Data Placement for Multi-user Interactive DTV

Raju Rangaswami raju@cs.ucsb.edu
Edward Chang echang@ece.ucsb.edu
Chen Li and **Milton Chen** chenli,miltchen@cs.stanford.edu

Abstract

In this paper, we propose an interactive DTV design that converts non-interactive broadcast DTV streams into interactive ones for multiple simultaneous viewers. To enable viewing interactivity, we show that it is critical to organize data intelligently for improving IO resolution, reducing disk latency and minimizing storage cost. We propose three data placement schemes that offer different tradeoffs between IO resolution, disk latency and storage. By employing different schemes under different workload scenarios, an intelligent system can minimize memory use and hence the system cost.

1 Introduction

To support interactive TV, we have proposed a receiver-based architecture [2, 4]. A receiver (a digital VCR or set-top box) in this architecture is equipped with a large disk. Using this, a viewer can pause a live program to take a break from viewing while the arriving broadcast stream continues to be written to the local disk. The viewer can resume watching the program after the pause with a delay, can replay or can fast-forward the program to get back in "sync" with the broadcast stream.

In this paper, we extend this architecture to support multiple interactive streams on one receiver. With the multi-stream capability, students in a virtual classroom or at a library can watch a live lecture at their own pace via one shared receiver. A family can use one receiver as the stream server at home to support interactivity at multiple playback devices. On the one hand, the receiver acts as a client for broadcasters (or servers) to enable interactivity. On the other hand, it caches streams for servicing a number of end-devices to amortize cost. We believe that this architecture is attractive because it is more *feasible* to deploy than traditional video-on-demand architectures, and it is more *cost-effective* than the single-user digital-VCR model. We discuss this further in section 2.

Designing such a client/server dual system to support interactivity, however, is a non-trivial task. First, the receiver must write broadcast signals to its disk as soon as possible to minimize memory use for caching streams. Second, the receiver must read data from the disk into main memory before the decoder runs out of data. In addition, when simultaneous fast-scans are requested, the reads can happen at very high rates. The system must ensure that all IOs meet their deadlines. We are interested in designing a system that satisfies all the above requirements at the minumum cost.

In this study we show that designing such a receiver efficiently requires consideration of at least three design parameters: IO resolution, disk latency, and storage cost (both memory and disk cost). Since the improvement on one performance factor can often lead to the degradation of the others, tradeoffs between these design parameters must be carefully considered. We propose and experiment with three data placement policies for supporting multiple simultaneous interactive streams. We analyze these policies' optimization objectives and quantify their memory and disk use. Our experimental results show that all our proposed policies use much less memory than a naive sequential placement policy.

2 Related Work

According to a recent survey by Redherring [1], both video on demand (VOD) and single-user digital-VCR devices have not generated much real interest. VOD suffers from bandwidth limitations and quality of service while personal VCRs have had limited acceptance due to high cost. However, interactive streaming video has gained increasing interest recently [3, 11] due to bandwidth explosion. In this study, we propose an architecture that plays a client/server dual role to enable interactivity for multiple end-users economically.

Several schemes have been proposed for supporting interactivity. These schemes can be divided into two approaches: 1) using separate fast-scan streams [6, 11, 12, 13], and 2) skipping frames in the regular stream [7]. The first approach may not be used in the broadcast scenario since a program is broadcast at one single rate. The frame-skipping approach can cause low IO-resolution and consequently low system throughput if not designed carefully. (We discuss this problem in detail in the next section.) The study of [5] proposes a client-side approach that re-encodes frames during normal playback and saves them for replay. This approach can be CPU-intensive since most compression schemes are encoding-side heavy and hence may hinder a CPU from decoding more streams.

3 Data Placement

Now, we propose our data placement schemes to support fast-scan operations for multiple streams. Without loss of generality, we can assume that an MPEG[1] stream consists of m frame-sequences; each sequence has δ frames on average, and is led by an I frame and followed by a number of P and B frames.

To support a K-times speedup fast-scan, the receiver displays one out of every K frames. To allow K to be any positive integer, however, the receiver can suffer from high IO, memory and CPU overhead. This is because a frame that is to be displayed (e.g., a B frame) may depend on some frames that are to be skipped (e.g., an I and a P frame). The receiver may have to end up reading, staging in main memory, and decoding much more frames than it displays. To avoid processing the frames

[1] The Advanced Television System Committee (ATSC) has adopted MPEG2 as the encoding standard of DTV and HDTV.

that are to be skipped, we do not involve any B frame in a fast-scan and it plays back a P frame only if the P frame's dependent I frame also involves in the fast-scan. This restriction will not support fast-scan of every speedup but a few, say five, selected speeds like in a DVD player. A key assumption that we make in designing our data organization schemes is that a fast-scan stream needs to be displayed at a lower rate so that the viewer can comprehend and react in time to the content. A typical DVD player plays a fast-scan stream at 3 to 8 fps (frames per second), instead of 24 to 30 fps, the regular playback speed.

