A Probabilistic Multi-Cycle Model for Serving Continuous Streams from Disks

Freddy Deppner, Benoit Gennart, Roger D. Hersch
Ecole Polytechnique Fédérale Lausanne, Switzerland

Abstract

Existing models for serving continuous streams from disks are generally based on worst-case deterministic head displacement times and on a round-robin strand service approach consisting of serving all strands once per round. We propose an alternative approach, where the head displacement time is described as a random variable with known probability density and where strands are served in a round-robin manner consecutively several times per service period. By establishing the frame loss rate separately in each cycle and for each of the strand's display deadlines, we compute the mean number of lost frames per second for a given configuration. By serving strands consecutively several times in consecutive cycles, we show that for a given frame loss rate, less buffer space is needed than in the conventional single round approach.

1 Introduction

Designing storage architectures for continuous streams is central to many applications such as video on demand, interactive TV and multimedia computing. With today's technology, storage server architectures are based on disk arrays either directly interfaced by cross-bar switches to high-speed networks [1] or hooked onto powerful computing systems offering network connections.

Since the data access rate from disks is generally the bottleneck when serving simultaneously multiple continuous streams, models capable of determining the number of continuous streams that can be served simultaneously and that give a good estimation of the required buffer sizes are of high importance.

A model for serving strands from disks has been established by Rangan and Vin [6] for the deterministic case. Tobagi [8] describes the trade-offs regarding the number of served strands, blockset sizes and buffering requirements for serving strands from multiple disks. The problem of striping scalable video streams onto disk arrays for the deterministic case has been analysed by Chang, Zakhor [3], Keeton and Katz [5]. Chang and Zakhor propose a striping strategy, where video data accessed during a single round is striped across the disks. They compute an optimal round time satisfying timing and buffer size limitation constraints. Keeton and Katz analyse by simulation and measurements the effect of varying the striping unit size.

Let us first introduce the terminology used throughout the paper. A continuous stream such as a video stream is composed of frames. A certain number of compressed frames forms a data block. A given number k of data blocks read contiguously from a single disk form a *blockset*. Blocksets associated with given streams are called strands. We consider models for serving n strand blocksets with a subsystem composed of a single disk and n buffers. We assume that while buffers associated with different strands are successively filled each with one blockset read from the disk, buffers are emptied in parallel according to each stream's frame playing rate R_f^i (circular buffering).

We assume that the different strand blocksets located on the same disk are stored at random disk locations and that all disk head positions are equiprobable. We also assume that disks have a constant data transfer rate and that the size of contiguously read data units (blocksets) is larger or equal to a single track, thereby enabling access cycles without rotational latency.

Various strategies exist for serving multiple strands simultaneously. One may for example, as in [2] define a fixed repetitive time frame called *round*, within which one blockset per served strand is accessed and transferred into its respective buffer. To respect deterministic isochronicity constraints, the sum of disk access times for all the n accessed blocksets should be smaller than the smallest display time of any of the considered strand blocksets. This display time defines the *round time*.

Since disk access time is composed of head displacement time and data transfer time, proportionally increasing the blockset size of each strand has the effect of increasing the round time, decreasing the influence of head displacement time and increasing the global sustained throughput.

Proc. IEEE Int. Conf. on Multimedia Computing and Systems, June 1996, Hiroshima, 612-617

In the approach we propose, we assume that we serve our n strands repetitively during m successive cycles. We define one *cycle* as the time it takes to serve one blockset per strand. Instead of waiting, as in other schemes [2], until the buffered blocksets of the previously read access cycle have been displayed, we continue accessing the disk for m successive cycles. Since track locations of strand blocksets are equiprobable, we can establish the probability density distribution for the head displacement. We then compute the probability of missing one, two or several frames on appropriate time intervals and compute for each strand a mean frame loss rate due to buffer underflow.

If we assume that for each strand the blockset size k_i is proportional to the display rate, the display time of m blocksets of any of the served streams defines a *service period* composed of m service cycles and a *slack time* (Figure 1). Alternately, we may consider the service period to be formed by m round times, the round time being defined as the time it takes to display one blockset of any one of the served streams. Since the cycle is generally lower than the round, after m cycles there is a high probability that the number of resident blocksets is higher than the number x_i of initial blocksets available at the start of the period. The number of additional blocksets residing in the buffers after m cycles defines the *slack time* during which no disk accesses are made for the currently served strands and the number of blocksets is reduced to its initial value.

