}_i^j = \{r_{i,0}^j, r_{i,1}^j, \ldots, r_{i,b_j-1}^j\}$$

where $r_{i,k}^j$ is the kth disk block of $\mathcal{V}_i^j$. Figure 3 illustrates the data block layout of the *first* eight segments[4] of the video file $\mathcal{V}$, which offers up to $n = 4$ display resolutions. Given the above notation, the first segment of $\mathcal{V}$ has the following data blocks:

$$\begin{aligned}
\mathcal{V}_0 &= \{\mathcal{V}_0^0 \cup \mathcal{V}_0^1 \cup \mathcal{V}_0^2 \cup \mathcal{V}_0^3\} \\
\mathcal{V}_0^0 &= \{r_{0,0}^0\}\ ,\ b_0 = 1 \quad ; \quad \mathcal{V}_0^1 = \{r_{0,0}^1\}\ ,\ b_1 = 1 \\
\mathcal{V}_0^2 &= \{r_{0,0}^2, r_{0,1}^2\}\ ,\ b_2 = 2 \\
\mathcal{V}_0^3 &= \{r_{0,0}^3, r_{0,1}^3, r_{0,2}^3, r_{0,3}^3\}\ ,\ b_3 = 4
\end{aligned}$$

Let $\mathcal{V}_i^0$ be the *base resolution* for segment i. It is important to note that enhancement $\mathcal{V}_i^j$, $0 < j \le n-1$, cannot form a complete displayable segment by itself. In other words, $\mathcal{V}_i^j$ is only a portion of the whole displayable segment. To display a segment for resolution j, it requires the reading of data blocks from the base resolution as well as all the data blocks from resolutions 1 to j. Let $\mathcal{R}_i^j$ be the jth display resolution retrieval set for the ith

[3] The data block size is chosen such that the overhead due to seek and rotational latency is not more than 5% of the overall disk response time, or we can combine adjacent level resolutions data into a disk block so as to improve the I/O bandwidth efficiency.

[4] We only need to show the disk assignment for the first eight segments because the remaining segments will have the same disk assignments as illustrated in Figure 3.

Segment / Disk	0	1	2	3	4	5	6	7
0	$r^0_{0,0}$	$r^1_{0,0}$	$r^2_{0,0}$	$r^2_{0,1}$	$r^3_{0,0}$	$r^3_{0,1}$	$r^3_{0,2}$	$r^3_{0,3}$
1	$r^3_{1,0}$	$r^3_{1,1}$	$r^3_{1,2}$	$r^3_{1,3}$	$r^0_{1,0}$	$r^1_{1,0}$	$r^2_{1,0}$	$r^2_{1,1}$
2	$r^2_{2,0}$	$r^2_{2,1}$	$r^0_{2,0}$	$r^1_{2,0}$	$r^3_{2,0}$	$r^3_{2,1}$	$r^3_{2,2}$	$r^3_{2,3}$
3	$r^3_{3,0}$	$r^3_{3,1}$	$r^3_{3,2}$	$r^3_{3,3}$	$r^2_{3,0}$	$r^2_{3,1}$	$r^0_{3,0}$	$r^1_{3,0}$
4	$r^1_{4,0}$	$r^0_{4,0}$	$r^2_{4,0}$	$r^2_{4,1}$	$r^3_{4,0}$	$r^3_{4,1}$	$r^3_{4,2}$	$r^3_{4,3}$
5	$r^3_{5,0}$	$r^3_{5,1}$	$r^3_{5,2}$	$r^3_{5,3}$	$r^1_{5,0}$	$r^0_{5,0}$	$r^2_{5,0}$	$r^2_{5,1}$
6	$r^2_{6,0}$	$r^2_{6,1}$	$r^1_{6,0}$	$r^0_{6,0}$	$r^3_{6,0}$	$r^3_{6,1}$	$r^3_{6,2}$	$r^3_{6,3}$
7	$r^3_{7,0}$	$r^3_{7,1}$	$r^3_{7,2}$	$r^3_{7,3}$	$r^2_{7,0}$	$r^2_{7,1}$	$r^1_{7,0}$	$r^0_{7,0}$

Fig. 3 A template for assigning a 4−resolutions video file onto $d = 8$ disks

segment, we have $\mathcal{R}_i^j = \bigcup_{k=0}^{j} \mathcal{V}_i^k$. Using the video file assignment example in Figure 3, we have

$$\mathcal{R}_0^3 = \mathcal{V}_0 = \{\mathcal{V}_0^0 \cup \mathcal{V}_0^1 \cup \mathcal{V}_0^2 \cup \mathcal{V}_0^3\}$$
$$\mathcal{R}_0^2 = \{\mathcal{V}_0^0 \cup \mathcal{V}_0^1 \cup \mathcal{V}_0^2\}$$
$$\mathcal{R}_0^1 = \{\mathcal{V}_0^0 \cup \mathcal{V}_0^1\} \quad ; \quad \mathcal{R}_0^0 = \{\mathcal{V}_0^0\}$$

The number of disk blocks in $\mathcal{R}_i^j$ of any segment i is equal to $\sum_{k=0}^{j} b_k$. In addition, we let $disk(r_{i,k}^j)$ denote the disk number, ranging from 0 to $d-1$, that has been allocated for storing the disk block r_{ik}^j. We also define $DISK(\mathcal{R}_i^j)$ to be the set of disks storing $\mathcal{R}_i^j$. Using the example in Figure 3, we have $disk(r_{0,0}^1) = 1$ and $DISK(\mathcal{R}_0^2) = \{0, 1, 2, 3\}$.

Definition 1 *The retrieval sets $\mathcal{R}_i^j$ and $\mathcal{R}_k^l$ are non-overlapping iff $DISK(\mathcal{R}_i^j) \cap DISK(\mathcal{R}_k^l) = \emptyset$.*

Referring to the example in Figure 3, $\mathcal{R}_0^1$, $\mathcal{R}_1^1$, $\mathcal{R}_2^1$ and $\mathcal{R}_3^1$ are non-overlapping because $DISK(\mathcal{R}_0^1) = \{0, 1\}$, $DISK(\mathcal{R}_1^1) = \{4, 5\}$, $DISK(\mathcal{R}_2^1) = \{2, 3\}$, and $DISK(\mathcal{R}_3^1) = \{6, 7\}$.

Definition 2 *A normal display rate is defined to be the rate corresponding to display resolution r, where $0 \leq r < n$, which is set by the viewing client upon his/her admission into the VOD system.*

This implies that upon admission, the user can set the normal display to be at any display resolution r where $0 \leq r \leq n-1$. In this case, the normal display of the i^{th} segment of a video file requires the VOD storage server to retrieve all disk blocks in the retrieval set $\mathcal{R}_i^r$. Referring to the example in Figure 3, if a user sets the normal display resolution at $r = 2$ upon his admission into the VOD storage server, assuming that the viewer starts viewing the video at segment 0, Figure 4 depicts the data block retrieval and delivery process for servicing this client. In this figure, the horizontal line represents time and the vertical arrows mark the time instances when the server has finished retrieving a segment of video data at display resolution 2. The time interval between any two vertical arrows is termed a *time slot.* All video blocks of a displayable segment must be retrieved within a time slot. In this example, the server retrieves data blocks of the retrieval set $\mathcal{R}_i^2$, for $i \geq 0$, from the parallel disk system at the i^{th} time slot. To view the video at the normal display resolution, the system needs to retrieve data blocks in $\mathcal{R}_i^2$ by the end of time slot i, $\forall i \geq 0$. The data blocks retrieved are first put into the system buffers before transmission to the viewing client (buffer management will be discussed in Section 5). During the course of video viewing, the viewer can request VCR functions, such as fast forward or fast rewind. These VCR operations can be accomplished by retrieving more segments per time slot, with these segments having a display resolution $r' < r$.

Definition 3 *The fast forward and rewind VCR rate is defined to be the rate corresponding to display resolution r', where $0 \leq r' < r < n$ and r is the normal display resolution of an admitted client. The VCR function can be implemented if the system can retrieve multiple consecutive segments of the video file at resolution r' within the same time slot.*

For example, Figure 5 illustrates the data retrieval and delivery process for the VCR display resolution $r' = 0$, given that the normal display resolution is $r = 2$. In a latter part of the paper, we illustrate how the VCR functionality can be supported efficiently based on the data block assignment algorithm.

In general, if the normal display is set at display resolution r, a user has the option of up to $r - 1$ different VCR viewing speeds, which can be expressed as:

$$VCR\_Speed(r, r') = \left\lfloor \frac{\sum_{j=0}^{r} b_j}{\sum_{j=0}^{r'} b_j} \right\rfloor \tag{2}$$

where $0 \leq r' < r \leq n-1$, In other words, the viewing client can choose up to r' resolutions (where $0 \leq r' < r \leq n-1$) for different VCR speeds. Since the number of disk blocks for display resolution r' is $\sum_{i=0}^{r'} b_i$, which is

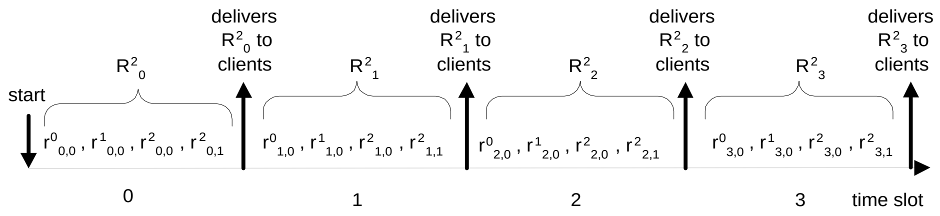
delivers R^2_0 to clients
delivers R^2_1 to clients
delivers R^2_2 to clients
delivers R^2_3 to clients
R^2_0
R^2_1
R^2_2
R^2_3
start
$r^0_{0,0}, r^1_{0,0}, r^2_{0,0}, r^2_{0,1}$
$r^0_{1,0}, r^1_{1,0}, r^2_{1,0}, r^2_{1,1}$
$r^0_{2,0}, r^1_{2,0}, r^2_{2,0}, r^2_{2,1}$
$r^0_{3,0}, r^1_{3,0}, r^2_{3,0}, r^2_{3,1}$
0
1
2
3
time slot

Fig. 4 Data retrieval and delivery process for normal display at display resolution $r = 2$

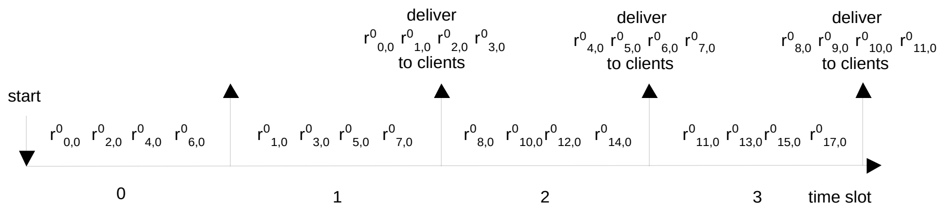
deliver $r^0_{0,0}$ $r^0_{1,0}$ $r^0_{2,0}$ $r^0_{3,0}$ to clients
deliver $r^0_{4,0}$ $r^0_{5,0}$ $r^0_{6,0}$ $r^0_{7,0}$ to clients
deliver $r^0_{8,0}$ $r^0_{9,0}$ $r^0_{10,0}$ $r^0_{11,0}$ to clients
start
$r^0_{0,0}$ $r^0_{2,0}$ $r^0_{4,0}$ $r^0_{6,0}$
$r^0_{1,0}$ $r^0_{3,0}$ $r^0_{5,0}$ $r^0_{7,0}$
$r^0_{8,0}$ $r^0_{10,0}$ $r^0_{12,0}$ $r^0_{14,0}$
$r^0_{11,0}$ $r^0_{13,0}$ $r^0_{15,0}$ $r^0_{17,0}$
0
1
2
3
time slot

Fig. 5 Data retrieval and delivery process for VCR display at display resolution $r' = 0$

smaller than the number of disk blocks for display resolution $r = \sum_{i=0}^{r} b_i$, the VOD storage server can retrieve more segments per unit time slot (for lower display resolution) and thereby obtains a faster viewing speed.

To illustrate, consider the video file in Figure 3. If upon admission, the user requests a normal display resolution of $r = 3$, then in order to view segment $\mathcal{V}_0$, the system needs to retrieve disk blocks in $\mathcal{R}_0^3 = \{\mathcal{V}_0^0 \cup \mathcal{V}_0^1 \cup \mathcal{V}_0^2 \cup \mathcal{V}_0^3\}$ or:

$$\bigcup_{l=0}^{3} \bigcup_{j=0}^{b_l} r_{0,j}^l$$

If a VCR function is desired, then the user can choose the following VCR speeds: $VCR\_Speed(3,2) = 8/4 = 2$, $VCR\_Speed = (3,1) = 8/2 = 4$, and $VCR\_Speed(3,0) = 8/1 = 8$. Suppose that the viewer chooses the VCR speed to be twice as fast as that of normal display. Then the system will retrieve disk blocks in $\mathcal{R}_0^2 \cup \mathcal{R}_1^2 = \{\mathcal{V}_0^0 \cup \mathcal{V}_0^1 \cup \mathcal{V}_0^2\} \cup \{\mathcal{V}_1^0 \cup \mathcal{V}_1^1 \cup \mathcal{V}_1^2\} = \{\cup_{l=0}^{2} \cup_{j=0}^{b_l} r_{0,j}^l\} \cup \{\cup_{l=0}^{2} \cup_{j=0}^{b_l} r_{1,j}^l\}$ in one time slot and thereby achieve a VCR viewing speed which is twice the normal display speed. It is important to note that during the VCR period, the system *does not retrieve or transmit more data* to the network (as compared to the amount of data transmitted during the normal display). Due to this property, we do not have to allocate extra I/O bandwidth or network bandwidth during the VCR display period. This property is very important because it can simplify the design of the buffer and the network bandwidth allocation procedures.

In addition to providing normal display and VCR functionalities, our proposed VOD storage server can maintain load balance during both normal and VCR display periods. This is accomplished by constructing appropriate data layout and scheduling schemes which ensure that the amount of I/O bandwidth needed by a stream from a disk is the same over a fixed period of time, whether the stream requests normal or VCR playback. We first give the formal definition of the load balancing property in our VOD storage system.

Definition 4 *The retrieval process for a display resolution r is considered to be load balanced over any $\mathcal{P}$ consecutive segments if the VOD storage server retrieves the same amount of data for each disk in the system in $P_r = \frac{d}{\sum_{i=0}^{r} b_i}$ time slots.*

Note that in general, the video data block retrieval process has to satisfy the following conditions in order to achieve the load balancing property:

1. $\cup_{i=k}^{\mathcal{P}+k-1} DISK(\mathcal{R}_i^r) = \{0, 1, \ldots d\}$, and
2. $\cap_{i=k}^{\mathcal{P}+k-1} DISK(\mathcal{R}_i^r) = \emptyset$

where k is any segment number in the video. The first condition indicates that all disks in the parallel disk system are involved in the retrieval process of the $\mathcal{P}$ consecutive segments and the second condition ensures that these consecutive segments are non-overlapping.