To improve IO resolution, we do not want to read in frames that are not involved in a fast-scan. We propose three placement schemes that separate a video into groups to improve IO resolution. However, separating frames into more than one file incurs additional IO overhead for writes. In addition, we might want to replicate data for improving IO resolution and reducing disk latency. We thus need to design the system carefully to manage the tradeoffs between the following design goals:

1. Improving IO resolution,
2. Reducing disk latency, and
3. Minimizing storage (memory and disk) cost.

We now propose three data placement schemes, each of which is designed to optimize on a subset of the design parameters.

3.1 Round-Robin (RR)

The round-robin scheme is a simple method to improve IO resolution. Let the I frames be numbered as $I_1, I_2, I_3, \ldots, I_m$. This placement scheme stores P and B frames in a separate file and distributes I frames among κ I-files, $F_1, F_2, \ldots, F_\kappa$, in the following round-robin manner:

$$F_i = \{I_{i+n\times\kappa} \mid n = 0, 1, 2, 3, \ldots, (m-1)\} \quad (1)$$

How this scheme works depends on the requested speedup (K) of the fast-scan. Suppose $\delta = 9$ and $\kappa = 3$. A $K = 3$-times speedup fast-scan reads only P frames from the PB-file and all the I frames from the κ I-files. A faster fast-scan (e.g., $K = 54$ or 81) requires reading only one I-file, but at a faster pace by skipping some I frames, which leads to low IO resolution. We could increase κ to improve IO resolution, but increasing κ increases disk latency for preparing I-files and for supporting lower speedup fastscans. To maintain good IO resolution without aggravating disk latency, we propose the TBT schemes next.

3.2 Truncated-Binary-Tree (TBT)

In the TBT approach, we attempt to provide good IO resolution at all fast-scan speeds while maintaining reasonable disk latency. In this scheme, the P frames are stored in a separate P-file and the B frames are stored in a B-file. To provide good IO resolution, we organize I frames into a truncated binary tree structure so that only wanted frames are read. An example TBT tree is presented in Figure 1. A normal playback requires retrieval of frames from the entire tree by performing an inorder traversal. For fast-scans, different tree levels are retrieved depending on the requested speed. The higher the speed, the higher are the tree-levels that the fast-scan operation accesses. For example, a 3δ-times speedup fast-scan accesses the I frames at levels two and three in the tree, and a 6δ-times fast-scan reads I frames at level three only.

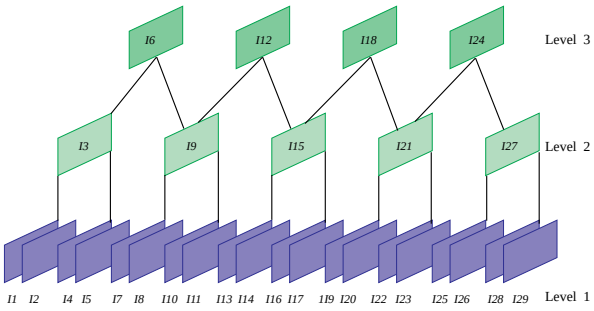


Figure 1: The Truncated Binary Tree Formation.

Formally, the TBT scheme distributes I frames among κ I-files, $F_1, F_2, \ldots, F_\kappa$, in the following manner:

$$F_1 = \{I_k \mid 1 \le k \le m \cap I_k \notin \{F_2 \cup F_3 \cup F_4 \ldots \cup F_\kappa\}\}$$
$$F_i = \{I_{n\times\alpha\times\beta^{i-2}} \mid n \in \{1, 2, 3, \ldots\} \cap$$
$$I_{n\times\alpha\times\beta^{i-2}} \notin \{F_{i+1} \cup F_{i+2} \ldots \cup F_\kappa\}\}, for\ 2 \le i \le \kappa. \quad (2)$$

where m is the total number of I frames in the stream, and α and β are parameters that are tunable to provide for different fast-scan speed requirements. For example, if we let $\delta = 9$, $\alpha = 3$ and $\beta = 3$, then the speedups available are $3, 9, 27, 81$, etc. If we increase β from three to four, then the accessible speedups become $3, 9, 36, 144$, etc. Figure 1 presents an example, which shows how I frames are distributed when $\kappa = 3$, $\alpha = 3$ and $\beta = 2$.