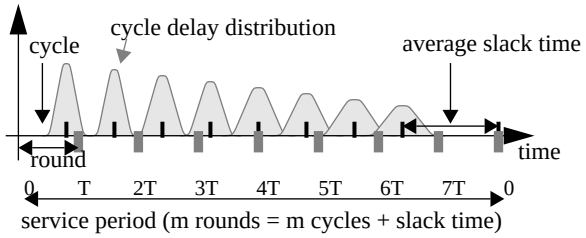

Figure 1. Service period

2 Admission control in the deterministic case

We assume data streams $F_1, F_2, \ldots, F_n$ with frame display rates $R_f^1, R_f^1, \ldots, R_f^n$ [frames/s] being accessed from disks at a granularity of $\eta_1, \eta_2, \ldots, \eta_n$ [frames/blocks] with frame sizes of $s_1, s_2, \ldots, s_n$[bytes/frame].

As described by Vin and Rangan [6] in the deterministic case, when serving n streams simultaneously from a single disk, assuming that client i accesses k_i blocks, and assuming that consecutive blocks within a single blockset are stored on consecutive disk locations, the following inequality must be satisfied:

$$\left(n \cdot l_{ds}^{max} + \sum_{i=1}^{n} k_i \cdot \frac{\eta_i \cdot s_i}{R_d} \right) \leq \min_{i \in [1,n]} \left(k_i \cdot \frac{\eta_i}{R_f^i} \right) \quad (eq\ 1)$$

Lets us consider the case where the block size $b = (\eta_i \cdot s_i)$ is constant for all strands and where strands are displayed at a byte rate of respectively $R_a^1, R_a^2, \ldots, R_a^n$ bytes/s. We assume that the number of blocks k_i per blockset read from the disk for each strand is proportional to the corresponding strand block display rate R_a^i ; i.e. $k_i = qR_a^i/b$. Therefore (eq 1) becomes

$$n \cdot l_{ds}^{max} + q \cdot \sum_{i=1}^{n} \frac{R_a^i}{R_d} \leq q \qquad (eq\ 2)$$

By considering the mean display byte rate $\bar{R}_a$, we obtain an inequality establishing a relationship between the number n of served streams and the proportionality constant q indicating how many blocks per blockset are to be read in each round for each strand.

$$q \geq \frac{n \cdot l_{ds}^{max}}{1 - \frac{n \cdot \bar{R}_a}{R_d}} \quad ; \quad n_{max} = \left\lfloor \frac{R_d}{\bar{R}_a} \right\rfloor \qquad (eq\ 3)$$

In the case of a probabilistic frame loss rate model we can use the same formula as above, but instead of taking the maximal head displacement time l_{ds}^{max} we take the mean head displacement time l_{ds}^{mean} . The maximal number of clients doesn't increase but at fixed blockset size we are able to serve more clients than in the deterministic case.

3 Probabilistic frame loss rate model

This section describes the probabilistic approach for establishing the frame loss rate of a server as a function of the number of users, the initial buffer space, and the number of rounds in a service period.

3.1 Probability distribution of disk head motion delays

Since blocksets are larger than or egal to a single track, we neglect the rotational latency and therefore model the head motion time l as a linear function of the covered distance

$$l = l_{ds}^{min} + \frac{l_{ds}^{max} - l_{ds}^{min}}{p^{max}} \cdot p$$

where l_{ds}^{min} is the fixed delay required for starting the head movement and p^{max} the maximum distance to be covered.

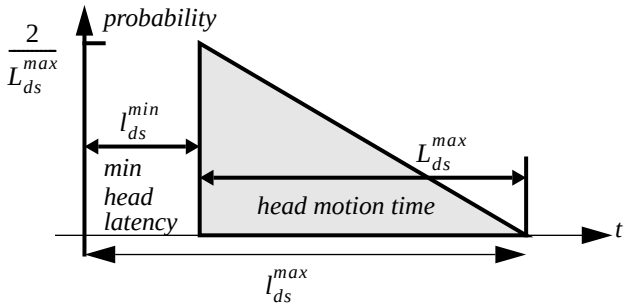

Figure 2. Arm motion delay distribution

We add the constant part of the arm motion time to the data transfer time. The random part of a single disk access is a random variable with probability distribution

$$P\left(L_{ds} = l\right) = \frac{2}{\left(L_{ds}^{max}\right)^2} \cdot \left(L_{ds}^{max} - l\right)$$

The disk access time for client *i* is $T_{ds}^{i} = L_{ds} + t_{ds}^{i}$, with $t_{ds}^{i} = l_{ds}^{min} + d_{ds}^{i}$ being deterministic and taking into account the actual data transfer time, and the minimum head latency. The actual data transfer time is a function of the client's throughput requirement. The admission control condition for n clients during one round (duration : *T*) described in section 2 is rewritten as:

$$n \cdot L_{ds}^{mean} + \sum_{i=0}^{n-1} t_{ds}^{i} < T$$

The sum of the *average* delays for n disk accesses must be smaller than the round time.

