Scrap : Data Reorganization and Placement of Two Dimensional Scalable Video in a Disk Array-based Video Server

Seung-Ho Lim, Man-Keun Seo and Kyu Ho Park
Computer Engineering Research Laboratory
Department of Electrical Engineering and Computer Science, KAIST
{shlim, mkseo}@core.kaist.ac.kr, kpark@ee.kaist.ac.kr

Abstract

In this paper, we propose an efficient data reorganization and placement for two dimensional scalable video in a disk array-based video server. In the scalable video coding (SVC), multi-dimensional scalability aspects are implemented, accordingly, all types of these scalability can be exploited at the same time. However, its flexibility would make a non contiguous retrieval of partial stream data, which degrade disk performance greatly. We note that in a disk array the video data can be split into several discontinuous sub-streams by the striping manner of disk array. In view of this, we reorganize sub-streams taking into account both of the decoding dependency of two dimensional scalable video and the location in a disk array. It is shown that the advantages of our placement include: minimizing the seek frequency while maximizing utilization of each disk, maximizing the load balancing and parallelism of disk array. The experimental results show that the proposed reorganization and placement of two dimensional scalable video can significantly improve the performance of video server.

keywords - Scalable Video Coding, Unit Sub-stream, Disk Array

1 Introduction

Scalable video schemes are devised to handle heterogeneity effectively. These are intended to encode the signal once at the highest resolution but enable decoding from partial streams depending on the specific rate and resolution required by a certain application [1]. Therefore, they can satisfy various requirements of multiple users simultaneously from only a single encoded stream. The most recent one of scalable video coding schemes is H.264/MPEG-4 scalable video coding (SVC) [2]. Since SVC provides a multi-dimensional scalability, it supports multiple temporal, spatial and SNR resolutions simultaneously. For this multi-dimensional scalability, SVC enables much more flexible adaptation to various demands of users and network conditions.

Although scalable video schemes have such advantages, the employment of a raw scalable video stream, i.e. video streams are placed in its decoding order, degrades overall disk throughput greatly in disk based server [3]. This is mainly due to the increase of disk requests. In general, a video server transmits streaming data to each client every service round. The service round can be defined as the time period in which the server should retrieve and transmit video data to guarantee real-time playback capability of the client. With a scalable video, the video server has to extract the exact sub-stream data that corresponds to the requested resolution, from the full resolution stream. In this case, the extracted sub-stream data may disperse in the disk. Thus, an access of scalable video stream at a corresponding resolution might incur more disk requests that degrade overall disk throughput severely. Alternatively, server retrieve all streams including extra sub-streams that are not requested but located between the currently requested sub-streams. However, it may also cause huge waste of disk bandwidth and memory buffer since a large amount of disk throughput should be consumed to retrieve unnecessary data and these should be retained in memory until transmitting them. The disk throughput is a crucial factor that must be taken into account in a video server design, since disk throughput may restrict the maximum number of clients serviced simultaneously.

In this paper, we propose an efficient data reorganization and placement for two dimensional scalable video in a disk array-based video server. We note that in a disk array the video data can be split into several discontinuous sub-streams by the striping manner of disk array. In this point of view, we reorganize sub-streams taking into account both of the decoding dependency of two dimensional scalable video and the location to be stored in a disk array.
It is shown that the advantages of our placement include: minimizing the seek frequency while maximizing utilization of each disk, maximizing the load balancing and parallelism of disk array. The experimental results show that the proposed reorganizing and placement of two dimensional scalable video can significantly improve the performance of video server. The rest of this paper is organized as follows. In section 2, we present related works. In the section 3, multi-dimensional scalable video coding and preliminaries for further description are presented. In section 4, our proposed data re-organization and placement algorithm in a disk array based video server is described. In section 5, we describe the experiment results, and finally, section 6 concludes our work.

2 Related Work

There have been several work for placement of scalable video stream or multi-resolution non-scalable video stream in one disk or a disk array. In multi-resolution non-scalable video stream, Shenoy [6] and Lim [7] have proposed a placement strategy that interleaves multi-resolution video stream on a disk array and enables a video server to efficiently support playback of these streams at different resolution levels. This placement algorithm ensures that each sub-stream within a stream is independently accessible at any resolution and the seek time and rotational latency overheads are minimized. In addition, they presented an encoding technique that enables a video server to efficiently support scan operations such as fast-forward and rewind. Rangaswami [8] developed the interactive media proxy that transforms non interactive broadcast or multicast streams into interactive ones. They carefully manage disk device by considering disk geometry for allocation and making several stream files according to the fast-forward levels. However, this method consumes large amount of storage space, and they did not consider disk array management.