We use the example in Figure 3 to illustrate the concept. If $r = 3$, then the VOD storage system is obviously load balanced for every segment retrieved. On the other hand, if a user starts a VCR function with $r' = 1$ at segment 0, then the VOD system needs to retrieve data blocks $\{r_{0,0}^0, r_{0,0}^1\}$, $\{r_{1,0}^0, r_{1,0}^1\}$, $\{r_{2,0}^0, r_{2,0}^1\}$, $\{r_{3,0}^0, r_{3,0}^1\}$. Since these four segments are non-overlapping, they can be retrieved in parallel and a VCR speed which is four times the speed of the normal display is achieved. As illustrated in Figure 3, regardless of the value of r', the disks remain balanced after the retrieval of 8 consecutive segments. Let us first illustrate how we can assign the video blocks of a video file which has $n = 4$ display resolutions on the proposed VOD storage system, and then we will present the general data placement algorithm.

3.1 Data Placement for a 4-resolution video file

It is important to observe that depending on the value of the display resolution r, the load balancing property is maintained after reading a certain number of consecutive video segments. For instance, using the example we have in Figure 3, if the viewer chooses a display resolution of $r = 3$ upon his admission to the VOD system, then each segment retrieval requires reading one data block from each disk; the load balancing feature is therefore maintained after each segment retrieval. On the other hand, if $r = 2$, then retrieving segments in $\mathcal{V}_0^2$ requires reading one data block from disks 0 to 3 and the retrieval of segment $\mathcal{V}_1^2$ requires reading one data block from disks 4 to 7; then the load balancing property is maintained after reading two consecutive segments. In general, for a given value of a display resolution r, the load balancing property is maintained after reading P_r consecutive segments where

$$P_r = \frac{d}{\sum_{i=0}^{r} b_i} \qquad 0 \le r < n \qquad (3)$$

Given the example illustrated in Figure 3, we can easily observe that each disk retrieves the same number of disk blocks after reading 8 consecutive segments, regardless of the values of the display resolution r.

We assume that the underlying video files satisfy the following conditions:

$$\sum_{i=0}^{r} b_i \bmod \sum_{i=0}^{r-1} b_i = 0 \qquad 1 \le r \le n-1 \qquad (4)$$

The implication of the above conditions is that the block size of the higher layer has to be some integer multiple of the block size of the lower layer. These conditions can ensure that during the VCR display, we can access multiple segments of data at a time and these consecutive segments are stored in some non-overlapping disks. It is important to point out that the above conditions are not restrictive in practice. For example, consider a 3-resolutions video file; then choices for the block size of (b_0, b_1, b_2) are: { (1,1,2),(1,1,4),(1,1,6),(1,1,8),(1,1,10),... (1,2,3),(1,2,6),(1,2,9),(1,2,12), (1,2,15),..., (1,3,4),(1,3,8), (1,3,12),(1,3,16),...} The above illustrates that there are *many* choices for choosing the block size for b_i. For example, if $(b_0, b_1, b_2) = (1, 1, 2)$, we can achieve two different VCR speeds which are two times and four times the normal display (refer to Equation (2)). For $(b_0, b_1, b_2) = (1, 1, 4)$, we can achieve VCR speeds which are three times and six times the normal display. Based on the mutli-resolution coding technique, we can compress different subband levels to the appropriate block size based on the VCR speed requirement.

The basic idea of the video block assignment is that once we determine the disk mapping for the i^{th} display resolution, we can then determine the disk assignment for the enhancement data blocks in resolution $i + 1$. To illustrate, consider a 4−resolutions video file in Figure 3. Let us assign the video data blocks, starting from the highest resolution first.

resolution = 3 : Since the total number of data blocks in the retrieval set $\mathcal{R}_i^3$ is equal to $\sum_{i=0}^{3} b_i = 8$, we have to use all d disks for this data block assignment. Although we know that we have to assign the data blocks in $\mathcal{R}_i^3$ to all d disks, we cannot determine which data block in $\mathcal{R}_i^3$ is assigned to which disk until the data block assignments corresponding to lower display resolutions have been determined.

resolution = 2 : To assign the data blocks for the retrieval set in $\mathcal{R}_i^2$, we have to ensure that if the viewer selects the normal display resolution $r = 3$ upon his admission, he can choose $r' = 2$ for his VCR speed later. To provide this particular VCR speed, the VOD system must be able to retrieve $VCR\_Speed(3, 2) = 8/4 = 2$ consecutive segments of display resolution $r' = 2$ in parallel. This implies that two consecutive retrieval sets $\mathcal{R}_i^2$ and $\mathcal{R}_{i+1}^2$, for $i \ge 0$, must be *non-overlapping.* One way to accomplish this is to assign the retrieval set $\mathcal{R}_i^2$ to disks $0, 1, 2, 3$ and the retrieval set $\mathcal{R}_{i+1}^2$ to disks $4, 5, 6, 7$. We can repeat this assignment in a round robin fashion so that any two consecutive retrieval sets of display resolution 2 are non-overlapping. According to our example, if we choose to assign the retrieval set $\mathcal{R}_0^2$ to the first four disks, then the enhancement data blocks in $\mathcal{V}_i^3$ can be assigned at this point. For example, the blocks in enhancement $\mathcal{V}_0^3 = \{r_{0,0}^3, r_{0,1}^3, r_{0,2}^3, r_{0,3}^3\}$, are assigned to disks $4, 5, 6$, and 7, respectively. The data block assignment of the enhancement to resolution 3 is illustrated in Figure 6.

resolution = 1 To assign the data blocks for the retrieval set in $\mathcal{R}_i^1$, we have to consider two cases:

- **case 1:** If the viewer selects a display resolution of $r = 3$ upon his admission and if he later chooses $r' = 1$ for the VCR speed, the VOD system must be able to retrieve $VCR\_Speed(3, 1) = 8/2 = 4$ consecutive segments of display resolution $r' = 1$ in parallel. Consequently, any four consecutive segments, $\mathcal{R}_i^1$, $\mathcal{R}_{i+1}^1$, $\mathcal{R}_{i+2}^1$, and $\mathcal{R}_{i+3}^1$, for $i \ge 0$, must be non-overlapping.
- **case 2:** If the viewer selects a display resolution of $r = 2$ upon his admission and if later he chooses $r' = 1$ for the VCR speed, the VOD system must be able to retrieve $VCR\_Speed(2, 1) = 4/2 = 2$ consecutive segments of display resolution $r' = 1$ in parallel. As a result, any two consecutive segments, $\mathcal{R}_i^1$ and $\mathcal{R}_{i+1}^1$ for $i \ge 0$, must be non-overlapping.

Clearly, if the condition in case 1 is satisfied, the condition in case 2 also holds. One way to accomplish the assignment is to put the data blocks of the retrieval set $\mathcal{R}_i^1$ on disks 0 and 1, the data blocks of $\mathcal{R}_{i+1}^1$ on disks 4 and 5, the data blocks of $\mathcal{R}_{i+2}^1$ on disks 2 and 3, and the data blocks of $\mathcal{R}_{i+3}^1$ on disks 6 and 7.

Segment / Disk	0	1	2	3	4	5	6	7
0	R^2_0				$r^3_{0,0}$	$r^3_{0,1}$	$r^3_{0,2}$	$r^3_{0,3}$
1	$r^3_{1,0}$	$r^3_{1,1}$	$r^3_{1,2}$	$r^3_{1,3}$	R^2_1			
2	R^2_2				$r^3_{2,0}$	$r^3_{2,1}$	$r^3_{2,2}$	$r^3_{2,3}$
3	$r^3_{3,0}$	$r^3_{3,1}$	$r^3_{3,2}$	$r^3_{3,3}$	R^2_3			
...	...	...	...	...	...	...	...	...

Fig. 6 Data block assignment for enhancement of resolution $r = 3$.

Segment / Disk	0	1	2	3	4	5	6	7
0	R^1_0		$r^2_{0,0}$	$r^2_{0,1}$	$r^3_{0,0}$	$r^3_{0,1}$	$r^3_{0,2}$	$r^3_{0,3}$
1	$r^3_{1,0}$	$r^3_{1,1}$	$r^3_{1,2}$	$r^3_{1,3}$	R^1_1		$r^2_{1,0}$	$r^2_{1,1}$
2	$r^2_{2,0}$	$r^2_{2,1}$	R^1_2		$r^3_{2,0}$	$r^3_{2,1}$	$r^3_{2,2}$	$r^3_{2,3}$
3	$r^3_{3,0}$	$r^3_{3,1}$	$r^3_{3,2}$	$r^3_{3,3}$	$r^2_{3,0}$	$r^2_{3,1}$	R^1_3	
...	...	...	...	...	...	...	...	...

Fig. 7 Data block assignment for enhancement data blocks of resolution $r = 2$

Once this assignment is made, the assignment of the enhancement data blocks of resolution $r = 2$ can be determined. For example, we can assign $\{r^2_{0,0}, r^2_{0,1}\}$ to disks 2 and 3, $\{r^2_{1,0}, r^2_{1,1}\}$ to disks 6 and 7, $\{r^2_{2,0}, r^2_{2,1}\}$ to disks 0 and 1, and $\{r^2_{3,0}, r^2_{3,1}\}$ to disks 4 and 5 as illustrated in Figure 7.

resolution $= 0$ To assign the data blocks of the retrieval set in $\mathcal{R}^0_i$, we must consider the following:

- **case 1:** $r = 3$ and $r' = 0$. In this case, the VOD system must be able to retrieve $VCR\_Speed(3, 0)$ $= 8$ consecutive segments in parallel. This implies that the disk assignment of data blocks in any eight consecutive segments, $\mathcal{R}^0_i \cdots \mathcal{R}^0_{i+7}$, must be non-overlapping.
- **case 2:** $r = 2$ and $r' = 0$. In this case, the VOD system must be able to retrieve $VCR\_Speed(2, 0)$ $= 4$ consecutive segments in parallel. This implies that the disk assignment of data blocks in any four consecutive segments, $\mathcal{R}^0_i \cdots \mathcal{R}^0_{i+3}$, must be non-overlapping.
- **case 3:** $r = 1$ and $r' = 0$. In this case, the VOD system must be able to retrieve $VCR\_Speed(1, 0)$ $= 2$ consecutive segments in parallel. This implies that the disk assignment of data blocks in any two consecutive segments, $\mathcal{R}^0_i$ and $\mathcal{R}^0_{i+1}$, must be non-overlapping.

As long as we can satisfy the condition in case 1, all conditions in other cases can be satisfied. The following disk assignment can ensure the above conditions: $\mathcal{V}^0_0 = \{r^0_{0,0}\}$ to disk 0, $\mathcal{V}^0_1 = \{r^0_{1,0}\}$ to disk 4, $\mathcal{V}^0_2 = \{r^0_{2,0}\}$ to disk 2, $\mathcal{V}^0_3 = \{r^0_{3,0}\}$ to disk 6, $\mathcal{V}^0_4 = \{r^0_{4,0}\}$ to disk 1, $\mathcal{V}^0_5 = \{r^0_{5,0}\}$ to disk 5, $\mathcal{V}^0_6 = \{r^0_{6,0}\}$ to disk 3, and $\mathcal{V}^0_7 = \{r^0_{7,0}\}$ to disk 7. Again, once we know the assignment of the retrieval set $\mathcal{R}^0_i$, we can then determine the disk assignment for the enhancement data blocks of resolution $r = 1$. The final assignment of our 4-resolution video file is illustrated in Figure 3. Note that for the remaining segments of the video file, we can use the same assignment pattern for each set of 8 consecutive segments and achieve the properties described above.

3.2 Data Block Assignment Algorithm for Homogeneous Video Files

In this section, we show the data block assignment algorithm for a VOD system that can support n display resolutions. Our data block assignment algorithm is based on the following assumptions:

(A1) : The number of disks in our VOD system is: $d = \sum_{i=0}^{n-1} b_i$.

(A2) : $\sum_{i=0}^{j} b_i \bmod \sum_{i=0}^{j-1} b_i = 0$ for $1 \leq j < n$.

Assumption (A1) ensures that there is a correct number of disks for retrieving all data blocks at the highest resolution[5] while assumption (A2) ensures that consecutive segments are non-overlapping during the VCR display period. Given these two assumptions, we propose the following data block assignment algorithm.

The assignment algorithm has two main steps: 1) create a mapping table and 2) assign the video data blocks to the disks using this mapping table. A mapping table which we call a *template* is first created when the VOD server performs the disk assignment for a video file. A template is actually a map of data blocks to disks with a finite number of segments. The number of segments

[5] We will relax this assumption in Section 6.

depends on the number of parallel disks d, and the maximum number of display resolutions, n, that the video file can support. The template is in the form of a table with size $\frac{d}{b_0} \times d$, where $\frac{d}{b_0}$ is the row size (or the number of initial segments of the video file) while d is the number of disks in the system. Figure 3 depicts a template (of size 8×8) for a video file supporting up to four display resolutions and having eight parallel disks in the system. Note that once the disk blocks are assigned for the first d/b_0 segments, the i^{th} segment of the video file will have the same disk block assignment as the $(i \bmod \frac{d}{b_0})^{th}$ row of the template. For example, for the system illustrated in Figure 3, data blocks for segments 8, 16, 24, ..., will have the same disk block assignment as the first row of the template in Figure 3.

Note that there are *two advantages* to using a template. Firstly, the size of the template is small and therefore we can store the template in the VOD server's main memory without utilizing too much buffer space. Secondly, any data block assignment or determination of a disk number during the data block retrieval can be translated to a simple template lookup. Therefore, it is computationally efficient to locate the disk number for any data block of the video file.

The main idea behind template construction is similar to the one we gave in Section 3.1. We start the video block assignment from the highest display resolution. Based on the same principles, the template construction consists of two steps : 1) set up *base templates* and 2) assign the enhancement data blocks to the base template.

A base template for display resolution j, where $0 \leq j < n$, is a collection of non-overlapping disk assignment sets dedicated to display resolution j. The disk assignments in a base template for display resolution j must satisfy the following conditions:

$$(C1) : \cup_{i=0}^{P_j-1} DISK(\mathcal{R}_i^j) = \{0, 1, \ldots, d-1\} \quad 0 \leq j < n$$
$$(C2) : \cap_{i=0}^{P_j-1} DISK(\mathcal{R}_i^j) = \emptyset \qquad 0 \leq j < n$$

where $P_j = \frac{d}{\sum_{i=0}^{j} b_i}$ is the number of consecutive segments for the base template of display resolution j. The first condition ensures that after retrieving P_j consecutive segments of display resolution j, the disk system is load balanced (i.e., the same amount of data will be retrieved from every disks in the system). The second condition ensures that P_j consecutive segments of display resolution j are non-overlapping so that it is possible to issue VCR operations without increasing the load on the VOD server or the network. Since the final template has $\frac{d}{b_0}$ rows, given a base template of display resolution j (which has P_j segments), we can repeat this base template $(d/b_0)/(d/\sum_{i=0}^{j} b_i) = \frac{\sum_{i=0}^{j} b_i}{b_0}$ times in the final template. Whenever the base templates dedicated to two consecutive display resolutions, say j and $j-1$, are made, we can then determine the disk assignment for the enhancement of resolution j at segment i by applying the following formula:

$$DISK(\mathcal{V}_i^j) = DISK(\mathcal{R}_i^j) - DISK(\mathcal{R}_i^{j-1}) \qquad (5)$$

The disk assignment can be constructed by recursively applying Equation (5) to all display resolutions j, where $0 \leq j < n$, for all segments i in the final template, where $0 \leq i \leq \frac{d}{b_0}$. The video data block assignment algorithm for a general $n-$resolution system is given in Figure 9.