3.3 Strand service strategy

Our approach is based on the analysis of a single service period, consisting of *m* rounds. A cycle is the actual length of time it takes to read *n* blocksets from disk. During a cycle, each client performs one disk-access, and reads one blockset per cycle (i.e. k_i blocks per cycle for client i). The admission control condition ensures that the average length of a cycle is always shorter than the length of a round.

The cycle is a random variable. The access delay distribution at the end of *d* consecutive disk-accesses is given by the sum of *d* independent random variables having the distribution of a single disk access. During *m* cycles covering a period, each disk access is started as soon as the previous disk access is complete : all disk accesses in a period are *consecutive*.

3.4 Frame loss rate due to buffer underflow

Assuming each client has the same initial buffer size of *x* blocksets, we are sure to meet all deadlines until the end of round *x*. At the end of round *x*, the probability $P_{bdm}^{0}(x)$ that client 0 encounters buffer underflow (and thus strand discontinuity) is the probability that the first disk access be longer than *x* rounds:

$$P_{bdm}^{0}(x) = P\left(L_{ds} + t_{ds}^{0} > xT\right)$$

In the same round, the probability that client i encounters buffer underflow is the probability of i+1 disk accesses being longer than x rounds:

$$P_{bdm}^{i}(x) = P\left(\sum_{j=0}^{i} L_{ds} + t_{ds}^{j} > xT\right)$$

More generally, the probability that client i encounters buffer underflow during round *(x+r)* is given by:

$$P_{bdm}^{i}(x+r) = P\left(\sum_{j=0}^{i+(r-1)n} L_{ds} + t_{ds}^{j|n} > xT + (r-1)T\right)$$

where *j|n* is the remainder of the division of *j* by *n*.
Note that with this strategy the last clients served in each round are underprivileged. To ensure similar service condition for all clients, we don't start all the clients at the same time but shift the starting time by T/n for successive clients: client 0 is started at time 0, client 1 at time T/n... In that case, the right term of the previous equation becomes *xT+(r-1)T+iT/n*.

The blockset deadline miss rate due to underflow, b_u^{mean}, over a complete service period of *m* rounds, is given by:

$$b_u^{mean} = \frac{1}{mT}\sum_{i=0}^{n-1}\left(\sum_{r=0}^{m-x} P_{bdm}^{i}(x+r)\right)$$

It is possible to compute the mean frame deadline miss rate. The number of frames that the client does not receive depends on how long after the deadline a requested blockset reaches the server memory. Considering a deadline *t*, and a blockset containing *k* frames (one frame per block) that reaches the server memory at $t+\Delta t$, the number of frames that can be served to the client is $\lfloor k \cdot (T - \Delta t)/T \rfloor$. Thus the probability that exactly f frames $(1 \le f \le k)$ are lost by client i during round *x+r* is:

$$P_{fdm}^{i}(x+r, f) = P\left((x+r-1)T + \frac{(f-1)T}{k} + i \cdot \frac{T}{n} \le \sum_{j=0}^{i+n(r-1)} L_{ds} + t_{ds}^{j|n} \le (x+r-1)T + \frac{fT}{k} + i \cdot \frac{T}{n}\right)$$

We get the mean frame loss rate due to underflow, r_u^{mean}, over a service period of m rounds by:

$$r_u^{mean} = \frac{1}{mT} \sum_{i=0}^{n-1} \sum_{r=0}^{m-x} \sum_{f=1}^{k} f \cdot P_{fdm}^{i}(x+r, f)$$

We can not extend the service period *m* indefinitely since we will encounter buffer overflow. To avoid buffer overflow we perform a fixed number *m* of cycles and then stop serving clients during the slack time, assuming a positive slack time. After the slack time, buffers reach their initial level and clients are served again during *m* consecutive cycles. A negative slack time means that we are late. We do not necessarily have a buffer underflow but the buffer size is lower than the initial buffer size. In that case, we drop the remaining disk-requests, and start the service period in the original situation.