For the scalable video data, Chang [3] have proposed a strategy for scalable video data placement that maximizes the total data transfer rate on a disk for an arbitrary distribution of requested data rates. The main concept of this strategy is the frame grouping, which orders data rate layers within one storage unit on a disk. It allows the optimal disk operation in each service round by performing one seek and a contiguous read of the exact amount of data requested. Kang [9] presented harmonic placement strategy. In this scheme, the layers are partitioned into a set of lower layers and a set of upper layers. In the lower layer group, they interleave data blocks of all layers within the same service round. Meanwhile, in the upper layer group, they cluster the data blocks in a layer together. Using this scheme, they can reduce disk seek time, since they can cluster the frequently accessed layers together. However, these schemes described above are not fully utilize the characteristics of scalable video in the video server that can provide multi-dimensional scalable video stream. They are limited to only a single dimensional scalable video. Our work is different from these work in that we propose a data reorganization and placement for two dimensional scalable video considering both disk utilization and load balancing in a disk array-based video server.

3 Two Dimensional SVC Rearrangement

In this section, the multi-dimensional characteristics of SVC are described, and our preliminary definitions are explained for further description. SVC provides tools for
three scalability dimensions, which are temporal scalability, spatial scalability and quality (SNR) scalability. We focus on the two of them, spatial and temporal scalability, for the sake of simplicity. Spatial scalability technique encodes a video into several levels that have different spatial resolutions each other. On the other hand, temporal scalability is a technique to encode a video sequence into several levels having different frame rate [5]. These scalability dimensions including spatial and temporal can be easily combined to a general scalable coding scheme which can provide a wide range of spatial and temporal scalability.

Figure 1(a) describes a combined scalability which support simultaneously for spatial and temporal scalability. When a combined scalability is considered, strict notion of layer does not need to apply any more [2]. Instead, we define combined scalability level that consists of $L_s$ and $L_t$, i.e. each scalability dimension has its own level. $L_s$ and $L_t$ represent spatial and temporal scalability level, respectively. The scalability level in each dimension represents the quality of the video in the corresponding dimension. In scalable video stream, data segments can be grouped into a minimum sub-stream that is capable of extending scalability level. Thus, in a scalable video server, data retrievals are requested in units of this minimum sub-stream. We define this sub-stream as unit sub-stream (US) for two-dimensional scalability. The US, $U^k(l, m)$, is defined as a partial stream of $k$th GOP, which is an essential sub-stream for reconstruction of video at the resolution of higher than spatial scalability level $l$ and temporal scalability level $m$. Thus, to reconstruct $k$th GOP at spatial scalability level $L_s$ and temporal scalability level $L_t$, the US’s, $U^k(l, m)$, should be extracted from the entire stream. $GOP^k(l, m)$, sub-streams for $k$th GOP at spatial scalability level $L_s$ and temporal scalability level $L_t$, is represented with US’s as follows:

$$ GOP^k(L_s, L_t) = \{U^k(l, m) | l \leq L_s, m \leq L_t \}. $$

In the Figure 1(a), relation between scalability level and US’s is described. It also represents how scalability level is related to frame rate and frame size.

The encoded scalable video stream are stored in unit of frame, as shown in Figure 1(b). The number marked on the top of each frame represents the decoding order. Basically, data of encoded video stream are stored with their decoding order. To exploit access pattern determined by scalability, data should be partitioned according to US’s for the first time. Figure 1(c) shows this data placement partitioned according to US’s. Starting from this placement, we propose a more efficient placement scheme in the following section.

4 Scrap: SCalable video Reorganization And Placement in a disk array

In this section, we explain the two dimensional scalable video reorganization and placement algorithm in a disk array, which we called Scrap. The goal of this algorithm is to maximize the number of concurrent clients serviced by maximizing effective disk utilization. In disk, reducing seek latency is closely related to disk scheduling between requests, that is out of scope of this paper. We assume that request scheduling is a round robin fashion between users with a pre-defined round duration. Instead of reducing seek latency itself, we try to reduce seek frequency while enlarging the data transfer time.