To illustrate the algorithm, let us consider the construction of the four base templates of a 4$-$resolutions video file on eight disks, as illustrated in Figure 8. The first step of template construction is to build the four base templates of the four different display resolutions. Our approach is to assign the data blocks from the highest display resolution first, i.e., display resolution 3 in this example. Let $D_j[i]$ denote the disk assignment set dedicated to display resolution j and segment i in the base template. Since the number of disk blocks for display resolution 3 is 8, the data blocks of each segment in resolution 3 have to be allocated to all $d = 8$ disks. We use $D_3[0]$ to denote the disk assignment set dedicated to the display resolution 3 at segment 0 in the base template. Hence, $D_3[0] = \{0, 1, 2, 3, 4, 5, 6, 7\}$ (refer to Figure 8(a)). Since the final template has 8 rows, we repeat this base template $\frac{\sum_{i=0}^{3} b_i}{b_0} = \frac{8}{1}$ times in the final template. Therefore, we have $D_3[i] = D_3[0]$ for $1 \leq i \leq 7$. Next, we create the base template of display resolution 2. Since the number of blocks for display resolution 2 is 4, which is half of the number of data blocks for display resolution 3, the base template for display resolution 2 has two segments, namely $D_2[0]$ and $D_2[1]$. To satisfy the two conditions $(C1)$ and $(C2)$ stated above, one possible disk assignment for the base template of display resolution 2 is as follows (refer to Figure 8(b)):

$$D_2[0] = \{0, 1, 2, 3\} \text{ and } D_2[1] = \{4, 5, 6, 7\}$$

Again, we can repeat this base template $\frac{\sum_{i=0}^{2} b_i}{b_0} = 4$ times in the final template. Similarly, there are $P_1 = 4$ consecutive segments for the base template of display resolution 1. To satisfy conditions $(C1)$ and $(C2)$, we can use the following possible assignment (refer to Figure 8(c)): $D_1[0] = \{0, 1\}, D_1[1] = \{4, 5\}, D_1[2] = \{2, 3\}$, and $D_1[3] = \{6, 7\}$ and we can repeat this base template two times in the final template. Finally, the base template of display resolution 0 has eight segments and one possible disk assignment is (refer to Figure 8(d)):

$$D_0[0] = \{0\}, D_0[1] = \{4\}, D_0[2] = \{2\}, D_0[3] = \{6\},$$
$$D_0[4] = \{1\}, D_0[5] = \{5\}, D_0[6] = \{3\} \text{ and } D_0[7] = \{7\}$$

After creating the base templates for these four display resolutions, we proceed to assign the enhancement data blocks of each resolution using Equation (5). Since $D_j[i] = DISK(\mathcal{R}_i^j)$ and $D_{j-1}[i] = DISK(\mathcal{R}_i^{j-1})$, the equation becomes:

$$DISK(\mathcal{V}_i^j) = D_j[i] - D_{j-1}[i] \qquad (6)$$

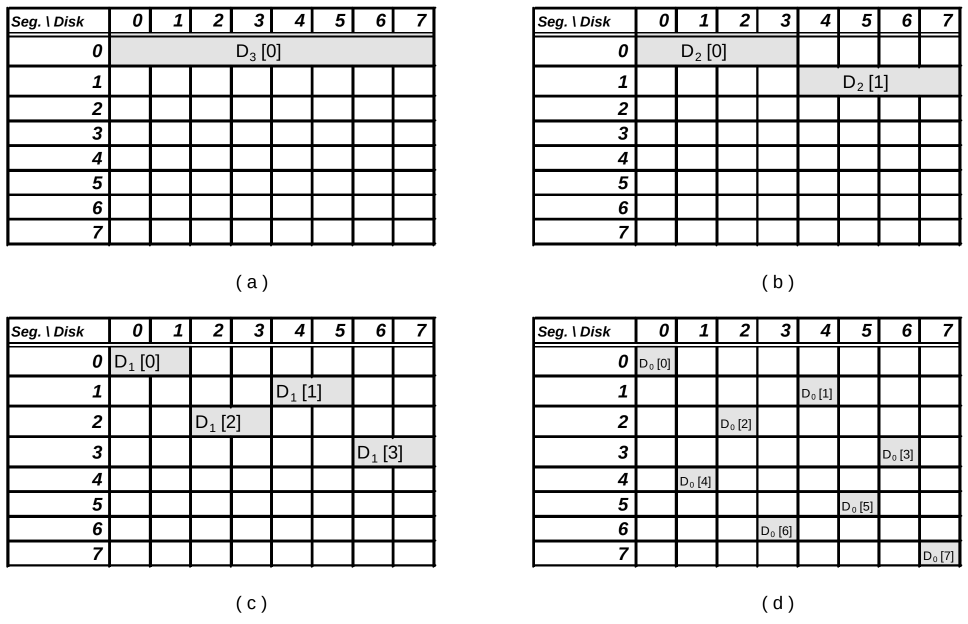

Fig. 8 The construction of a base template for a 4 display resolutions video file.

for $0 \leq j < n$ and $0 \leq i < d/b_0$. By applying this equation, we can obtain the final template, as shown in Figure 3.

4 Disk Scheduling and Admission Control

In the previous section, we have developed a framework of a multi-resolution video file system as well as the corresponding data block assignment algorithm. This framework does not make any assumptions about how the system admits a new client and how it schedules the retrievals of video data blocks. In this section, we present a disk scheduling algorithm for data retrieval of different video streams which efficiently utilizes the aggregate bandwidth of the parallel disk system. Then we present an admission control algorithm, which decides whether a new client can be admitted into the system. Given the disk scheduling and the admission control algorithms, we then define and prove that our system exhibits the load balancing property.

4.1 Disk Scheduling Algorithm

Suppose that the VOD system stores c video files that offer the same degree of display resolutions. Let us denote each video file by m_x where $0 \leq x < c$. Upon receiving the first playback request for video file m_x from a client, the server creates a stream $S_{id}^{m_x}$ for that client, where id is a unique number identifying a particular video stream in the system. Each stream $S_{id}^{m_x}$ is associated with a transfer rate of PR data sets per round. A data set consists of a segment or fragment of several segments of video data, depending on the user's display mode. For example, if a user is viewing the video under the normal display mode, then the data set consists of one segment of video. On the other hand, if the user is viewing the video under the VCR mode, then the data set consists of several segments, depending on the resolution r' that the viewing client chooses. As stated in the previous section, once the viewing client is admitted to the VOD storage server with display resolution r, the system retrieves the same amount of information (e.g., bytes) per time slot regardless of the normal or the VCR display modes.

Our VOD system retrieves data for each video stream in a sequence of *service rounds*. During each service round, the server retrieves a fixed amount of data for each client from the disks. In general, a single stream does not require the entire bandwidth capacity of the disk subsystem; thus, we "group" such streams into *sessions*, where streams in the same session access non-overlapping disks and each session comes as close as possible to utilizing the entire bandwidth capacity of the disk subsystem. Therefore, we divide a service round into multiple sessions such that for each session, all the disks in the storage server can be utilized in parallel (i.e., if there are enough requests to utilize all of the disk I/O bandwidth in the same session). Within each session, the storage server can schedule data retrieval of multiple clients such that the data requests of each clients are to *non-overlapping* disks.

To clarify the concept of *sessions*, we consider the following illustration. From Equation (1), we know that the total disk latency of retrieving $\mathcal{U}$ bytes, which is the size of a video data block, is $T^{max}(\mathcal{U})$ seconds, and therefore the *minimum* duration for a session is $T^{max}(\mathcal{U})$. Each disk should start to fetch the required data set at the beginning of a session, and the data set should be avail-

```
Algorithm $\mathcal{V}$   Video Object Placement Algorithm
Procedure n_RESOLUTION_VIDEO_OBJECT_ASSIGNMENT $(n, d)$

Begin
1   Let $D_{n-1}[0] \leftarrow \{0, 1, ...., d-2, d-1\}$;
2   For $k \leftarrow n-2$ to $0$ Do

3     For $l \leftarrow 0$ to $P_{k+1} - 1$ Do /* base template construction */
4       $X \leftarrow D_{k+1}[l]$;
5       For $i \leftarrow 0$ to $\frac{P_k}{P_{k+1}} - 1$ Do
6         For $j \leftarrow 0$ to $\sum_{m=0}^{k} b_m - 1$ Do
7           $x \leftarrow$ the least element in X;
8           $D_k[P_{k+1} \times i + l] \leftarrow D_k[P_{k+1} \times i + l] \cup \{x\}$;
9           $X \leftarrow X - \{x\}$;
10        Endfor
11      Endfor
12    Endfor

13    For $l \leftarrow 0$ to $\frac{P_0}{P_k} - 1$ Do /* disk assignment for the enhancement data blocks */
14      For $i \leftarrow 0$ to $P_{k+1} - 1$ Do
15        For $j \leftarrow 0$ to $\frac{P_k}{P_{k+1}} - 1$ Do
16          $\mathcal{V}^{k+1}_{l*P_k + P_{k+1} \times j + i} \leftarrow D_{k+1}[i] - D_k[P_{k+1} \times j + i]$;
17        Endfor
18      Endfor
19    Endfor

20  Endfor

21  For $l \leftarrow 0$ to $P_0$ Do
22    $\mathcal{V}^0_l \leftarrow D_0[l]$
23  Endfor

24  Assign $R^k_j$ components ,$\forall k \in \{0, 1, \ldots, k\}$, for segment $j$ according to $\mathcal{V}^k_j$
End
```

Fig. 9 N-resolution video block assignment algorithm when the number of disks in the system is equal to $d = \sum_{i=0}^{n-1} b_i$.

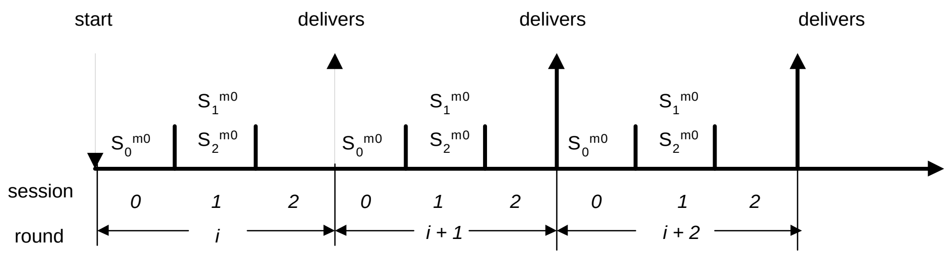

Fig. 10 Schedules for the disk retrievals for streams $S_0^{m_0}$,$S_1^{m_0}$, and $S_2^{m_0}$.

able in the buffers by the end of the session. We denote the data blocks of display resolution k of video stream $S_{id}^{m_x}$ that can be retrieved during session l in service round i by $Segment(S_{id}^{m_x}, i, l, k)$, where the resulting set is equivalent to $\mathcal{R}_j^k$ of video m_x. Consider the example shown in Figure 10, and suppose that the parallel disk system contains a video m_0 using the layout in Figure 3. Let $user_0$, $user_1$ and $user_2$ be the viewers requesting video m_0 with normal display resolutions $r = 3, 1$, and 1, respectively. The VOD system will create three streams, $S_0^{m_0}$, $S_1^{m_0}$, and $S_2^{m_0}$ and allocate enough buffer space[6] to hold the data being retrieved. Assuming that $S_0^{m_0}$ and $S_1^{m_0}$ are requesting segment 0 and $S_2^{m_0}$ is requesting segment 1 at round i, the VOD system will allocate the data retrieval for $S_0^{m_0}$ to session 0 (because $S_0^{m_0}$ requires data retrieval from all eight disks) and for $S_1^{m_0}$ and $S_2^{m_0}$ to

[6] The amount of buffer space needed will be quantified in Section 5.

session 1. The reason that the VOD system can assign both streams $S_1^{m_0}$ and $S_2^{m_0}$ to session 1 is because the retrieval sets of these two streams are non-overlapping. After round i, the data retrieval process will repeat itself in the following rounds.

In general, the number of streams that can be supported during a service round depends on the display QoS requirements of all the admitted viewing clients. Since a video segment of a higher display resolution requires more data blocks than one of a lower display resolution, more disks are involved in the data retrieval of a higher resolution video segment. If a client selects his normal display resolution as the highest display resolution possible, then all d disks are involved every time a segment of video is retrieved and hence no disk bandwidth is available to serve other clients in the same session. If the selected normal display resolution, however, is not the highest possible display resolution, then the server does not need to involve all d disks in each session for a stream. Consequently, it is possible to schedule data retrieval of more than one video stream in a single session. The degree of parallel retrieval, therefore, depends on the chosen display resolutions of the associated streams. In this paper, we classify the video streams into two categories, namely, 1) the simultaneous access streams and 2) the sequential access streams. If two or more video streams are scheduled to access the parallel disks simultaneously in a single session of the same service round, then we call these simultaneous access streams; otherwise we call these sequential access streams. Formally, we have:

Definition 5 *The data retrieval process for $S_i^{m_x}$ and $S_j^{m_y}$ can be scheduled in the same session if*

$$DISK(Segment(S_i^{m_x}, a, b, k)) \cap DISK(Segment(S_j^{m_y}, a, b, k')) = \emptyset \qquad \forall a, b, k, k'$$

where a is the service round number, b is the session number of service round a, and k and k' are the normal display resolutions for $S_i^{m_x}$ and $S_j^{m_y}$, respectively.

Definition 6 *The data retrieval process for $S_i^{m_x}$ and $S_j^{m_y}$ can only be scheduled under two different sessions if*

$$DISK(Segment(S_i^{m_x}, a, b, k)) \cap DISK(Segment(S_j^{m_y}, a, b, k')) \neq \emptyset \qquad \forall a, b, k, k'$$

where a is the service round number, b is the session number of service round a, and k and k' are the normal display resolutions for $S_i^{m_x}$ and $S_j^{m_y}$, respectively.

It is important to point out that it is not necessary for a stream to be scheduled in the same session in each service round, i.e., retrieval of stream $S_i^{m_x}$ which is scheduled for retrieval in session l of service round k may be scheduled in session j of service round $k+1$ where $l \neq j$. This is because two simultaneous access streams in the current service round may become two sequential access streams in a later service round. For example, using the retrieval schedule illustrated in Figure 10, if another video m_1 in the storage system exists and its final template is shown in Figure 11, then consider the scenario where a new client requests the video service of video m_1 with normal display resolution $r = 1$. The VOD server creates a stream $S_3^{m_1}$ for the new incoming client and schedules the I/O resources for the stream. It is clear that stream $S_3^{m_1}$ must be scheduled in session 2 of service round i because:

$$\begin{aligned}
&DISK(Segment(S_0^{m_0}, i, 0, 3)) \cap \\
&\qquad DISK(Segment(S_3^{m_1}, i, 0, 1) \neq \emptyset \quad \text{and} \\
&DISK(Segment(S_1^{m_0}, i, 1, 1)) \cap \\
&\qquad DISK(Segment(S_3^{m_1}, i, 1, 1) \neq \emptyset \quad \text{and} \\
&DISK(Segment(S_2^{m_0}, i, 1, 1)) \cap \\
&\qquad DISK(Segment(S_3^{m_1}, i, 1, 1) \neq \emptyset
\end{aligned}$$

However, all three streams $S_1^{m_0}$, $S_2^{m_0}$, and $S_3^{m_1}$ can be scheduled in session 1 of service round $i+1$ because

$$\begin{aligned}
&DISK(Segment(S_1^{m_0}, i+1, 1, 1)) \cap \\
&\qquad DISK(Segment(S_3^{m_1}, i+1, 1, 1)) = \emptyset \quad \text{and} \\
&DISK(Segment(S_2^{m_0}, i+1, 1, 1)) \cap \\
&\qquad DISK(Segment(S_3^{m_1}, i+1, 1, 1)) = \emptyset
\end{aligned}$$

As a result, streams $S_1^{m_0}$, $S_2^{m_0}$, and $S_3^{m_1}$ become simultaneous access streams in round $i+1$; this scenario is depicted in Figure 12.