The mean error rate is thus the sum of the frame deadline miss rate and the blockset drop rate at the end of the period. The probability of having to drop *at least* one disk request at the end of a period is

$$CP_{bsd}^{1} = P\left(\sum_{i=0}^{m \cdot n - 1} L_{ds} + t_{ds}^{i|n} > mT \right)$$

The probability of having to drop *at least* two disk requests is

$$CP_{bsd}^{2} = P\left(\sum_{i=0}^{m \cdot n - 2} L_{ds} + t_{ds}^{i|n} > mT \right)$$

The probability of having to drop *at least d* disk requests is

$$CP_{bsd}^{d} = P\left(\sum_{i=0}^{m \cdot n - d} L_{ds} + t_{ds}^{i|n} > mT \right)$$

The probability of dropping *exactly d* disk request is $P_{bsd}^{d} = CP_{bsd}^{d} - CP_{bsd}^{d+1}$. In order to restore the buffer levels at the beginning of a service period, the maximum number of blockset requests we might have to drop at the end of the previous service period is equal to $n \cdot x$, the initial number of blocksets available in the server buffer at the beginning of the period. The $n \cdot x$ blockset request-drop corresponds to the case where, at the end of the period, the server buffers are completely empty.

The average number of blockset drops d^{mean} at the end of a period is

$$d^{mean} = \sum_{d=1}^{nx} d \cdot P_{bsd}^{d} = \sum_{d=1}^{nx} CP_{bsd}^{d} - n \cdot x \cdot CP_{bsd}^{nx+1}$$

The average number of frame drops at the end of a period r_p^{mean} is the number of frames *k* per blockset multiplied by the mean number of blockset drops. The frame drop rate r_p^{mean} is:

$$r_p^{mean} = \frac{1}{mT} \cdot (k \cdot d_{mean})$$

3.5 Avoiding buffer overflow.

The difference between the highest disk throughput and the sum of the playing rates over the random length serving cycles gives the throughput at which the buffers may fill up over the initial number of blocksets. In order to avoid buffer overflow, each strand buffer should be large enough not to overflow during the *m-1* service cycles following the initial disk access for the considered strand. The highest buffer filling rate (worst-case) is given when during the considered *m-1* service cycles, the head displacement time is minimal ($l_{ds} = l_{ds}^{min}$).

In the case where all strands have a blockset size proportional to their playing rates and an identical number of bytes per block $b_s = \eta_i \cdot s_i$, the worst-case required buffer size b_i per strand is given by the sum of the number x_i of initial blocksets and the *m* blocksets read from the disks minus the size of the buffer played during *m-1* cycles, i.e during the shortest possible $(m-1) \cdot n$ disk access times.

$$b_i = x_i \cdot k_i \cdot b_s + m \cdot k_i \cdot b_s - (m-1)\left(n \cdot l_{ds}^{min} + \frac{\sum_{j=1}^{n} k_j \cdot b_s}{R_d} \right) \cdot R_a^{i}$$

Let us analyze the required worst-case buffer size b_{req} after *m* cycles and for different blockset sizes, assuming all streams having equal characteristics (identical blockset size *k*, identical number of initial blocksets *x*, identical byte display rate R_a). With a data transfer time $t_{ds} = k \cdot b_s / R_d$, the worst case buffer size b_{req} becomes

$$b_{req} = (x+m)\, k \cdot b_s - (m-1) \cdot n \cdot R_a \cdot (t_{ds} + l_{ds}^{min})$$

The required buffer space grows at a much higher rate when the blockset size *k* increases than when the number of cycles *m* increases. We will see that in order to achieve a given frame loss rate, a certain combination of blockset size and number of cycles minimizes the required buffer space.

4 Design trade-offs and further considerations

This section analyzes the effect of three parameters on the server performance, namely the blockset size *k*, the initial server buffer size *x* and the number of cycles *m* per service period.

We consider a single disk with a minimum, respectively average and maximum latency of 5ms respectively 10ms and 20ms. The throughput is 6MBytes/s. Clients are visualizing a video sequence at a frame rate of 30 frames/s. Each strand is displayed at a rate of 800KBytes/s. According to (eq 3), we can accept 7 clients with a blockset size of at least 32 frames for the minimal blockset size and of 63 frames in the deterministic case.

As explained in section 3.4, the mean frame loss rate is the sum of the underflow and period-blockset-drop rates. The underflow rate can be reduced and even decreased to zero by increasing the initial number of buffered blocksets. Another way of reducing the underflow rate is to increase the blockset size, which reduces the number of disk accesses and hence the relative disk-access latency cost. The period-blockset-drop rate is independent from the initial number of blocksets. It only depends on the number of disk accesses actually completed at the end of a period. There are only two ways to reduce the period-blockset-drop rate: increasing the blockset size or increasing the period length.