For the scalable video, the requested video streams are likely to be retrieved with discontinuous manner in one service round duration, since the extracted sub-stream data disperse in the disk. Thus, an access of these streams at a corresponding resolution might incur more disk requests. To reduce seek overhead, server can retrieve the sub-streams including extra sub-streams that are not requested but located between the currently requested sub-streams. In the view of this, our retrieval policy is that one disk request is generated per one round duration for each disk, even though it retrieves unnecessary sub-streams. We try to find the optimal placement based on this retrieval policy. Meanwhile, the request load balancing is important between disks of a disk array. When video streams are stored into disk array, disk striping is performed by dividing the video data into blocks according to their decoding order and storing these blocks into different disks. Sub-streams at a corresponding resolution might be located in some disks but not in some disks. It incurs a biased disk requests and load imbalance between disks, which is not efficient in a disk array-based server. Thus, the optimal data placement can be obtained by finding the placement which satisfies both of two criteria:

**Criterion 1.** For each disk request, the server should retrieve minimum unnecessary sub-streams to maximize disk utilization during one service round

**Criterion 2.** The server should generate disk requests to balance loads between disks during one service round

Let us suppose two dimensional scalable video that has three spatial scalability levels and five temporal scalability levels, and we have a disk array consisting of four disks. Scalable video stream is originally arranged and partitioned into USs, as described in the previous section. Then, these are initially stored into disks, in which the stripe means the closed set of one round duration, as shown in Figure 2. The GOP data can be filled to match with striping distance using FGS layer of quality scalability, which is described as.
from the criterion 1, we obtain the first data placement by the following equation.

\[
R(S) = \sum_{i=1}^{L} \sum_{j=1}^{M} p_{ij} \cdot \{ \sum_{k=1}^{N} R_{ij}(S_k) \}. \tag{2}
\]

We can obtain several candidate placements from the Eq. 2. In the next step, we select the one that can maximize the disk load balancing from the criterion 2. Let \( L_{ij}(S) \) denote the load balancing factor for scalability level \( L_s = i \) and \( L_t = j \). Load balancing between disks means how the disk requests are distributed as even as possible, so the overall load balancing factor, \( L(S) \), can be described as following equation.

\[
L(S) = \sum_{i=1}^{L} \sum_{j=1}^{M} p_{ij} \cdot \{ \sum_{k=1}^{N} R_{ij}(S_k) \}. \tag{3}
\]

where \( \delta_{ij} \) denote the number of disks to be accessed for scalability level \( L_s = i \) and \( L_t = j \). Our placement policy is finding the stream sequence \( S_k \) for each disk by finding maximum of the Eq. 3. Actually in the equation, it is impossible to know \( p_{ij} \) exactly in advance, since the distribution of scalability levels may change dynamically as clients’ capabilities and conditions of network, over which clients and a video server are connected, change. Even though the \( p_{ij} \) for every \( i \) and \( j \) can be obtained, a dynamic data placement strategy, which updates the placement of stream data periodically according to data access pattern, can impose heavy load on a video server and render management of the video server much more complex. Thus, in our placement, the client distribution probability is assumed to be pre-defined parameter. In particular, the placement can be optimal when all the scalability level is requested in the same probability.

From now on, we explain the optimal placement search algorithm. Since all the possible placement sequences are \( O((L \cdot M)! \), where \( L = \text{max}(L_s), M = \text{max}(L_t) \), the complexity for finding the optimal data placement is very high, and becomes higher as the number of scalability levels increases. To handle the complexity problems, we use a heuristic placement algorithm that achieves local optimal placement for every scalability levels. In the description, \( S_{\alpha}^{(i)} \) denotes the data sequence ordered after \( \alpha \)th iteration. Let \( \alpha \) be the number for US selection. The selected US is relocated in the \( \alpha \)th location within the sequence, so we call it scraper. Both of \( \alpha \) and \( \alpha \) is the natural number between 1 and \( (L \cdot M) \). It is based on one service round duration, and related to one GOP in here. This local search algorithm is started from the initial decoding placement, as shown in Figure 2. The procedure of local optimal placement search is described as follows.