In general, a new stream can be scheduled in service round i and session l if the following conditions are satisfied. Let $S(i, l)$ denote the set of all admitted streams in round i and session l. The VOD system can schedule the data retrieval for a new stream s at display resolution r in round i and session l iff:

$$DISK(s, i, l, r) \cap DISK(x, i, l, j) = \emptyset \qquad (7)$$

For all $x \in S(i, l)$ and for $0 \leq j < n$.

In order to facilitate the I/O scheduling order of the data block retrievals for various streams as well as to simplify the scheduling of a new request, the VOD server uses a *retrievals scheduling map* to accomplish the I/O scheduling process. Figure 13 illustrates the scheduling map for the four streams in Figure 12. In the scheduling map, the first column corresponds to the service round number, the second column corresponds to the session number and the remaining columns correspond to disk numbers (where $d = 8$ in our example). If a disk is being utilized in session l of service round i, the corresponding entry in the map is marked with the stream number. For example, stream $S_0^{m_0}$ requires the disk set $DISK(Segment(S_0^{m_0}, i, 0, 3)) = \{0, 1, 2, 3, 4, 5, 6, 7\}$ to transfer data during session 0 of service round i, and consequently, all disks are marked with $S_0^{m_0}$ in session 0 of service round i. It is important to point out that the

Segment / Disk	0	1	2	3	4	5	6	7
0	$r^3_{0,1}$	$r^3_{0,2}$	$r^3_{0,3}$	$r^0_{0,0}$	$r^1_{0,0}$	$r^2_{0,0}$	$r^2_{0,1}$	$r^3_{0,0}$
1	$r^1_{1,0}$	$r^2_{1,0}$	$r^2_{1,1}$	$r^3_{1,0}$	$r^3_{1,1}$	$r^3_{1,2}$	$r^3_{1,3}$	$r^0_{1,0}$
2	$r^3_{2,1}$	$r^3_{2,2}$	$r^3_{2,3}$	$r^2_{2,0}$	$r^2_{2,1}$	$r^0_{2,0}$	$r^1_{2,0}$	$r^3_{2,0}$
3	$r^2_{3,1}$	$r^0_{3,0}$	$r^1_{3,0}$	$r^3_{3,0}$	$r^3_{3,1}$	$r^3_{3,2}$	$r^3_{3,3}$	$r^2_{3,0}$
4	$r^3_{4,1}$	$r^3_{4,2}$	$r^3_{4,3}$	$r^1_{4,0}$	$r^0_{4,0}$	$r^2_{4,0}$	$r^2_{4,1}$	$r^3_{4,0}$
5	$r^0_{5,0}$	$r^2_{5,0}$	$r^2_{5,1}$	$r^3_{5,0}$	$r^3_{5,1}$	$r^3_{5,2}$	$r^3_{5,3}$	$r^1_{5,0}$
6	$r^3_{6,1}$	$r^3_{6,2}$	$r^3_{6,3}$	$r^2_{6,0}$	$r^2_{6,1}$	$r^1_{6,1}$	$r^0_{6,0}$	$r^3_{6,0}$
7	$r^2_{7,0}$	$r^1_{7,1}$	$r^0_{7,0}$	$r^3_{7,0}$	$r^3_{7,1}$	$r^3_{7,2}$	$r^3_{7,3}$	$r^2_{7,0}$

Fig. 11 A template for video file m_1.

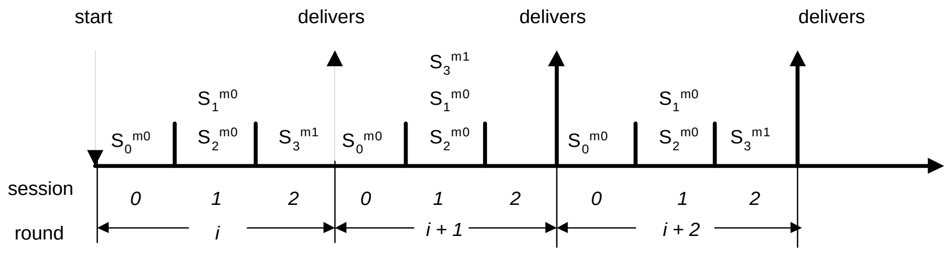

Fig. 12 Adding a stream $S_3^{m_1}$ to the existing schedule in Figure 10.

length of the retrieval scheduling map is $P_0 = d/b_0$ service rounds because all the disk assignment patterns are repeated for every P_0 segments as explained in the previous section. Therefore, the retrievals scheduling map can be stored in main memory of the VOD storage server without using too much of the VOD storage server's memory resources.

4.2 Admission Control

Let us describe how the VOD system can admit a new stream into the system. In general, there are two conditions to satisfy:

1. *The continuous display requirement of a stream.* The time between displaying two consecutive segments is $\frac{1}{PR}$. Therefore, the system has to place the next segment of video data into the buffers within $\frac{1}{PR}$ seconds. Assume the system has admitted k streams already and is deciding whether it is possible to admit a new stream. Then this condition states that the total disk latency of $k+1$ streams should be less than or equal to the playback time interval between consecutive segments of any stream.
2. *Individual disk bandwidth requirement.* Since the bandwidth of a disk is limited, it can only support a maximum number of streams at a given time interval. If currently there are M streams using disk j for data retrieval, then we have to ensure that the disk has sufficient bandwidth for retrieving $M+1$ streams.

Based on the two conditions described above, we can obtain the admission control algorithm as shown in Figure 14. The algorithm consists of two parts. The first part (lines 1 through 7 of Figure 14) examines whether the disk sets of the to-be-admitted client, $DISK(R^{res}_{seg\_start+i})$, of stream $k+1$ satisfy above given two conditions where

$$T^{max}_{j,l,i}(\mathcal{U}) = \begin{cases} T^{max}(\mathcal{U}) & \text{if stream } l \text{ is scheduled on disk } j \text{ in round } i. \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

$$B_j = \text{maximum bandwidth of disk } j = \frac{T^{max}_{transfer}(\mathcal{U})}{T^{max}(\mathcal{U})} \quad (9)$$

If these two conditions are satisfied, the algorithm advances to the second part (lines 8 through 18 of Figure 14) to schedule the incoming stream into the disk slots of the scheduling map. For instance, consider the scenario in Figure 12. Suppose that the number of maximum sessions supported by each disk is 4 and that streams $S_0^{m_0}$, $S_1^{m_0}$, and $S_2^{m_0}$ are already admitted before service round i. After the system has received a request for allocating resources for a new stream $S_3^{m_1}$ at service round i, it invokes the Admit_User procedure to check whether the request of stream $S_3^{m_1}$ can be fulfilled. The first part of the algorithm verifies the possibility of adding load on the disk sets $DISK(R_0^1) = \{3,4\}$, $DISK(R_1^1) = \{7,0\}$, $DISK(R_2^1) = \{5,6\}$, $DISK(R_3^1) = \{1,2\}$, $DISK(R_4^1) = \{3,4\}$, $DISK(R_5^1) = \{7,0\}$, $DISK(R_6^1) = \{5,6\}$, and $DISK(R_7^1) = \{1,2\}$ at service rounds $i, i+1, i+2, i+3, i+4, i+5, i+6$, and $i+7$, respectively. In this case, the test is passed because the maximum load on each disk at any given service round is 2. The second part of the

round	session	0	1	2	3	4	5	6	7
0	0	S^{m0}_{0}							
	1	S^{m0}_{1}		▨	▨	S^{m0}_{2}		▨	▨
	2	▨	▨	▨	S^{m1}_{3}		▨	▨	▨
1	0	S^{m0}_{0}							
	1	S^{m1}_{3}	▨	S^{m0}_{2}		S^{m0}_{1}		▨	S^{m1}_{3}
	2	▨	▨	▨	▨	▨	▨	▨	▨
2	0	S^{m0}_{0}							
	1	▨	▨	S^{m0}_{1}		▨	▨	S^{m0}_{2}	
	2	▨	▨	▨	▨	▨	S^{m1}_{3}		▨
3	0	S^{m0}_{0}							
	1	S^{m0}_{2}	S^{m0}_{2}	▨	▨	▨	▨	S^{m0}_{1}	S^{m0}_{1}
	2	▨	S^{m1}_{3}	S^{m1}_{3}	▨	▨	▨	▨	▨
4	0	S^{m0}_{0}							
	1	S^{m0}_{1}		▨	▨	S^{m0}_{2}		▨	▨
	2	▨	▨	▨	S^{m1}_{3}		▨	▨	▨
5	0	S^{m0}_{0}							
	1	S^{m1}_{3}	▨	S^{m0}_{2}		S^{m0}_{1}		▨	S^{m1}_{3}
	2	▨	▨	▨	▨	▨	▨	▨	▨
6	0	S^{m0}_{0}							
	1	▨	▨	S^{m0}_{1}		▨	▨	S^{m0}_{2}	
	2	▨	▨	▨	▨	▨	S^{m1}_{3}		▨
7	0	S^{m0}_{0}							
	1	S^{m0}_{2}	S^{m0}_{2}	▨	▨	▨	▨	S^{m0}_{1}	S^{m0}_{1}
	2	▨	S^{m1}_{3}	S^{m1}_{3}	▨	▨	▨	▨	▨

Fig. 13 Retrievals scheduling map of Figure 12.

admission control algorithm is to perform the scheduling computation. Take the first segment as an example; when the system schedules the retrieval for segment R_0^1 at service round i, it is found that $DISK(R_0^3)$ of video m_0, $DISK(R_0^1)$ of video m_1, and $DISK(R_1^1)$ of video m_0 overlap on disk 4. Therefore, a new session, session 2, is created at service round i and the corresponding slot in session 2 of service round i is marked with $S_3^{m_1}$. The result of the map for the first P_0 segments is shown in Figure 13. Note that the admission control algorithm can be translated to searching for *available slots* in the scheduling map and since the scheduling map is small, we can make admission control decisions very quickly.

4.3 Load Balancing Property

In addition to providing the normal playback and VCR playback capabilities, our VOD system also has the load balancing property as defined in the previous section. Before we present the formal proof that the proposed VOD system is load balanced, we need the following results.

Lemma 1 *In the proposed VOD storage system, the number of data blocks per segment retrieved for display resolution r, is an integer multiple of the number of data blocks per segment retrieved for display resolution r', where $0 \le r' \le r < n$.*

Proof: Suppose that there exist l consecutive display resolutions between r and r' such that $r' < r_1 < r_2 < \cdots < r_l < r$. Consider the case that $r_l = r - 1$ and $r' = r_1 - 1$. By (A2) in Section 3.2, the following relationship holds for any two consecutive display resolutions:

$$\frac{\sum_{i=0}^{k} b_i}{\sum_{i=0}^{k-1} b_i} = C_{k-1}$$

where C_{k-1} is an integer. Therefore, for any two consecutive display resolutions between r and r', we have the following relationships:

$$\frac{\sum_{i=0}^{r} b_i}{\sum_{i=0}^{r_l} b_i} = C_{r_l}, \frac{\sum_{i=0}^{r_l} b_i}{\sum_{i=0}^{r_{l-1}} b_i} = C_{r_{l-1}},$$
$$\frac{\sum_{i=0}^{r_{l-1}} b_i}{\sum_{i=0}^{r_{l-2}} b_i} = C_{r_{l-2}}, \ldots, \frac{\sum_{i=0}^{r_1} b_i}{\sum_{i=0}^{r'} b_i} = C_{r'}$$

where C_{r_l}, $C_{r_{l-1}}$, $\ldots$, $C_{r'}$ are integers. By substitution, we have:

$$\frac{\sum_{i=0}^{r} b_i}{\sum_{i=0}^{r'} b_i} = C_{r_l} \cdot C_{r_{l-1}} \ldots C_{r'}$$

where $C_{r_l} \cdot C_{r_{l-1}} \ldots C_{r'}$ is an integer. Therefore, $\sum_{i=0}^{r} b_i$ is an integer multiple of $\sum_{i=0}^{r'} b_i$, for $0 \le r' \le r < n$. ∎

The following lemma states the load balancing property of our proposed VOD storage server under the normal display mode.

```
Algorithm   Admission Control Algorithm
Procedure Admit_User($k+1, stream, res, seg\_start$)

/* $k+1$ : the to-be admitted $k+1^{st} streams$ */
/* $stream$ : the stream, $S_i^{m_x}$, that associate with video $m_x$ */
/* $res$ : the normal display resolution to which the client subscribed */
/* $seg\_start$ : the starting segment number where the client issued
        the playback command */

Begin

1   For $i \leftarrow 0$ to $P_0 - 1$ Do /* test whether the system can admit a new client */
2     For all $j \in DISK(R^{res}_{seg\_start+i})$ Do
3       If ( $\sum_{i=0}^{k} T^{max}_{j,l,i}(\mathcal{U}) > \frac{1}{PR}$ ) or ( $\sum_{i=0}^{k} PR \cdot \mathcal{U} > B_j$ )
4         Return false
5       Endif
6     Endfor
7   Endfor

8   For $i \leftarrow 0$ to $P_r - 1$ Do /* Update the scheduling map for the newly admitted client */
9     For all session $l \in$ round $i$ Do
11      If $\neg\exists$ a slot for $DISK(R^{res}_{seg\_start+i})$
12        create a new session
13      Endif
14      mark the slot of disk $j$ at session $l$ of round $i$ with $stream$
15    Endfor
16  Endfor
17  Return true

End Admit_User
```

Fig. 14 Admission control algorithm.

Lemma 2 *Let the normal display resolution be r, where* $0 \leq r < n$, *then the amount of data retrieved from every disk is the same over* P_r *consecutive service rounds, where* $P_r = \frac{d}{\sum_{i=0}^{r} b_i}$.