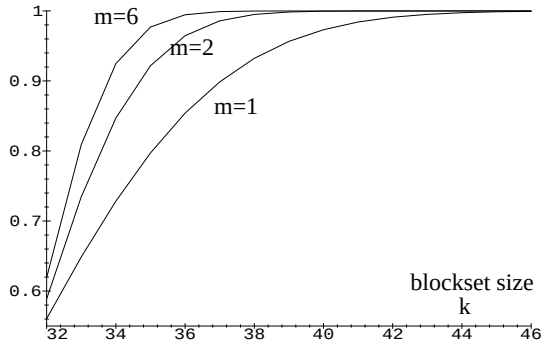

Figure 3. Probability of having a positive slack time

Figure 3 displays the probability of having a positive slack time as a function of the blockset size for a service period length *m* of 1, 3, and 6 rounds. As expected, for *m*=1, this probability increases with the blockset size and becomes equal to 1 for a blockset size *k*>62 since we fall back on the deterministic case. If we increase the number of cycles, the probability converges faster to 1.

For a given period length *m* and blockset size *k*, the number of lost frames at the end of the period is an irreductible part of the mean error rate. If we increase the number of initial blocksets, the frame loss rate due to buffer underflow will drop to zero and only the period blockset drop remains. In our example, for *m=6* and *k=32*, the frame loss rate due to blockset drop at period ends is about 0.27 frames per second per client. The frame loss rate due to buffer underflow is about 2.5 frames per second per client for an empty initial buffer, 0.06 frames/s for an initial buffer of 2/10 blockset per client, and 10^{-8} frames/s for an initial blockset size of 3/10 blocksets per client. Thus a small initial buffer size enables the elimination of the error rate due to buffer underflow.

Figure 4 shows the mean frame loss rate as a function of the number of cycles (period length) and the blockset size for a fixed initial buffer size of 3/10 blockset per client.

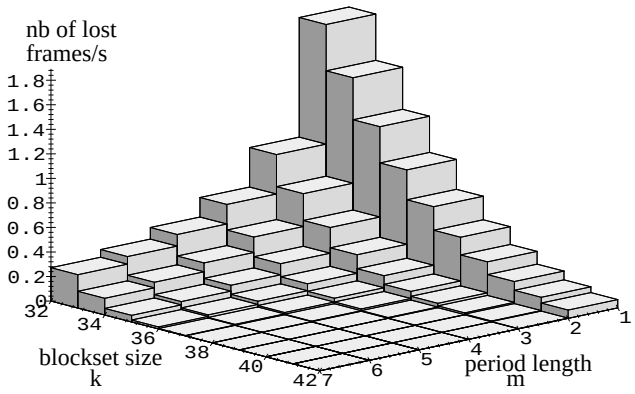

Figure 4. Frame loss rate as a function of period length and blockset size

We can reduce the frame loss rate by increasing the number of initial blocksets, the number of cycles or the blockset size. However all approaches increase the buffering requirements. The required buffer space grows at a much higher rate when the blockset size *k* increases than when the number of cycles *m* increases. For a given frame loss rate, the blockset size is much greater for *m*=1 than for example for *m*=5. Thus, in order to minimize the required buffer space, the number of cycles *m* should be greater than 1. This is the reason for proposing the multi-cycle approach for serving continuous streams.

In our example, for a given frame loss rate, the blockset size is reduced to the mean blockset size k=32 when *m* grows to infinity. We are however limited by the available memory space. For a given error rate, we must choose *k* and *m* so as to minimize the required buffer size. In the case of our example, the plane in figure 5 represents the total buffer requirement for a client in order to avoid buffer overflow. The two curves on this plane are respectively the buffer requirements for strands with a frame loss rate of 1/

1000 frames/s and of 1/100 frames/s. For these two error rates, the minimal required buffer sizes B_{min}(0.001fr/s) = 56.6 frames and B_{min}(0.01fr/s) = 54.6 frames are obtained respectively for (m=3, k=41), (m=4, k=38). In the case of a single cycle service period identical to the single round approach developed in previous models [6], but assuming random head displacement times, we would obtain for the same frame loss rates as above a required buffer size of 65, respectively 61 frames per client. In the deterministic case, the required buffer size would be at least 72 frames per client (see section 2). Thus, in this example, where the disk utilization is nR_a / R_d = *93%*, the probabilistic approach brings in the case of a standard single round approach a reduction of 9.7% in buffer space at a frame loss rate of 1/1000 frames/s and of 15% at a frame loss rate of 1/100 frame/s. The multi-cycle probabilistic approach brings a reduction of 21.4% in buffer space at a frame loss rate of 1/1000 frames/s and of 24.1% at a frame loss rate of 1/100 frame/s.