3.3 Replicated Truncated-Binary-Tree (RTBT)

Although the TBT scheme enjoys improved IO resolution, its disk latency can be high due to the need to read data from more than one file. We now introduce a replication scheme that trades disk storage for reducing disk latency. Again, to achieve good IO resolution, we use the same I frame distribution method as the TBT scheme. In addition, we replicate I frames at higher levels of the tree in all files at lower levels. The P-file is now changed to contain all the I frames also whereas the B-file contains all the I frames as well as the P frames. Formally, the I frames in the I-files are changed as follows:

$$F_1 = \{I_k \mid 1 \le k \le m\}$$
$$F_i = \{I_{n\times\alpha\times\beta^{i-2}} \mid n \in \{1, 2, 3, \ldots\}\}, for\ 2 \le i \le \kappa. \quad (3)$$

For example, let $\kappa = 3$, $\alpha = 4$, and $\beta = 2$. Three I-files contain the following I frames:

$$F_1 = \{I_1, I_2, I_3, I_4, \ldots\},\ \ F_2 = \{I_4, I_8, I_{12}, I_{16}, \ldots\}$$
$$F_3 = \{I_8, I_{16}, I_{24}, I_{32}, \ldots\}$$

The advantage of this scheme is that we have one sequential file for supporting each fast-scan speed and hence can achieve 100% IO resolution and minimum disk latency at the same time. During normal playback, for instance, only file F_1 (plus the PB-file) is read. During fast-scans, only one I-file is read. Improving IO resolution reduces memory use. But replicating data on disk increases disk storage cost and increases data transfer overhead to replicate data in multiple I-files.

The three schemes have distinct advantages and disadvantages. Table 1 summarizes the pros (with positive signs) and cons (with negative signs) of the schemes discussed in this section. In the next section, we analyse these pros and cons quantitatively.

Scheme	*IO Resolution*	*Disk Latency*	*Disk Cost*
Sequential	–	+	+
RR	+	–	+
TBT	++	–	+
RTBT	++	+	–

Table 1: Scheme Summary

4 Evaluation

Parameter	*Description*
$\mathcal{N}_w$	*Number of broadcast channels (write streams)*
$\mathcal{N}_r$	*Number of interactive request (read streams)*
ρ	*Ratio of write requests to read requests*
f	*Fraction of read streams that are fast-scans*
$\mathcal{TR}$	*Minimum disk data transfer rate*
$\gamma(d)$	*Worst-case latency function to seek d cylinders*
$\mathcal{DR}$	*Average display rate of MPEG stream*
$\mathcal{DR}_f$	*Average display rate of a fastscan stream*
N	*Throughput of the disk (number of requests served)*
α , β	*Adaptive Tree parameters*
κ	*Number of exclusive I-frame substreams*
ω_w	*Number of files to be written to for one stream*
ω_{rn}	*Number of files to be read for normal stream*
ω_{rf}	*Number of files to be read for fastscan stream*
η	*Space overhead*
Φ	*IO resolution*

Table 2: System Parameters.

As we have discussed, three factors affect the performance of fast-scans: 1) IO resolution, 2) disk latency and 3) storage overhead. In this section, we evaluate our schemes with respect to these factors. The system parameters are described in table 2. The minimum amount of memory required to support $\mathcal{N} = \mathcal{N}_r + \mathcal{N}_w$ simultaneous streams is given by[2]:

$$\mathcal{M}_{min} = \frac{2 \cdot \mathcal{DR} \cdot {\mathcal{N}_r}^2 \cdot \mathcal{TR} \cdot \gamma(d) \cdot X \cdot Y}{\mathcal{TR} - \left[(\rho\eta + (1-f) + \frac{f\mathcal{DR}_f}{\Phi\mathcal{DR}}) \cdot \mathcal{N}_r \cdot \mathcal{DR}\right]}$$

where,

$$X = \left[\rho\omega_w + (1-f)\omega_{rn} + f\omega_{rf}\right], Y = \left[\rho + (1-f) + f\frac{\mathcal{DR}_f}{\mathcal{DR}}\right]$$

Parameter Name	*Value*
Disk Capacity	*26 GBytes*
Number of cylinders, CYL	*50,298*
Min. Transfer Rate $\mathcal{TR}$	*130 Mbps*
Rotational Speed (RPM)	*5,400*
Full Rotational Latency Time	*11.2 ms*
Min. Seek Time	*2.0 ms*
Average Seek Time	*8.9 ms*
Max. Seek Time	*18 ms*