Proof: If a user has subscribed to the display rate $r = n-1$ (e.g., the highest resolution display rate), then it is clear that the VOD storage server will retrieve one disk block for every d disks in the system in one service round. For the case where $0 \leq r < n-1$, we observe that not all d disks will be involved in the data retrieval in one service round. However, the amount of data transfered by each of the d disks will be the same after P_r consecutive segments are retrieved. By Lemma 1, we have:

$$\frac{\sum_{i=0}^{n-1} b_i}{\sum_{i=0}^{r} b_i} = \frac{d}{\sum_{i=0}^{r} b_i} = P_r$$

where P_r is an integer. This implies that given the disk set $\mathcal{D} = \{0, 1, \ldots, d-2, d-1\}$, it is possible for us to partition the disk set into P_r disjoint partitions with each partition having $\sum_{i=0}^{r} b_i$ disks. Lines 3 to 12 in the video placement algorithm of Figure 9 ensure that these P_r consecutive segments are non-overlapping and that they span the whole disk set $\mathcal{D}$. If a user subscribes to a video display resolution r from segment j at service round k, then P_r disjoint segments, starting from segment j to $j + P_r - 1$ are retrieved and they satisfy the following relationship:

$$DISK(R^r_j) \cap DISK(R^r_{j+1}) \cap \ldots \cap DISK(R^r_{j+P_r-2}) \cap DISK(R^r_{j+P_r-1}) = \emptyset \quad (10)$$

$$DISK(R^r_j) \cup DISK(R^r_{j+1}) \cup \ldots \cup DISK(R^r_{j+P_r-2}) \cup DISK(R^r_{j+P_r-1}) = \{0, 1, 2, \ldots, d-1\} \quad (11)$$

Equation (10) shows that these P_r consecutive segments are disjoint and therefore, data blocks of display resolution r are non-overlapping in these P_r consecutive segments. Equation (11) ensures that all d disks will be involved in I/O retrieval after P_r consecutive segment retrievals. Note that the *base template* construction (line 3 to line 12 in Figure 9) of our proposed data block assignment algorithm is based on the principles of Equations (10) and (11). According to our disk scheduling policy, one segment of a video is retrieved during each service round. As a result, the VOD storage server retrieves one data block from all d disks over P_r consecutive service rounds. ■

The following lemma states the load balancing property of our proposed VOD storage server under the VCR display mode.

Lemma 3 *If a user has subscribed to a display resolution r as the normal display resolution admission to the VOD system, then the amount of data transfered by each disk during the VCR display period using display resolution r', where $0 \leq r' < r \leq n-1$, is the same over P_r consecutive service rounds, where $P_r = \frac{d}{\sum_{i=0}^{r} b_i}$.*

Proof: First, note that the disk I/O schedule for a video stream has already been allocated upon the admission of a new client; this allocation also guarantees the I/O bandwidth for any VCR request. In other words, the data blocks being retrieved for VCR display are under the same disk I/O bandwidth allocation as the normal display retrieval, but instead of retrieving the data blocks corresponding to the normal display resolution r, the VOD system retrieves the data blocks corresponding to the VCR display resolution r'.

Based on the template construction in Figure 9, lines 13-19, the base template of display resolution r is repeated every P_r segments. Therefore, the disk assignment of data blocks for resolution r of the j^{th} segment is the same as the disk assignment of data blocks for $(j + l \cdot P_r)^{th}$ segment, where $l = 1, 2, \ldots$. Let $\mathcal{C}_r = \sum_{i=0}^{r} b_i / \sum_{i=0}^{r'}$. Based on Lemma 1, we know that the number of data blocks per segment for resolution r is an integer multiple of the number of blocks per segment for resolution r'. This implies that it is possible to partition the disk set for display resolution r into C_r disjoint sets, with each set having $\sum_{i=0}^{r'} b_i$ disks for storing the segment of display resolution r'. Therefore, we can create a disk assignment set for the data blocks at resolution r of segment j to be equal to the union of the disk assignment for data blocks at resolution r' of segments $j+i \cdot P_r$, where $0 \leq i \leq \mathcal{C}_r - 1$. Mathematically, we can express this statement as:

$$DISK(R_j^{r'}) \cap DISK(R_{j+P_r}^{r'}) \cap \ldots \cap DISK(R_{j+(C_r-1)\cdot P_r}^{r'}) = \emptyset \quad (12)$$

and

$$DISK(R_j^{r'}) \cup DISK(R_{j+P_r}^{r'}) \cup \ldots \cup DISK(R_{j+(C_r-1)\cdot P_r}^{r'}) = DISK(R_j^{r}) \quad (13)$$

The above equations imply that retrieval of video blocks for resolution r' in VCR mode, which is operated under the disk scheduling of the normal display mode at resolution r, are non-overlapping in P_r consecutive segments. Since at most one video block is being transfered from each disk during the VCR display mode of resolution r' and since C_r segments are retrieved in one service round, the load on every disk in the system is balanced over P_r consecutive service rounds. ■

Theorem 1 *The proposed VOD system is load balanced over $\mathcal{S}$ consecutive service rounds, where $\mathcal{S}$ is an integer multiple of P_r and r is the normal display resolution for a client.*

Proof: Lemma 2 states that the load on every disk is balanced over P_r consecutive service round retrievals under the normal display mode, and Lemma 3 states that the load on every disk is also balanced over P_r consecutive round retrievals under the VCR display mode. Therefore, our proposed VOD system has the load balancing property. ■

5 Buffer Space Management

In this section, we present the buffer organization of our proposed VOD storage system. We also quantify the buffer requirements for each admitted viewer as well as the associated start-up latency for VCR display.

5.1 Buffer Space Organization

In general, we use memory buffers in the VOD system to hold the data retrieved from the disk sub-system before it is transmitted to the viewing clients. In our proposed VOD system, we use a *dual buffering* approach in which there are two distinct sets of buffers which are used in an alternating manner. For example, during a service round of an admitted client, the data retrieved from the disk sub-system is stored in one set of buffers. During the following service round, another set of data is retrieved and stored in the second set of buffers. At the same time, the data that was stored in the first set of buffers is transmitted to the client. The cycle repeats such that data retrieved in one service round is stored in one set of buffers while the data retrieved in the previous service round is transmitted from the other set of buffers. There are several reasons for choosing the dual buffering approach; for instance, it gives more flexibility to the storage sub-system for retrieving data from different sessions within the same service round so as to minimize the disk overhead (e.g., total seek overhead for all data retrieved within the same service round). Another reason for using the dual buffering approach is to allow the viewer to have a smooth playback of video without experiencing data starvation or data overflow.

5.2 Buffer Space Requirements

Once a viewing client has been admitted to the system, the VOD storage server has to reserve sufficient buffer space for that viewing client. The amount of buffer space allocation depends on the normal display resolution r that the viewer chooses as well as the display resolution r' during VCR service. Therefore, in order to quantify the buffer space requirements of a viewing user, we have to consider two different display modes.

Assuming that the viewing client selects r as the normal display resolution. The VOD storage system will retrieve data in the set $\mathcal{R}_i^r$ at the i^{th} service round. Therefore, the amount of buffer space (in units of disk blocks) that is required for normal display, which we denote by B_n^r, is:

$$B_n^r = 2\sum_{i=0}^{r} b_i \tag{14}$$

Note that the 2 is due to the fact that the system uses a dual buffering scheme. To illustrate, if the viewing client chooses $r = 2$ for the video file illustrated in Figure 3, then the VOD storage server needs to reserve 4 disk blocks worth of buffer space for that client.

The analysis of the buffer space requirements during the VCR mode is more complicated. To illustrate, consider the video file in Figure 3. Assume that the viewer selects $r = 2$ and $r' = 1$ for his normal and VCR display resolutions, respectively. Since the disk I/O schedule is allocated based on the normal display, the retrieval of data blocks for the VCR display under a single service round may not belong to consecutive video segments. For example, if the VCR function is issued starting at service round i for segment 0, then the data blocks retrieved for VCR display are as follows:

Data blocks retrieved for service round i:
$\{ r^0_{0,0}, r^1_{0,0}, r^0_{2,0}, r^1_{2,0} \}$
Data blocks retrieved for service round $i+1$:
$\{ r^0_{1,0}, r^1_{1,0}, r^0_{3,0}, r^1_{3,0} \}$
Data blocks retrieved for service round $i+2$:
$\{ r^0_{4,0}, r^1_{4,0}, r^0_{6,0}, r^1_{6,0} \}$
Data blocks retrieved for service round $i+3$:
$\{ r^0_{5,0}, r^1_{5,0}, r^0_{7,0}, r^1_{7,0} \}$

This indicates that to deliver the VCR display service in the above example, the VOD storage server has to first buffer the retrieved data, and then start the transmission to the viewer no earlier than the beginning of round $i+2$.

In general, if the viewer wants to switch from normal display to VCR display, then the viewer has to experience a *VCR start-up latency*, denoted by T^r_{vcr}, which is defined as the number service rounds between the retrieval of the first set of VCR data blocks and the transmission of the first group of VCR data blocks to the viewer, given that the normal display resolution is r, $0 \leq r < n$. In the example illustrated above, the VCR start-up latency is 2 service rounds[7]. Once the VCR start-up latency is determined, we can then quantify the required buffer space allocation under the VCR display mode. The following theorem states that the VCR start-up latency is only a function of the normal display resolution r.

Theorem 2 *Given that the normal display resolution of a viewing client is r, the VCR start-up latency, T^r_{vcr}, is equal to $d/\sum_{i=0}^{r} b_i$, where d is the number of parallel disks in the VOD system.*

Proof: Note that $\sum_{i=0}^{r} b_i$ is equal to the number of disk blocks that constitute a segment of display resolution r. Therefore, if r is the normal display resolution and the system has to be able to support any VCR display resolution r', where $0 \leq r' < r < n$, the to achieve a VCR speed of $VCR\_Speed(r, r')$, $\sum_{i=0}^{r} b_i / \sum_{i=1}^{r'} b_i$ consecutive segments have to be delivered to the viewing client in a single service round. The modularity property of disk blocks (assumption A2 in Section 3.2) implies that the VOD system has to retrieve d disk blocks before delivering any data blocks to the viewing clients. Thus, based on the normal display schedule, it takes $d/\sum_{i=0}^{r} b_i$ service rounds for the system to retrieved d data blocks. ∎

Using the example in Figure 3, if the viewing client selects $r = 3$, then $T^r_{vcr} = 1$, if $r = 2$, then $T^r_{vcr} = 2$, and if $r = 1$, then $T^r_{vcr} = 4$. If the duration of the service round is 0.5 seconds, then T^r_{vcr} is between 1 and 2 seconds. This result is also interesting in the sense that if a viewer subscribes to a higher display resolution (or a higher QoS), then the viewer will experience a shorter VCR start-up delay, i.e., better QoS.

Theorem 3 *Given that the normal display resolution of a viewing client is r, the amount of buffer space (in units of disk blocks) that is required for VCR display, denoted by B^r_{vcr}, is d.*

Proof: Based on Theorem 2, the system needs to wait for $d/\sum_{i=0}^{r} b_i$ service rounds before data delivery can begin. In each service round, a maximum of $\sum_{i=0}^{r} b_i$ data blocks are retrieved by the VOD storage server. Therefore, the system needs to allocate $B^r_{vcr} = d$ disk blocks of buffer space for VCR display. ∎

Corollary 1 *Given that the normal display resolution of a viewing client is r, the amount of buffer space B (in units of disk blocks) that needs to be allocated for the entire viewing duration is:*

$$B = \max\{2\sum_{i=0}^{r} b_i, d\}$$

Proof: The result is a direct application of Equation (14) and Theorem 3. ∎

[7] Note that the duration of a service round is fixed based on the maximum number of clients that can be admitted and the level of QoS that the system can support, as illustrated in [5]. For example, the duration of a service round can range from 0.25 to 1.5 seconds.

6 Generalization of the Video Block Placement Algorithm

In Section 3, we discussed video data block assignment to a parallel disk system where the number of disks is equal to $d = \sum_{i=0}^{n-1} b_i$ and $n > 0$ is the number of display resolutions. Note that d is the *minimum* number of disks that is required in order to maintain the load balancing property of the VOD storage server under both the normal and the VCR display modes. In general, the number of disks in the VOD storage server, denoted by D, can be larger than $d = \sum_{i=0}^{n-1} b_i$. In this section, we describe how we can extend the video data block assignment algorithm to the case where $D > \sum_{i=0}^{n-1} b_i$.

There are many reasons for having the need to increase the number of disks in the VOD system. For example, if we want to increase the number of viewing clients that can be serviced within one service round, or if we want to store more video data (e.g., movies) in the VOD system so that the viewer may have more choices for their movie selection. Regardless of the reason, whenever we increase the number of disks in the VOD system, we need to create a new video disk block assignment template so that the load balancing property can be preserved.

One obvious choice for the values of D that can preserve the load balancing property and at the same time retain the characteristics of I/O scheduling, admission control, and buffer space requirements, as discussed in the previous sections, is:

$$D = k \sum_{i=0}^{n-1} b_i \qquad \text{for } k = 1, 2, \ldots$$

In other words, if D is an integer multiple of $d = \sum_{i=0}^{n-1} b_i$, then we can assign the first $P_0 = d/b_0$ consecutive segments to the first d disks, the next P_0 consecutive segments to the next d disks and so on in a round robin manner. All the results that we developed, for instance, for I/O scheduling, admission control, and buffer space requirements can still be applied. Then the load balancing property is maintained across all disks after $k \cdot P_r$ (where $P_r = d / \sum_{i=0}^{r} b_i$) consecutive segments are retrieved.

However, when D is not an integer multiple of $\sum_{i=0}^{n-1} b_i$, the disk block assignment algorithm is more complicated. We first illustrate how we can accomplish the video block assignment for the video file illustrated in Figure 3 for $D = 10$ and $d = 8$ and then present the general disk block assignment algorithm for any value of D.