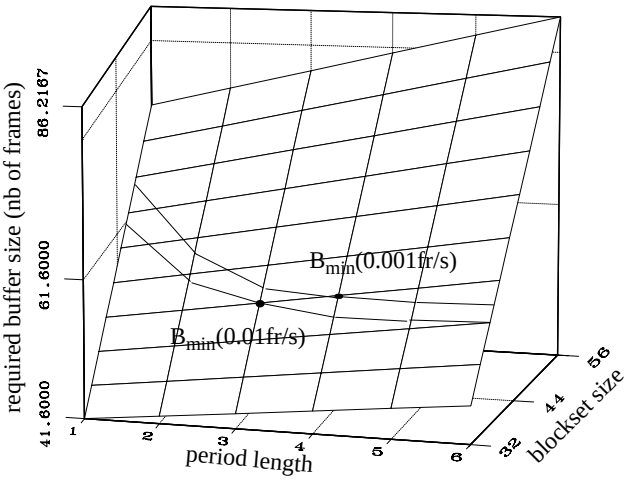

Figure 5. Total required buffer size as a function of blockset size and number of cycles

5 Conclusion

We have presented a new approach for serving continuous streams from disks. Instead of having a slack time at the end of each round, we consider serving all strands consecutively in a round-robin manner during m cycles. By modelling the head displacement time as a random variable and by accepting a given frame loss rate, we need to reserve less buffer space for serving the same number of clients than in the deterministic case. We have developed an approach for computing precisely the mean frame loss rate under given operating conditions, such as the blockset size, the initial number of blocksets loaded into the buffer before starting to display the strands, and the number of cycles during which we consecutively serve all strands in a round-robin manner. Making use of this new strand serving strategy, we are able to compute for a given frame loss rate the number of cycles and the blockset size minimizing the required buffer size. The greater slack times we obtain by choosing a number of cycles greater than one enable blocksets to be preloaded for strands whose service starts in one of the following periods.

Instead of serving clients during a fixed period length of m cycles, we could serve clients until buffers reach a maximal level and then initiate slacktime. Such a strategy would have the benefit of avoiding blockset drop at the end of the service period; it is however more difficult to model since service period length becomes variable.

Our model could also be useful for analyzing variable bit rate strands such as variable quality MPEG implementations. These MPEG coders may enforce output bitrates having given predefined density probabilities. Combined with a scan-edf sheduling algorithm, the head displacement time is then nearly deterministic and the data transfer time becomes the random part of the blockset data access time.

References

[1] Ann L. Drapeau et. al., “RAID-II, A high-bandwidth network file server”, In Proc. 21th Int. Symp. Computer Architecture, Chicago, 1994, 234-244,

[2] P. Lougher, D. Shepard, “The Design and Implementation of a Continuous Media Storage Server”, Proc. 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, Nov 92, LNCS 712, Springer Verlag, 1993, 69-80

[3] E. Chang, A. Zakhor, “Scalable Video Data Placement on Parallel Disk Arrays”, Proceedings SPIE Symposium on Elecronic Imaging, San Jose, Feb. 1995, SPIE Vol. 2185, 208-221

[4] D.J. Gemmel, H.M. Vin, D.K Kandlur, P.V. Rangan, L.A. Rowe, “Multimedia Storage Servers: A Tutorial”, IEEE Computer, May 1995, 40-49

[5] K. Keeton, R.H. Katz, “Evaluating video layout strategies for a high-performance storage server”, Multimedia Systems, No 3, 1995, 43-52

[6] H. Vin, P. Rangan, “Designing a Multi-User HDTV Storage Server”, IEEE Journal on Selected Areas in Communications, Vol 11, No. 1, Jan 1993, 153-164

[7] A.L. N. Reddy, J.C. Wyllie, “I/O Issues in a Multimedia System”, IEEE Computer, March 1994, 69-74

[8] F. A. Tobagi, J. Pang, R. Baird, M. Gang, “Streaming RAID - a disk array management system for video files”, Proc. First ACM Conf. on Multimedia, Anaheim, Calif, Aug 1993, 393-400