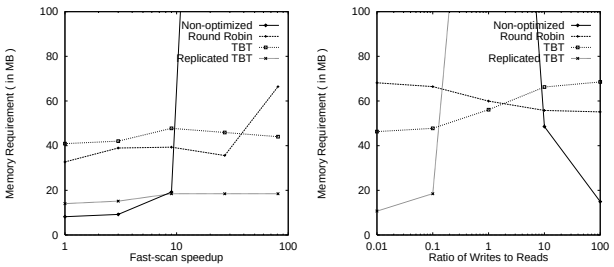
Figure 2: Quantum Fireball Lct Disk Parameters

Figure 2 lists the parameters for the Quantum Fireball disk that we use to study and compare our data placement schemes. In addition, we assume that the peak data consumption and input rates are $\mathcal{DR} = 6.4$ Mbps (the standard DTV bitrate). We also assume that fast-scan streams are played at 5 fps ($\mathcal{DR}_f$) so that the viewer can comprehend and react in time to the content. For computing the seek overhead we follow closely the model developed in [8, 10], which has been proven to be asymptotically close to the real disks. Note that although a different set of disk parameters can lead to different absolute performance values, the relative performance between schemes remain the same.

[2]Please refer to the extended version [9] for a derivation of the equation.

4.1 Memory Requirement

To measure system performance and cost, we calculate the memory requirement for the different schemes we have proposed. Given a system with sufficient disk storage, the memory use determines the cost of the system. We also compute overall cost (memory and disk storage) later in this section.



(a) Varying speedup (b) Varying ρ

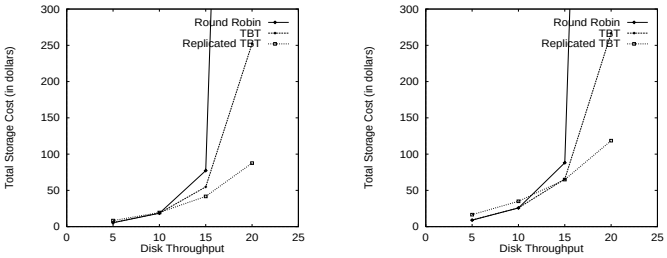
Figure 3: System Memory Requirement.

Figure 3(a) compares the memory use of our placement schemes with the non-optimized scheme (the scheme that stores the video in a sequential file) against varying speeds(K) of fast-scan. For this comparison we assume that the throughput of the disk (N) is 15 and the writes-to-reads ratio (ρ) was set at 0.1. We vary this ratio subsequently. We also assume $\delta = 9$, $\psi = 3$. The truncated binary tree is assumed to be designed with $\alpha = \beta = 3$. When K is large, the non-optimized scheme simply runs out of disk bandwidth to support fast-scans because of poor IO resolution. The round robin scheme uses much less main memory, but even this scheme starts suffering at high speedups. The TBT schemes maintain almost constant memory requirement for all speedups due to their good IO resolution.

Figure 3(b) compares the memory requirement of the placement scheme with the non-optimized scheme against varying write-to-read ratios (ρ). For this experiment, we assumed that the disk was operating at a throughput of 15 simultaneous streams. We also assumed that 20% of the read requests were fast-scan requests, which is reasonable for an interactive TV system. We then calculated the worst-case memory requirement for each scheme (by varying the fast-scan speedup) and obtained numbers for a range of writes-to-reads ratios, ranging from 0.01 to 100. We note from figure 3(b) that it is not clear which scheme would be the most appropriate for a system with dynamic request distribution. Indeed, each scheme performs optimally for a subset of the request distribution spectrum. An intelligent system would need to switch between different schemes to ensure optimal performance under changing request ratios.

To evaluate each placement scheme with respect to all system variables simultaneously, we performed experiments by varying the fast-scan speedup (K) as well as the writes-to-reads ratio (ρ). We do not present them here due to space constraint. Interested readers may refer to the extended version [9] of the paper. We summarize these detailed experiments in the observations section.

4.2 Storage Optimization



(a) 15 mins buffer (b) 30 mins buffer

Figure 4: Total Cost (disk + memory) for TBT schemes.