6.1 Disk Block Mapping for $D = 10$ and $d = 8$

Consider the case where we want to perform video block assignment for the video file in Figure 3 and the number of disks in the system, D, is equal to 10. Since the minimum number of disks required is $d = \sum_{i=0}^{3} b_i = 8$, we would have two *extra disks* that do not have any video block assignments if we use the template given in Figure 3. Therefore, we need to design a new template for this case so that the VOD storage system can provide all the VCR functionalities and maintain the load balancing property. The central idea in creating a new template is to consider the template given in Figure 3 as a *fragment* of the final template. Then we can perform disk block assignment for the next fragment of the template (or for the following eight consecutive segments from segment 8 to segment 15) such that the following conditions hold:

$(C1)$: the non-overlapping property of consecutive segments *within* each fragment is preserved, (15)

$(C2)$: the non-overlapping property of consecutive segments *between* the fragments is preserved. (16)

Let us illustrate the video disk block assignment for the first two fragments (or for the first 16 consecutive segments). We assign the video blocks, starting from the highest display resolution first.

resolution $= 3$: Since the total number of data blocks for R_i^3 is 8, we can assign the data blocks in R_i^3, for $i \in [0, \ldots, 7]$, to the disk set $\{0,1,2,3,4,5,6,7\}$ and the data blocks in R_i^3, for $i \in [8, \ldots, 15]$, to the disk set $\{8,9,0,1,2,3,4,5\}$. Again, we cannot determine the assignment of the enhancement data blocks until we assign the data blocks in the lower display resolutions.

resolution $= 2$: To map the data blocks of $\mathcal{R}_i^2$, where $0 \le i \le 15$, we must ensure that if a user subscribes to a normal display resolution $r = 3$, then he should be able to issue a VCR command with a VCR speed-up of $VCR\_Speed(3,2) = 8/4 = 2$, which is two times faster than the normal display. This implies that any two consecutive segments of display resolution 2 must be non-overlapping. Note that the non-overlapping requirement should be enforced for segments within the fragment and between two consecutive fragments. For example, we can assign the data blocks of $\mathcal{R}_i^2$, for $i \in [0,1,\ldots,7]$, using the same template we described in Figure 3. Then the data blocks in $\mathcal{R}_i^2$, where $i \in [8,10,12,14]$, can be assigned to the disk set $\{8,9,0,1\}$, and the data blocks in $\mathcal{R}_i^2$, where $i \in \{9,11,13,15\}$, can be assigned to the disk set $\{2,3,4,5\}$. The mapping of $\mathcal{V}_i^2$, for $0 \le i \le 15$, is illustrated in Figure 15. It is clear that the $VCR\_Speed(3,2) = 2$ can be made available within the new fragment (from segments 8 to 15) and across the two consecutive fragments (segments 0 to 15). Once the mapping of the data blocks in $\mathcal{V}_i^2$ is known, the data block assignment for the enhancement blocks in $\mathcal{V}_i^3$ can be determined.

resolution $= 1$: To map the data blocks in $\mathcal{R}_i^1$, we have to consider two cases:

Seg. / Disk	0	1	2	3	4	5	6	7	8	9
0	R^2_0				$r^3_{0,0}$	$r^3_{0,1}$	$r^3_{0,2}$	$r^3_{0,3}$		
1	$r^3_{1,0}$	$r^3_{1,1}$	$r^3_{1,2}$	$r^3_{1,3}$	R^2_1					
2	R^2_2				$r^3_{2,0}$	$r^3_{2,1}$	$r^3_{2,2}$	$r^3_{2,3}$		
3	$r^3_{3,0}$	$r^3_{3,1}$	$r^3_{3,2}$	$r^3_{3,3}$	R^2_3					
4	R^2_4				$r^3_{4,0}$	$r^3_{4,1}$	$r^3_{4,2}$	$r^3_{4,3}$		
5	$r^3_{5,0}$	$r^3_{5,1}$	$r^3_{5,2}$	$r^3_{5,3}$	R^2_5					
6	R^2_6				$r^3_{6,0}$	$r^3_{6,1}$	$r^3_{6,2}$	$r^3_{6,3}$		
7	$r^3_{7,0}$	$r^3_{7,1}$	$r^3_{7,2}$	$r^3_{7,3}$	R^2_7					
8			$r^3_{8,2}$	$r^3_{8,3}$	$r^3_{8,0}$	$r^3_{8,1}$			R^2_8	
9	$r^3_{9,0}$	$r^3_{9,1}$	R^2_9						$r^3_{9,2}$	$r^3_{9,3}$
10			$r^3_{10,2}$	$r^3_{10,3}$	$r^3_{10,0}$	$r^3_{10,1}$			R^2_{10}	
11	$r^3_{11,0}$	$r^3_{11,1}$	R^2_{11}						$r^3_{11,2}$	$r^3_{11,3}$
12			$r^3_{12,2}$	$r^3_{12,3}$	$r^3_{12,0}$	$r^3_{12,1}$			R^2_{12}	
13	$r^3_{13,0}$	$r^3_{13,1}$	R^2_{13}						$r^3_{13,2}$	$r^3_{13,3}$
14			$r^3_{14,2}$	$r^3_{14,3}$	$r^3_{14,0}$	$r^3_{14,1}$			R^2_{14}	
15	$r^3_{15,0}$	$r^3_{15,1}$	R^2_{15}						$r^3_{15,2}$	$r^3_{15,3}$

Fig. 15 Fragment assignment for display resolution $r = 2$.

Seg. / Disk	0	1	2	3	4	5	6	7	8	9
0	R^1_0		$r^2_{0,0}$	$r^2_{0,1}$	$r^3_{0,0}$	$r^3_{0,1}$	$r^3_{0,2}$	$r^3_{0,3}$		
1	$r^3_{1,0}$	$r^3_{1,1}$	$r^3_{1,2}$	$r^3_{1,3}$	R^1_1		$r^2_{1,0}$	$r^2_{1,1}$		
2	$r^2_{2,0}$	$r^2_{2,1}$	R^1_2		$r^3_{2,0}$	$r^3_{2,1}$	$r^3_{2,2}$	$r^3_{2,3}$		
3	$r^3_{3,0}$	$r^3_{3,1}$	$r^3_{3,2}$	$r^3_{3,3}$	$r^2_{3,0}$	$r^2_{3,1}$	R^1_3			
4	R^1_4		$r^2_{4,0}$	$r^2_{4,1}$	$r^3_{4,0}$	$r^3_{4,1}$	$r^3_{4,2}$	$r^3_{4,3}$		
5	$r^3_{5,0}$	$r^3_{5,1}$	$r^3_{5,2}$	$r^3_{5,3}$	R^1_5		$r^2_{5,0}$	$r^2_{5,1}$		
6	$r^2_{6,0}$	$r^2_{6,1}$	R^1_6		$r^3_{6,0}$	$r^3_{6,1}$	$r^3_{6,2}$	$r^3_{6,3}$		
7	$r^3_{7,0}$	$r^3_{7,1}$	$r^3_{7,2}$	$r^3_{7,3}$	$r^2_{7,0}$	$r^2_{7,1}$	R^1_7			
8	R^1_8		$r^3_{8,2}$	$r^3_{8,3}$	$r^3_{8,0}$	$r^3_{8,1}$			$r^2_{8,0}$	$r^2_{8,1}$
9	$r^3_{9,0}$	$r^3_{9,1}$	$r^2_{9,0}$	$r^2_{9,1}$	R^1_9				$r^3_{9,2}$	$r^3_{9,3}$
10	$r^2_{10,0}$	$r^2_{10,1}$	$r^3_{10,2}$	$r^3_{10,3}$	$r^3_{10,0}$	$r^3_{10,1}$			R^1_{10}	
11	$r^3_{11,0}$	$r^3_{11,1}$	R^1_{11}		$r^2_{11,0}$	$r^2_{11,1}$			$r^3_{11,2}$	$r^3_{11,3}$
12	R^1_{12}		$r^3_{12,2}$	$r^3_{12,3}$	$r^3_{12,0}$	$r^3_{12,1}$			$r^2_{12,0}$	$r^2_{12,1}$
13	$r^3_{13,0}$	$r^3_{13,1}$	$r^2_{13,0}$	$r^2_{13,1}$	R^1_{13}				$r^3_{13,2}$	$r^3_{13,3}$
14	$r^2_{14,0}$	$r^2_{14,1}$	$r^3_{14,2}$	$r^3_{14,3}$	$r^3_{14,0}$	$r^3_{14,1}$			R^1_{14}	
15	$r^3_{15,0}$	$r^3_{15,1}$	R^1_{15}		$r^2_{15,0}$	$r^2_{15,1}$			$r^3_{15,2}$	$r^3_{15,3}$

Fig. 16 Fragment assignment for display resolution $r = 1$.

- **case 1:** If the viewer subscribes to a normal display resolution $r = 3$, then he should be able to select a VCR speed-up of $VCR\_Speed(3, 1) = 8/2 = 4$ times the normal display speed by using $r' = 1$. This implies that any four consecutive video segments, $\mathcal{R}^1_i$, $\mathcal{R}^1_{i+1}$,$\mathcal{R}^1_{i+2}$, and $\mathcal{R}^1_{i+3}$ must be non-overlapping within and across the two consecutive fragments.
- **case 2:** If the user subscribes to a normal display resolution $r = 2$, then he should be able to select a VCR speed-up of $VCR\_Speed(2, 1) = 4/2 = 2$ times the normal display speed by using $r' = 1$. This implies that any two consecutive segments, $\mathcal{R}^1_i$ and $\mathcal{R}^1_{i+1}$, must be non-overlapping within and across the two consecutive fragments.

It is clear that if case 1 holds, then case 2 holds as well. Therefore, one of the possible mappings of data blocks in $\mathcal{R}^1_i$ is to use the disk mapping described in Figure 3 for $i \in [0, 1, \ldots, 7]$ and for the remaining $i \in [8, 9, \ldots, 15]$ to use:
$DISK(\mathcal{R}^1_8) = DISK(\mathcal{R}^1_{12}) = \{0, 1\}$; $DISK(\mathcal{R}^1_9) = DISK(\mathcal{R}^1_{13}) = \{4, 5\}$;
$DISK(\mathcal{R}^1_{10}) = DISK(\mathcal{R}^1_{14}) = \{8, 9\}$; $DISK(\mathcal{R}^1_{11}) = DISK(\mathcal{R}^1_{15}) = \{2, 3\}$.

It is important to note that any four consecutive segments of display resolution 1 are non-overlapping within and across the consecutive disk mapping fragments. Once this assignment is made, the data block assignments of enhancement blocks in $\mathcal{R}^1_i$ can be determined. This assignment is illustrated in Figure 16.

resolution $= 0$: To map the data blocks in $\mathcal{R}^0_i$, three cases must be considered:

- **case 1:** $r = 3$ and $r' = 0$. To support $VCR\_Speed(3, 0) = 8$, any eight consecutive segments of $\mathcal{R}^0_i$ must be non-overlapping.
- **case 2:** $r = 2$ and $r' = 0$. To support $VCR\_Speed(2, 0) = 4$, any four consecutive segments of $\mathcal{R}^0_i$ must be non-overlapping.
- **case 3:** $r = 1$ and $r' = 0$. To support $VCR\_Speed(1, 0) = 2$, any two consecutive segments of $\mathcal{R}^0_i$ must be non-overlapping.

If the first case is satisfied, then the other two cases will also hold. One possible approach is to use the assignment given in Figure 3 for $i \in [0, 1, \ldots, 7]$ and for $i \in [8, \ldots, 15]$, to use $DISK(\mathcal{R}^0_8) = \{0\}$, $DISK(\mathcal{R}^0_9) = \{4\}$, $DISK(\mathcal{R}^0_{10}) = \{8\}$, $DISK(\mathcal{R}^0_{11}) = \{2\}$,

Seg. / Disk	0	1	2	3	4	5	6	7	8	9
0	R^0_0	$r^1_{0,0}$	$r^2_{0,0}$	$r^2_{0,1}$	$r^3_{0,0}$	$r^3_{0,1}$	$r^3_{0,2}$	$r^3_{0,3}$		
1	$r^3_{1,0}$	$r^3_{1,1}$	$r^3_{1,2}$	$r^3_{1,3}$	R^0_1	$r^1_{1,0}$	$r^2_{1,0}$	$r^2_{1,1}$		
2	$r^2_{2,0}$	$r^2_{2,1}$	R^0_2	$r^1_{2,0}$	$r^3_{2,0}$	$r^3_{2,1}$	$r^3_{2,2}$	$r^3_{2,3}$		
3	$r^3_{3,0}$	$r^3_{3,1}$	$r^3_{3,2}$	$r^3_{3,3}$	$r^2_{3,0}$	$r^2_{3,1}$	R^0_3	$r^1_{3,0}$		
4	$r^1_{4,0}$	R^0_4	$r^2_{4,0}$	$r^2_{4,1}$	$r^3_{4,0}$	$r^3_{4,1}$	$r^3_{4,2}$	$r^3_{4,3}$		
5	$r^3_{5,0}$	$r^3_{5,1}$	$r^3_{5,2}$	$r^3_{5,3}$	$r^1_{5,0}$	R^0_5	$r^2_{5,0}$	$r^2_{5,1}$		
6	$r^2_{6,0}$	$r^2_{6,1}$	$r^1_{6,0}$	R^0_6	$r^3_{6,0}$	$r^3_{6,1}$	$r^3_{6,2}$	$r^3_{6,3}$		
7	$r^3_{7,0}$	$r^3_{7,1}$	$r^3_{7,2}$	$r^3_{7,3}$	$r^2_{7,0}$	$r^2_{7,1}$	$r^1_{7,0}$	R^0_7		
8	R^0_8	$r^1_{8,0}$	$r^3_{8,2}$	$r^3_{8,3}$	$r^3_{8,0}$	$r^3_{8,1}$			$r^2_{8,0}$	$r^2_{8,1}$
9	$r^3_{9,0}$	$r^3_{9,1}$	$r^2_{9,0}$	$r^2_{9,1}$	R^0_9	$r^1_{9,0}$			$r^3_{9,2}$	$r^3_{9,3}$
10	$r^2_{10,0}$	$r^2_{10,1}$	$r^3_{10,2}$	$r^3_{10,3}$	$r^3_{10,0}$	$r^3_{10,1}$			R^0_{10}	$r^1_{10,0}$
11	$r^3_{11,0}$	$r^3_{11,1}$	R^0_{11}	$r^1_{11,0}$	$r^2_{11,0}$	$r^2_{11,1}$			$r^3_{11,2}$	$r^3_{11,3}$
12	$r^1_{12,0}$	R^0_{12}	$r^3_{12,2}$	$r^3_{12,3}$	$r^3_{12,0}$	$r^3_{12,1}$			$r^2_{12,0}$	$r^2_{12,1}$
13	$r^3_{13,0}$	$r^3_{13,1}$	$r^2_{13,0}$	$r^2_{13,1}$	$r^1_{13,0}$	R^0_{13}			$r^3_{13,2}$	$r^3_{13,3}$
14	$r^2_{14,0}$	$r^2_{14,1}$	$r^3_{14,2}$	$r^3_{14,3}$	$r^3_{14,0}$	$r^3_{14,1}$			$r^1_{14,0}$	R^0_{14}
15	$r^3_{15,0}$	$r^3_{15,1}$	$r^1_{15,0}$	R^0_{15}	$r^2_{15,0}$	$r^2_{15,1}$			$r^3_{15,2}$	$r^3_{15,3}$

Fig. 17 Fragment assignment for display resolution $r = 0$.

$DISK(\mathcal{R}^0_{12}) = \{1\}$, $DISK(\mathcal{R}^0_{13}) = \{0\}$, $DISK(\mathcal{R}^0_{14}) = \{9\}$, and $DISK(\mathcal{R}^0_{15}) = \{3\}$. Again, any eight consecutive segments of $\mathcal{R}^0_i$ are non-overlapping within and across two consecutive fragments. This assignment is illustrated in Figure 17.

The above procedure illustrates how we can construct two consecutive fragments for the first 16 segments of the video file. In order to achieve the load balancing property, we have to repeat this process and create the third fragment (for segments 16 to 23), the fourth fragment (for segments 24 to 31) and the fifth fragment (for segments 32 to 40). The final template for the video file in Figure 3 with $D = 10$ is illustrated in Figure 18. For the example we have just illustrated, it can be observed that the load balancing property is maintained after retrieving $P_r \cdot \frac{D}{D-d}$ consecutive segments, assuming that the viewer subscribes to normal display resolution r, $0 \le r < n$.

6.2 The Video Block Placement Algorithm

The example in Figures 15 to 17 illustrates the construction of the first two fragments, one for the first 8 consecutive segments and the other for the next eight consecutive segments, of the video file under the condition that $D = 10$. Let us now present the general video data block assignment algorithm.