1. Reorganize a raw scalable video stream, of which data are basically placed in their decoding order, into US’s. Thus, data are ordered according to scalability level. Then, let \( i = 1 \) and \( \alpha = 1 \), accordingly the initial sequence is considered as \( S_1^{(i)} \).
2. Whenever \( i \) increases, the sequence of stream, \( S_{\alpha}^{(i)} \), is re-ordered, in which the scraper, \( US_{\alpha} \), is relocated in the \( \alpha \)th location within the sequence.
3. For each sequence \( S_{\alpha}^{(i)} \), it is splitted into sub sequences, \( S_{\alpha}^{(i)} \), for each disk in a disk array. Then, the total retrieval size, \( R(S_{\alpha}^{(i)}) \), and load balancing factor, \( L(S_{\alpha}^{(i)}) \) is calculated for that sequence from the Eq. 2.

Figure 2. Initial placement in a disk array with 3x4 spatial and temporal scalability levels
4. While the $\alpha$ increases from 1 to $(L \cdot M)$, the scraper is changed. Using this US, $US_0$, the search algorithm is repeated from 2 to 3. Finally, the local optimal sequence of stream, $S$, is selected at the end of the repeat.

When we apply this search algorithm to the initial sequence of Figure 2, we can obtain the placement of Figure 3. In the Figure, there is no unnecessary retrieval, also requests are distributed to 4 disks. According to our local search algorithm, the complexity for finding local optimal sequence is $O((L \cdot M)^2)$, which is much lower than $O((L \cdot M)!)$, while presenting near optimal placement sequence.

5 Evaluation

In this section, we evaluate the performance of our placement scheme. The video server system used in our experiments consists of 2.4GHz Pentium CPU, 512MB RAM and disk array with 4 SCSI disks, model ST318304FC. In the experiment, the scalable video has three spatial scalability and five temporal scalability, so 15 USs exist for one GOP. We fixed the size of each US into 32KB, which is possible with the help of quality scalability. The duration of one service round is set to two seconds so two GOPs are mixed. In the placement, we have configured the disk array with 256KB strip size, which means every 256KB data are interleaved across the disks. Thus, two GOPs are allocated in one stripes with four strips, and next service round is returned to the starting disk.

First, we have experimented for each scalability level with fifty clients access, i.e. fifty clients access the same scalability level. Five placements are compared; (a)original, (b)initial, (c)temporal, (d)spatial and (e)scrap. The original and initial store video data with initial decoding order. The difference between the original and others is request generation patterns; the original generate non-sequential small requests for the exactly necessary USs, and the others generate large sequential data including unnecessary USs. temporal and spatial store video for the one dimensionally (spatial or temporal) aligned order. scrap stores the data with the proposed algorithm. The performance metric is average service time for one round duration. From the results, we identify that the scrap gives much better performance at overall scalability levels.

Next, we experiment for various resolutions assuming that heterogeneous clients access different scalability levels each other. Client distributions represent proportions of clients in each scalability level, which are shown in Table 1. In the table, Narrow, Medium and Wide represent that clients are concentrated on low, middle and high scalability levels, respectively. The $S_k$ and $T_k$ represent kth spatial and temporal scalability levels, respectively. Figure 5 plots these results. From the figures, we identify several features. First of all, in all figures, we identify that scrap gives lowest average response times at all distributions, which enhances server throughput. When we compare the experiments be-
tween spatial and temporal placement, the spatial placement gives better performance than temporal in case of $T – Narrow$ and $T – Medium$, and the temporal placements shows vice versa. It is because spatial and temporal placement react sensitively to their scalability levels. Lastly, as shown in the result of original method, it gives good performance when the concurrent clients are small, however, the average service time increase exponentially as clients increases due to the numerous disk requests.

6 Conclusion

Scalable video schemes are devised to effectively handle heterogeneity. For this multi-dimensional scalability, SVC enables much more flexible adaptation to various demands of users and network conditions. In this paper, we devise an efficient data reorganization and placement for two dimensional scalable video in a disk array-based video server. We note that in a disk array the video data can be split into several discontinuous sub-streams in the striping manner of disk array. In view of this, we reorganize sub-streams taking into account both of the decoding dependency of two dimensional scalable video and the place to be stored in a disk array. It is shown that the advantages of our placement include: minimizing the seek frequency while maximizing utilization of each disk, maximizing the load balancing and parallelism of disk array. The experimental results show that the proposed reorganizing and placement of two dimensional scalable video can significantly improve the performance of video server.

References