The replicated TBT scheme replicates data on disk (i.e., uses more disk storage) in order to decrease memory use. In order to quantify the trade-off between disk storage and memory use, we estimate the total cost of a system. In our computations we assumed disk cost as being approximately one-hundredth of memory cost[3], which is assumed to be a dollar per megabyte. The storage cost is directly proportional to the amount of buffer made available to the user for fast-scan operations. For example, providing the user with 30 minutes of rewind or forward buffer requires a storage space of $30 \times 60 \times 6.4$ megabits for standard DTV streams. Since we are interested in the storage trade-off for the TBT scheme, which performs best under dominating reads, we conducted the experiments assuming $\rho = 0.1$ and the worst-scase fastscan speedup ($K = 81$). The presence of distinct crossover points in figure 4 suggests the cost-performance trade-off existing between the various placement schemes. As the number of streams supported increases, the memory cost as well as the disk storage cost increases. However, disk storage cost increases at a lower rate. For 15 and 30 minutes of disk buffering (figure 4(a) and 4(b)), the crossover points occur approximately at throughputs of 10 and 15 streams respectively. These crossover points guide design decisions under varying load. Intuitively we can make the following additional observations:

- If the amount of disk buffer required increases, then the crossover point would occur at a higher level of throughput.
- For higher ρ's, the crossover point would occur at a lesser disk throughput.

Thus, given system requirement and the cost ratios for memory and disk, we know whether replication is beneficial.

4.3 Observations

From the experimental results we make four observations:

- A non-optimized placement scheme (SEQ) incurs huge memory requirement and hence is not desirable.
- For low speedup ($K \leq \kappa \times \delta$) fast-scans, the RR scheme performs well over a wide range of writes-to-reads ratios.
- For high speedup ($K > \kappa \times \delta$) fast-scans, the RR scheme suffers from poor IO resolution. The TBT schemes enjoy good IO resolution and achieve higher throughput than the round robin scheme given the same amount of memory.
- When supporting a large number of users, at low writes-to-reads ratios, the replicated TBT scheme performs best.

[3] This ratio has been more or less constant over recent years

5 Conclusion

We have studied data placement strategies for supporting interactive DTV. The placement schemes we proposed are attractive alternatives to the naive sequential placement policy, which has a high memory overhead. The Round-Robin scheme is easier to conceive and provides good performance for low speedups and for small number of simultaneous users. The Truncated Binary Tree scheme scales well with fast-scan speeds and also performs well for more users. Data replication can further improve the performance of the TBT scheme by trading off some disk storage. We believe that with efficient data organization, the multi-user receiver that we propose in this paper is a more practical and economical approach than the traditional VOD and single-user digital-VCR approaches for enabling viewing interactivity.

References

[1] *http://www.redherring.com/industries/2000/0801/ind-video080100.html*, August 2000.

[2] E. Chang. Maximizing qos for interactive dtv clients. *The Computer Communications Journal (Special Issue), Elsevier*, 23(3):205–218, Febeuary 2000.

[3] E. Chang and H. Garcia-Molina. Effective memory use in a media server. *Proceedings of the 23rd VLDB Conference*, pages 496–505, August 1997.

[4] E. Chang and H. Garcia-Molina. Medic: Memory and disk cache for multimedia clients. *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pages 493–499, June 1999.

[5] M. S. Chen and D. D. Kandlur. Downloading and stream conversion: Supporting interactive playout of videos in a client station. *ICMCS*, May 1995.

[6] M. S. Chen and D. D. Kandlur. Stream conversion to support interactive video playout. *IEEE Multimedia*, 3(2):51–58, 1996.

[7] W. Feng, F. Jahanian, and S. Sechrest. Providing vcr functionality in a constant quality video-on-demand transportation service. *ICMCS*, June 1996.

[8] D. Kotz, S. B. Toh, and S. Radhakrishnan. A detailed simulation model of the hp 97560 disk drive. *Dartmouth College Technical Report PCS-TR94-220*, 1994.

[9] R. Rangaswami and E. Chang. Data placement for multi-user interactive dtv (extended version). *UCSB Technical Report http://www.cs.ucsb.edu/∼raju/research/publications.html*, February 2001.

[10] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *Computer*, 2:17–28, 1994.

[11] P. Shenoy and H. Vin. Efficient support for interactive operations in multi-resolution video servers. *ACM Multimedia Systems*, 7(3), May 1999.

[12] P. J. Shenoy and H. M. Vin. Efficient support for scan operations in video servers. *Proceedings of the 3rd ACM International Conference on Multimedia*, pages 131–140, 1995.

[13] W. Tavanapong, K. Hua, and J. Wang. A framework for supporting previewing and vcr operations in a low bandwidth environment. *Proceedings of the 5th ACM Multimedia Conference*, November 1997.