The general data block assignment algorithm is composed of a number of *stages*, which depends on the value of D, the number of disks in the system, as well as the minimum number of required disks $d = \sum_{i=0}^{n-1} b_k$. Each stage consists of two steps: 1) map the fragment created in the previous stage to a new disk set and 2) modify the current fragment based on the new disk mapping. Upon completion of these two steps, a stage is said to be completed and the disk assignment proceeds to the next stage by repeating these two steps. In what follows, we describe in detail the two steps involved in the generalized data block assignment algorithm:

step 1: **Mapping the fragment to a new disk set**
The purpose of this step is to maintain the load balancing property across all D disks. In Section 6.1, we have illustrated the data block assignment that maps the fragment with $d = 8$ disks to the disk system with $D = 10$ disks. This implies that there are $d_{diff} = D - d = 2$ disks that did not received any data block assignments at each stage. In order to maintain the load balancing property, the fragment is re-mapped to a different disk set at each stage. In general, the un-mapped disks in the previous stage are mapped to the disk set of the fragment at the current stage. Figure 16 is used to illustrate this concept. In this figure, the disk set $\{8, 9\}$ is not mapped to the fragment in stage 1. In stage 2, the fragment will include the disk set $\{8, 9\}$ where the resulting disk set is $\{8, 9, 0, 1, 2, 3, 4, 5\}$, as shown in Figure 19 (a). In general, we map every data block, $r^j_{i,k}$ in the fragment of the previous stage to the new fragment of the current stage as:

$$disk(r^j_{i,k}) = (disk(r^j_{i,k}) + \lfloor \frac{i}{P_0} \rfloor \times d) \bmod D \quad (17)$$

For instance, data block $r^0_{8,0}$, which is assigned to disk $disk(r^0_{8,0}) = 0$ using the fragment created at stage 1, is mapped to $(disk(r^0_{8,0}) + \lfloor \frac{8}{8} \rfloor \times 8) \bmod 10 = 8$ in the fragment at stage 2.

step 2: **Modification of the fragment**
The reason for the modification of the fragment created in step 1 is to guarantee that the VCR functionalities are feasible in the new mapping. In general, we can map data block $r^j_{i,k}$, which has been re-mapped to the new disk set in step 1, to a new disk number as follows:

$$disk(r^j_{i,k}) = (disk(r^j_{i,k}) + (\lfloor \frac{i}{P_0} \rfloor \times$$

Seg. / Disk	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	$r^{0}_{0,0}$	$r^{1}_{0,0}$	$r^{2}_{0,0}$	$r^{2}_{0,1}$	$r^{3}_{0,0}$	$r^{3}_{0,1}$	$r^{3}_{0,2}$	$r^{3}_{0,3}$		
1	$r^{3}_{1,0}$	$r^{3}_{1,1}$	$r^{3}_{1,2}$	$r^{3}_{1,3}$	$r^{0}_{1,0}$	$r^{1}_{1,0}$	$r^{2}_{1,0}$	$r^{2}_{1,1}$		
2	$r^{2}_{2,0}$	$r^{2}_{2,1}$	$r^{0}_{2,0}$	$r^{1}_{2,0}$	$r^{3}_{2,0}$	$r^{3}_{2,1}$	$r^{3}_{2,2}$	$r^{3}_{2,3}$		
3	$r^{3}_{3,0}$	$r^{3}_{3,1}$	$r^{3}_{3,2}$	$r^{3}_{3,3}$	$r^{2}_{3,0}$	$r^{2}_{3,1}$	$r^{0}_{3,0}$	$r^{1}_{3,0}$		
4	$r^{1}_{4,0}$	$r^{0}_{4,0}$	$r^{2}_{4,0}$	$r^{2}_{4,1}$	$r^{3}_{4,0}$	$r^{3}_{4,1}$	$r^{3}_{4,2}$	$r^{3}_{4,3}$		
5	$r^{3}_{5,0}$	$r^{3}_{5,1}$	$r^{3}_{5,2}$	$r^{3}_{5,3}$	$r^{1}_{5,0}$	$r^{0}_{5,0}$	$r^{2}_{5,0}$	$r^{2}_{5,1}$		
6	$r^{2}_{6,0}$	$r^{2}_{6,1}$	$r^{1}_{6,0}$	$r^{0}_{6,0}$	$r^{3}_{6,0}$	$r^{3}_{6,1}$	$r^{3}_{6,2}$	$r^{3}_{6,3}$		
7	$r^{3}_{7,0}$	$r^{3}_{7,1}$	$r^{3}_{7,2}$	$r^{3}_{7,3}$	$r^{2}_{7,0}$	$r^{2}_{7,1}$	$r^{1}_{7,0}$	$r^{0}_{7,0}$		
8	$r^{0}_{8,0}$	$r^{1}_{8,0}$	$r^{3}_{8,2}$	$r^{3}_{8,3}$	$r^{3}_{8,0}$	$r^{3}_{8,1}$			$r^{2}_{8,0}$	$r^{2}_{8,1}$
9	$r^{3}_{9,0}$	$r^{3}_{9,1}$	$r^{2}_{9,0}$	$r^{2}_{9,1}$	$r^{0}_{9,0}$	$r^{1}_{9,0}$			$r^{3}_{9,2}$	$r^{3}_{9,3}$
10	$r^{2}_{10,0}$	$r^{2}_{10,1}$	$r^{3}_{10,2}$	$r^{3}_{10,3}$	$r^{3}_{10,0}$	$r^{3}_{10,1}$			$r^{0}_{10,0}$	$r^{1}_{10,0}$
11	$r^{3}_{11,0}$	$r^{3}_{11,1}$	$r^{0}_{11,0}$	$r^{1}_{11,0}$	$r^{2}_{11,0}$	$r^{2}_{11,1}$			$r^{3}_{11,2}$	$r^{3}_{11,3}$
12	$r^{1}_{12,0}$	$r^{0}_{12,0}$	$r^{3}_{12,2}$	$r^{3}_{12,3}$	$r^{3}_{12,0}$	$r^{3}_{12,1}$			$r^{2}_{12,0}$	$r^{2}_{12,1}$
13	$r^{3}_{13,0}$	$r^{3}_{13,1}$	$r^{2}_{13,0}$	$r^{2}_{13,1}$	$r^{1}_{13,0}$	$r^{0}_{13,0}$			$r^{3}_{13,2}$	$r^{3}_{13,3}$
14	$r^{2}_{14,0}$	$r^{2}_{14,1}$	$r^{3}_{14,2}$	$r^{3}_{14,3}$	$r^{3}_{14,0}$	$r^{3}_{14,1}$			$r^{1}_{14,0}$	$r^{0}_{14,0}$
15	$r^{3}_{15,0}$	$r^{3}_{15,1}$	$r^{1}_{15,0}$	$r^{0}_{15,0}$	$r^{2}_{15,0}$	$r^{2}_{15,1}$			$r^{3}_{15,2}$	$r^{3}_{15,3}$
16	$r^{3}_{16,0}$	$r^{3}_{16,1}$	$r^{3}_{16,2}$	$r^{3}_{16,3}$			$r^{0}_{16,0}$	$r^{1}_{16,0}$	$r^{2}_{16,0}$	$r^{2}_{16,1}$
17	$r^{0}_{17,0}$	$r^{1}_{17,0}$	$r^{2}_{17,0}$	$r^{2}_{17,1}$			$r^{3}_{17,0}$	$r^{3}_{17,1}$	$r^{3}_{17,2}$	$r^{3}_{17,3}$
18	$r^{3}_{18,0}$	$r^{3}_{18,1}$	$r^{3}_{18,2}$	$r^{3}_{18,3}$			$r^{2}_{18,0}$	$r^{2}_{18,1}$	$r^{0}_{18,0}$	$r^{1}_{18,0}$
19	$r^{2}_{19,0}$	$r^{2}_{19,1}$	$r^{0}_{19,0}$	$r^{1}_{19,0}$			$r^{3}_{19,0}$	$r^{3}_{19,1}$	$r^{3}_{19,2}$	$r^{3}_{19,3}$
20	$r^{3}_{20,0}$	$r^{3}_{20,1}$	$r^{3}_{20,2}$	$r^{3}_{20,3}$			$r^{1}_{20,0}$	$r^{0}_{20,0}$	$r^{2}_{20,0}$	$r^{2}_{20,1}$
21	$r^{1}_{21,0}$	$r^{0}_{21,0}$	$r^{2}_{21,0}$	$r^{2}_{21,1}$			$r^{3}_{21,0}$	$r^{3}_{21,1}$	$r^{3}_{21,2}$	$r^{3}_{21,3}$
22	$r^{3}_{22,0}$	$r^{3}_{22,1}$	$r^{3}_{22,2}$	$r^{3}_{22,3}$			$r^{2}_{22,0}$	$r^{2}_{22,1}$	$r^{1}_{22,0}$	$r^{0}_{22,0}$
23	$r^{2}_{23,0}$	$r^{2}_{23,1}$	$r^{1}_{23,0}$	$r^{0}_{23,0}$			$r^{3}_{23,0}$	$r^{3}_{23,1}$	$r^{3}_{23,2}$	$r^{3}_{23,3}$
24	$r^{3}_{24,0}$	$r^{3}_{24,1}$			$r^{2}_{24,0}$	$r^{2}_{24,1}$	$r^{0}_{24,0}$	$r^{1}_{24,0}$	$r^{3}_{24,2}$	$r^{3}_{24,3}$
25	$r^{0}_{25,0}$	$r^{1}_{25,0}$			$r^{3}_{25,2}$	$r^{3}_{25,3}$	$r^{3}_{25,0}$	$r^{3}_{25,1}$	$r^{2}_{25,0}$	$r^{2}_{25,1}$
26	$r^{3}_{26,0}$	$r^{3}_{26,1}$			$r^{0}_{26,0}$	$r^{1}_{26,0}$	$r^{2}_{26,0}$	$r^{2}_{26,1}$	$r^{3}_{26,2}$	$r^{3}_{26,3}$
27	$r^{2}_{27,0}$	$r^{2}_{27,1}$			$r^{3}_{27,2}$	$r^{3}_{27,3}$	$r^{3}_{27,0}$	$r^{3}_{27,1}$	$r^{0}_{27,0}$	$r^{1}_{27,0}$
28	$r^{3}_{28,0}$	$r^{3}_{28,1}$			$r^{2}_{28,0}$	$r^{2}_{28,1}$	$r^{1}_{28,0}$	$r^{0}_{28,0}$	$r^{3}_{28,2}$	$r^{3}_{28,3}$
29	$r^{1}_{29,0}$	$r^{0}_{29,0}$			$r^{3}_{29,2}$	$r^{3}_{29,3}$	$r^{3}_{29,0}$	$r^{3}_{29,1}$	$r^{2}_{29,0}$	$r^{2}_{29,1}$
30	$r^{3}_{30,0}$	$r^{3}_{30,1}$			$r^{1}_{30,0}$	$r^{0}_{30,0}$	$r^{2}_{30,0}$	$r^{2}_{30,1}$	$r^{3}_{30,2}$	$r^{3}_{30,3}$
31	$r^{2}_{31,0}$	$r^{2}_{31,1}$			$r^{3}_{31,2}$	$r^{3}_{31,3}$	$r^{3}_{31,0}$	$r^{3}_{31,1}$	$r^{1}_{31,0}$	$r^{0}_{31,0}$
32			$r^{0}_{32,0}$	$r^{1}_{32,0}$	$r^{2}_{32,0}$	$r^{2}_{32,1}$	$r^{3}_{32,0}$	$r^{3}_{32,1}$	$r^{3}_{32,2}$	$r^{3}_{32,3}$
33			$r^{3}_{33,0}$	$r^{3}_{33,1}$	$r^{3}_{33,2}$	$r^{3}_{33,3}$	$r^{0}_{33,0}$	$r^{1}_{33,0}$	$r^{2}_{33,0}$	$r^{2}_{33,1}$
34			$r^{2}_{34,0}$	$r^{2}_{34,1}$	$r^{0}_{34,0}$	$r^{1}_{34,0}$	$r^{3}_{34,0}$	$r^{3}_{34,1}$	$r^{3}_{34,2}$	$r^{3}_{34,3}$
35			$r^{3}_{35,0}$	$r^{3}_{35,1}$	$r^{3}_{35,2}$	$r^{3}_{35,3}$	$r^{2}_{35,0}$	$r^{2}_{35,1}$	$r^{0}_{35,0}$	$r^{1}_{35,0}$
36			$r^{1}_{36,0}$	$r^{0}_{36,0}$	$r^{2}_{36,0}$	$r^{2}_{36,1}$	$r^{3}_{36,0}$	$r^{3}_{36,1}$	$r^{3}_{36,2}$	$r^{3}_{36,3}$
37			$r^{3}_{37,0}$	$r^{3}_{37,1}$	$r^{3}_{37,2}$	$r^{3}_{37,3}$	$r^{1}_{37,0}$	$r^{0}_{37,0}$	$r^{2}_{37,0}$	$r^{2}_{37,1}$
38			$r^{2}_{38,0}$	$r^{2}_{38,1}$	$r^{1}_{38,0}$	$r^{0}_{38,0}$	$r^{3}_{38,0}$	$r^{3}_{38,1}$	$r^{3}_{38,2}$	$r^{3}_{38,3}$
39			$r^{3}_{39,0}$	$r^{3}_{39,1}$	$r^{3}_{39,2}$	$r^{3}_{39,3}$	$r^{2}_{39,0}$	$r^{2}_{39,1}$	$r^{1}_{39,0}$	$r^{0}_{39,0}$

Fig. 18 The final template illustrating the data block assignment for the first 40 video segments when $D = 10$.

$$(D - d)) \bmod \sum_{i=0}^{n-2} b_i) \bmod D \quad (18)$$

For instance, data block $r^0_{8,0}$ in Figure 18 is mapped to $(8 + (\lfloor \frac{8}{8} \rfloor \times 2) \bmod 4) \bmod 10 = 0$. The fragment creation algorithm is given in Figure 20. The variable *start_seg* is a system parameter that specifies the first segment number of the template to be mapped. The resulting mapping of the first 5 stages for the example in Section 6.1 is shown in Figure 18. It is important to note that for this example, the load balancing property is maintained for any display resolution r, where $0 \leq r \leq n - 1$, over the first 40 segments. The remaining segments of the video file can be assigned by using this resulting template in a modular fashion. The following two theorems show that the VOD system can provide VCR functionality service and at the same time maintain the load balancing property in the general case where $D > d$. It is important to point out that the empty space in Figure 20 does not imply that the space is not being utilized, i.e., this is a logical rather than a physical layout on the disk, used to illustrate that on a per disk basis we utilize the same amount of storage space. One can see that the empty space can be eliminated by "pushing" the data blocks, below the empty space, upwards (e.g., data block $r^0_{32,0}$ on disk 2 can be placed in the 24th row). Since the template

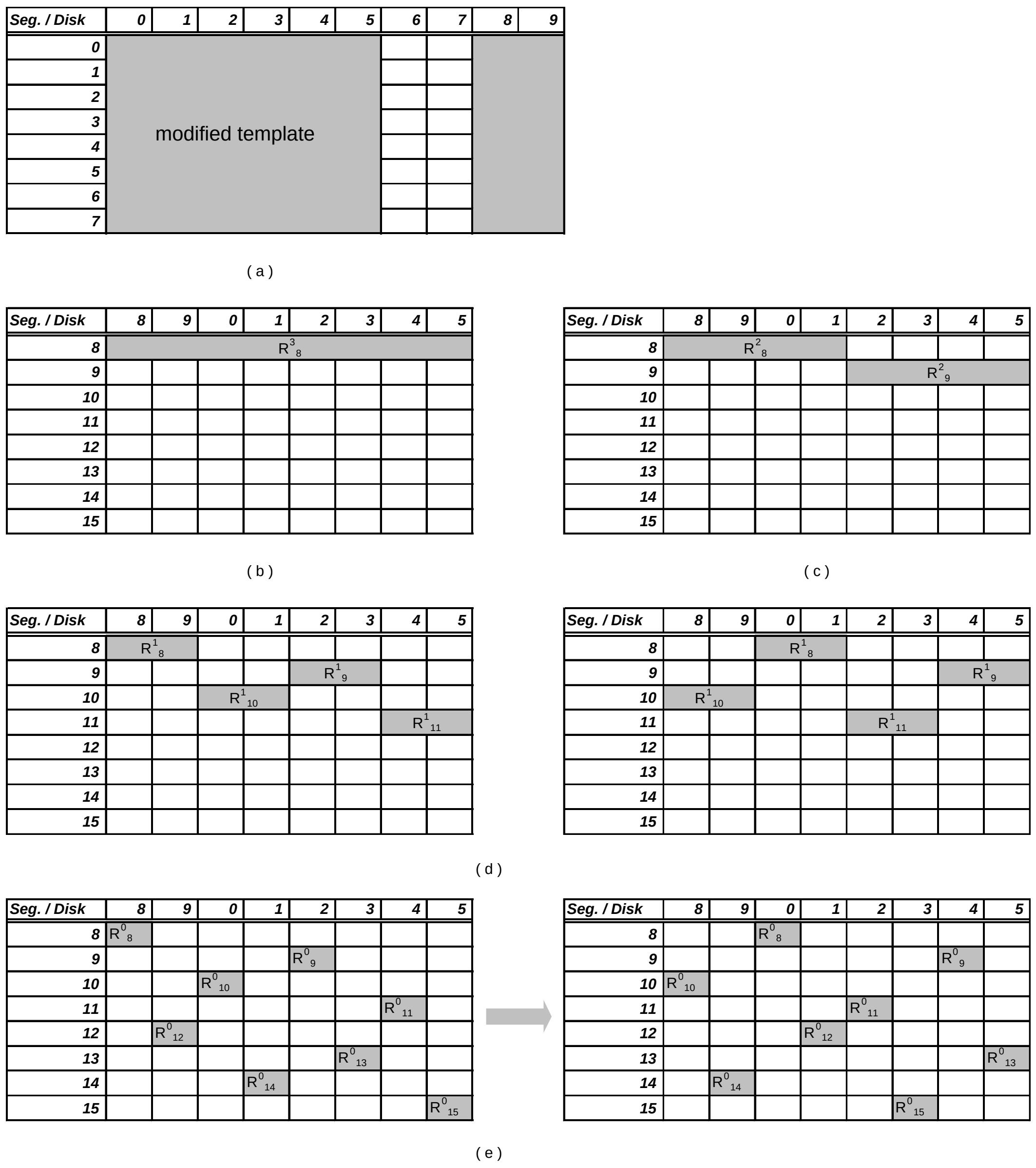

Fig. 19 The transformation principle of fragments for 4−display resolution file.

Algorithm $\mathcal{P}$ *Fragment creation for each stage*

```
Procedure NEW_FRAGMENT_CREATION ($start\_seg$)

Begin
    For $j \leftarrow n-1$ to $0$ Do
        For $i \leftarrow start\_seg$ to $start\_seg + P_0$ Do
            For $k \leftarrow 0$ to $b_j$ Do
                $disk(r^j_{i,k}) \leftarrow (disk(r^j_{i,k}) + \lfloor \frac{i}{P_0} \rfloor \times d) \bmod D$
              $disk(r^j_{i,k}) \leftarrow (disk(r^j_{i,k}) + (\lfloor \frac{i}{P_0} \rfloor \times (D-d)) \bmod \sum_{i=0}^{n-2} b_i) \bmod D$
            Endfor
        Endfor
    Endfor
End NEW_FRAGMENT_CREATION
```

Fig. 20 Fragment creation algorithm.

is small and can be stored in memory, data address translation is quick.

Theorem 4 *The VOD server can provide VCR functionality for any $VCR\_Speed(r, r')$ speed-up where r and r', $0 \leq r' < r \leq n-1$, are the normal and the VCR display resolutions, respectively.*

Proof: In order for the VOD server to provide VCR service with speed-up $VCR\_Speed(r, r')$, consecutive $P_{r'}$ video segments of display resolution r' must be non-overlapping. The non-overlapping condition must include the following cases: (1) non-overlapping within the fragment and, (2) non-overlapping between two consecutive fragments that were created in any two consecutive stages. Let T_1 and T_2 be the fragments which are created in consecutive stages (as illustrated in Figure 21). The first segment of template T_1 is l and the first segment of template T_2 is $l + P_0$. Let $\mathcal{D} = \{0, 1, \ldots, D-1\}$ be the entire disk set and D_i be the disk set for template T_i. Then we define the un-mapped disk set for template T_i as $\delta_{D_i} = \mathcal{D} - D_i$ for $i = 1, 2$. To show the capability of delivering the VCR service, we have to consider the following two cases:

case 1: within the fragment T_1 or T_2 :
The VOD server is capable of providing VCR service with $VCR\_Speed(r, r')$ speed-up within T_1 and T_2, which we proved in Lemma 3.

case 2: across the two fragments T_1 and T_2:
Suppose the user starts the VCR command at segment j, where $l + 1 \leq j \leq l + P_0 - 1$. We have to show that $P_{r'}$ consecutive segments, part of which belong to T_1 and part to T_2, must be non-overlapping. Assume that there are x segments in template T_1 and y segments in template T_2, such that $j + x = l + P_0$ and $x + y = P_{r'}$. Note that the data blocks of display resolution r' from segment j to segment $j + x - 1$ are non-overlapping (Lemma 2). Moreover, the data blocks of display resolution r' from segment $l + P_0$ to segment $l + P_0 + y - 1$ are also non-overlapping (Lemma 2). Our template creation algorithm (Figure 20, lines 4 and 5) guarantees that the data block assignment of display resolution r' at $l + P_0$ must be in the disk set $DISK(\mathcal{R}_j^{r'}) \cup \delta_{D_1}$. Since $DISK(\mathcal{R}_{l+P_0}^{r'}) \cap \delta_{D_1} \cap DISK(\mathcal{R}_j^{r'}) \cap \ldots \cap DISK(\mathcal{R}_{l+P_0-1}^{r'}) = \emptyset$, $DISK(\mathcal{R}_{l+P_0}^{r'})$ must be non-overlapping with the disk set $DISK(\mathcal{R}_j^{r'}) \cup \ldots \cup DISK(\mathcal{R}_{l+P_0-1}^{r'})$. A similar argument can be applied to segments $l+P_0+1$ to $l+P_0+y-1$. As a result, segments $\mathcal{R}_j^{r'} \ldots \mathcal{R}_{l+P_0+y-1}^{r'}$ are non-overlapping. Therefore, VCR functionality with speed-up $VCR\_Speed(r, r')$ is feasible throughout the playback of the video file. ∎

Theorem 5 *The VOD storage server achieves the load balancing property over $\frac{L.C.M.(D,D-d)}{H.C.F.(D,D-d)} \times P_r$ consecutive service rounds, where r is the normal display resolution, D and d are the total number of disks in the VOD storage server and the number of disks used for the construction of a fragment, respectively, L.C.M. is the least common multiplier function, and H.C.F. is the highest common factor function.*

Proof: By Theorem 1, we know that all disks receive requests for the same number of data blocks over P_0 consecutive service rounds if $d = D$. If we map the fragments in a round-robin fashion, the VOD server can retrieve the same number of data blocks for all disks over a number of service rounds. The **L.C.M.**$(D, D-d)$ guarantees that the un-mapped disk set, δ_{D_k}, for template T_k is evenly distributed to the entire disk set $\mathcal{D}$ in **L.C.M.**$(D, D-d)$ consecutive stages. The term **H.C.F.**$(D, D-d)$ guarantees that the number of service rounds needed to achieve the load balancing property is bounded. As a result, the data blocks for display resolution j are evenly mapped across all disks over $\frac{\mathbf{L.C.M.}(D,D-d)}{\mathbf{H.C.F.}(D,D-d)} \times P_r$ segments. ∎

7 Conclusions

In this paper, we proposed a video data block placement scheme which aims at providing a load balanced cost-effective video-on-demand storage system by taking advantage of subband video coding techniques. The band splitting characteristic of subband video coding enables us to stripe the video frame components across the disks and utilize the disk bandwidth more effectively. Furthermore, the subband video coding scheme offers the multi-resolution and multi-rate properties for each encoded video file. The multi-resolution coding embeds video data of different display resolutions in a single video stream; thus the system can provide multi-resolution viewing services without replicating several sets of the same video with different resolution support. The multi-rate property allows the video server system to extract subsets of the embedded video stream for use in VCR functionality. These two properties enable the system to store one set of video data and support multi-resolution as well as VCR display services.

The principle behind the data block placement scheme involves allocating video blocks belonging to the same display resolution of two consecutive video segments to disjoint sets of disks. The purpose of this scheme is to maintain the same bandwidth requirements on disks and the network during both, normal and VCR display periods. In order to make the placement strategy feasible, two conditions have to be satisfied. The first condition is that $d = \sum_{i=0}^{n} b_i$ disks must be used, and the second condition is that the number of disk blocks of display resolution j, $\sum_{i=0}^{j} b_i$ must be an integer multiple of the number of disk blocks of display resolution $j-1$. These two conditions, together with our placement algorithm, guarantee that video blocks of any two consecutive video

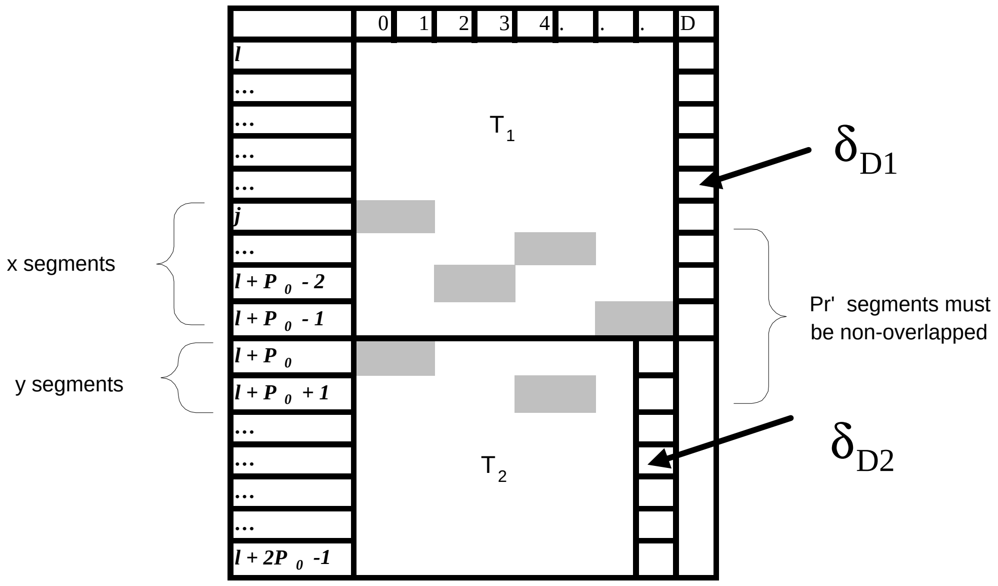

Fig. 21 Fragments for T_1 and T_2.

segments belonging to the same display resolution (except for the highest display resolution) can be stored on different sets of disks. In the latter part of the paper, we relax the first condition and present the data block placement algorithm where the number of disks in the system D, can be larger than d.

In addition to offering constant data transfer requirements during normal and VCR display periods, the data block placement scheme also maintains the load balancing feature for all possible display modes. The load balancing feature of the storage system avoids the occurrence of “hot spots” which is possible in “conventional” storage architectures and is key to providing cost-effective services. A mathematical proof of this important feature is also given in this work.

System resource management is a critical issue in designing video servers. Generally, system resources, including computing power, I/O bandwidth, as well as memory buffers, are relatively expensive. Thus, if they are not well managed, the system will not be cost-effective. To deal with this problem, we have developed efficient retrieval scheduling, buffer management, and admission control policies. With the aid of a retrieval scheduling map, the video server can admit an appropriate number of users effectively so that the system resources are not over-utilized, i.e., high utilization of disk bandwidth can be achieved while still maintaining continuity in data delivery.

Acknowledgments: We would like to thank the anonymous referees for their helpful and insightful suggestions.

References

1. E. Chang and A. Zakhor. Subband video coding based on velocity filters. *IEEE Visual Signal Processing Workshop*, pages 2288–2291, September 1992.
2. M.S. Chen and P.S. Yu D. D. Kandlur. Storage and retrieval methods to support fully interactive playout in a disk-array-based video server. In *ACM Multimedia Systems*, volume 3(2), pages 126–135, 1995.
3. A. Zakhor E. Chang. Scalable video data placement on parallel disk arrays. In *Proceedings of SPIE Symposium on Storage and Retrieval for Image and Video Databases II, San Jose, California*, volume 2185, pages 208–223, 1994.
4. A.D. Gelman, H. Kobrinski, L.S. Smoot, S.B. Weinstein, M.Fortier, and D. Lemay. A store and forward architecture for video on demand service. *proc. IEEE ICC*, pages 27.3.1 – 27.3.5, 1991.
5. L. Golubchik, John C.S. Lui, E. de Souza E. Silva, and R. Gail. Evaluation of tradeoffs in resource management techniques for multimedia storage servers. In *IEEE International Conference on Multimedia Systems*, 1999.
6. T.D.C. Little H.J. Chen, A. Krishnamurthy and D. Venkatesh. A scalable video on demand service for the provision of vcr-like functions. In *Proc. 2nd International Conference on Multimedia Computing Systems, Washington DC.*, May 1995.
7. K. Keeton and R.H. Katz. Evaluating video layout strategies for a high performance storage server. In *ACM Multimedia Systems*, volume 3(2), pages 43–52, 1995.
8. Deepak R. Kenchammana-Hosekote and Jaideep Srivastava. An I/O scheduler for continuous media, part I: Steady state. Technical Report 95-039, University of Minnesota, Computer Science Department, 1995.
9. Jens-Rainer Ohm. Three-dimensional subband coding with motion compensation. *IEEE Trasaction on Image Processing*, 3(5):559 – 571, September 1994.
10. David Pegler, David Hutchison, Phillip Lougher, and Doug Shepherd. A scalable architecture for multimedia storage. In *High-Speed Networking for Multimedia Applications*, chapter 16, pages 127–263. Kluwer Academic Publishers, 1996.
11. Christine I. Podilchuk, Nikil S. Jayant, and Nariman Farvardin. Three-dimensional subband coding of video.

IEEE Trasaction on Image Processing, 4(2):125 – 139, February 1995.

12. D. Rotem and J.L. Zhao. Buffer management for video database systems. In *the* 11th *International Conference on Data Engineering*, March 1995.
13. David Taubman and Avideh Zakhor. Multirate 3-d subband coding of video. *IEEE Transactions on Image Processing*, 3(5):572–588, September 1994.
14. J. Woods. *Subband Image Coding.* Kluwer Academic. Boston, MA, 1991.
15. J.W. Woods and S.D. O'Neil. Subband coding of images. *IEEE Transactions of Acoustics, Speech and Signal Processing*, 34(5):1278–1288, 1986.