Track-Pairing: a Novel Data Layout for VOD Servers with Multi-Zone-Recording Disks *

Yitzhak Birk
Electrical Engineering Department
Technion – Israel Institute of Technology
Haifa 32000, Israel

Abstract

Multi-zone recording increases disk capacity by approximating fixed linear recording density. With fixed rotation speed, however, transfer rate varies with track location, as does the number of video streams that can be played concurrently. This paper proposes Track-Pairing, a deterministic scheme for static intra-disk data layout. By recording each movie alternately on "outer" tracks and their "inner" counterparts, a disk's throughput becomes independent of the viewers' choices and its guaranteed streaming capacity is maximized. With a 1.8:1 ratio of outermost to innermost track capacities, guaranteed streaming capacity is increased by 40 percent, and is merely 22 percent below the streaming capacity of the outermost track! Temporal overhead is modest, and required buffer sizes are smaller than those with the Logical Tracks scheme, which also maximizes guaranteed throughput. Track-Pairing has been implemented under Microsoft's Windows NT, and can be extended to multiple disk drives.

1 Introduction

A video-on-demand (VOD) storage server must produce a large number of concurrent streams of data. Each such stream is typically read from contiguous locations on disk into RAM buffers in large chunks (to reduce disk-arm overhead), and is subsequently streamed to the viewers over a distribution network using fine-grain time-multiplexing. A stream's data rate, R_v, is several times lower than the sustained transfer rate of a single magnetic disk drive (150–600KB/s vs. 3–6MB/s). Once its viewing begins, a stream must not overrun or starve the available RAM buffers. The server must respond promptly to user requests, but the response time to subsequent requests for data may be masked at the cost of extra buffer memory.

*This research was supported in part by the Fund for the Promotion of Research at the Technion and by the Franz Ollendorff Fund. Some of the research was done at Hewlett Packard Labs. The author holds the Milton and Lillian Edwards Academic Lectureship. *Email: birk@ee.technion.ac.il*

VOD differs quite significantly from other prominent applications of large-scale storage subsystems: in on-line transaction processing, for example, performance is measured in accesses per second, and continuity is not an issue; in scientific computing, one often wishes to maximize the transfer rate for a single stream; in file servers, there is usually no notion of streams, and the exact performance measures depend on file size and the type of access. The organization of data in a disk array used for OLTP is discussed in [1]. For general-purpose workstations, schemes such as placing the most latency-critical data in centrally-located tracks and placing different types of data in different disk drives have been proposed [2] [3].

Video servers are storage-centric systems which are likely to be bandwidth- rather than storage-limited since higher-capacity disk drives are less expensive per unit of data. Disks must therefore be used efficiently. However, one must also keep the cost of RAM buffers, which are required for masking disk response time and for storing the chunks of data received from disk, in check. The key performance measure for the storage subsystem of a VOD server is thus throughput per drive, and the regularity of retrieval of data for the various ongoing video streams is important for the minimization of required buffer size.

A number of video-server projects are presently being carried out in industry as well as in universities. Some employ special-purpose hardware with real-time operating systems in an attempt to squeeze the most out of the storage devices and optimally match the computation and communication resources to the VOD requirements. Others use off-the-shelf equipment in an attempt to gain from the price/performance advantage of a large sales volume. Some projects focus on the higher levels of the application, while others are aimed at producing cost-effective "data pumps". The Shark project at IBM [4][5], for example, uses mostly application software written on top of the AIX operating system running on standard RS/6000 systems or their multiprocessor extensions to support diverse VOD services and applications. Special consider-

0-8186-7105-X/95 $4.00

ation is given to the scheduling of requests on behalf of different video streams. The StarWorks video server by Starlight Networks [6][7] uses a real-time operating system on a PC platform.

Multi-zone recording, sometimes referred to as zone-bit recording, is an approximation of fixed linear recording density aimed at increasing a disk's storage capacity. In conjunction with the inevitably-fixed rotation speed of fast disk drives, it results in a dependence of streaming capacity on the radial location of the data being read, and thus on viewing choices. This paper focuses on data layout within MZR disk drives, attempting to optimize performance for streaming applications. Its results can be incorporated into many existing video-server designs.

The remainder of the paper is organized as follows. Section 2 reviews multi-zone recording, section 3 introduces Track-Pairing, and section 4 offers concluding remarks.

2 Multi-zone recording and VOD

2.1 Fixed linear recording density

Fixed linear recording density entails recording a constant number of bits per unit length of a track. The capacity of the innermost track, c_{min}, is equal to that of every track with fixed angular density, but track capacity increases linearly with track radius. Because of the fixed radial spacing of tracks, their capacities thus form an arithmetic sequence. The capacity advantage of fixed linear density over fixed angular density is thus by a factor of

$$\frac{C_{linear}}{C_{angular}} = \frac{c_{max} + c_{min}}{2 \cdot c_{min}},$$

where c_{max} is the capacity of the outermost track. Letting $c_{max}/c_{min} = 1.8$, similar to the HP C2490A 3.5" drive [8], this factor is 1.4.

2.2 Multi-zone recording

In order to reduce the complexity of data organization and to facilitate track-sparing, fixed linear density is usually approximated by *"multi-zone recording"*, sometimes referred to as "zone bit recording". With MZR, the disk is partitioned into "zones" of contiguous tracks. The tracks in any given zone all have the same capacity, which is dictated by the permissible linear density and the length of the zone's innermost track. The HP C2247 3.5" 1GB disk drive, for example, is divided into 8 zones, and the recording density is kept within 5 percent of fixed linear density. The HP C2490A 2GB drive is similarly divided into 14 zones [8],[9]. For facility of exposition and in view of the very close approximation of fixed linear density by MZR, we will speak of the two interchangeably throughout the remainder of the paper except for explicit discussions of the differences.

2.3 Streaming capacity with MZR disks

With fixed rotation speed, transfer rate is proportional to track capacity. The question is to what extent this higher but track-dependent transfer rate can be translated into a higher streaming capacity for VOD.

Proposition 1 *If data for any given stream is placed contiguously in a single set of contiguous tracks on one drive or in the same contiguous sets of tracks on several drives, then: 1) with no restrictions on how a viewer watches a movie (pause, skip, repeat, etc.), the successful concurrent streaming of any given set of movie instances can only be guaranteed based on the transfer rate from the innermost track position of every movie, and 2) the guaranteed throughput of the server (worst case) is equal to that with fixed-angular-density drives.*

Proof. 1) Follows directly from the fact that the viewers of the respective movie instances (there may be multiple concurrent viewers of different locations in the same movie) may all concurrently request viewing of material from the respective innermost tracks. 2) Follows from 1) and the special case in which every movie that is being viewed occupies the innermost track of some drive. □

Proposition 2 *Striping of data across several disk drives improves streaming capacity.*

Proof. Given an arbitrary layout of multiple movies, pick one movie that occupies the innermost track of some disk. Whenever this movie is being viewed, it follows from Proposition 1 that the remaining streaming capacity of the server must be calculated under the assumption that this movie is being read from the innermost track.

Next, stripe this movie across multiple disks on a contiguous set of tracks that includes the innermost ones, moving the data (of other movies) found in those tracks to locations originally occupied by the chosen movie. This operation does not change the smallest (innermost) track occupied by the chosen movie, so its direct effect on the above calculation remains unchanged. The location-swap with other movies moved their data to faster tracks, which did not worsen their situation and may have increased the smallest track numbers occupied by them. Specifically, if striping is across all disks, such an increase is guaranteed. Repeated application of this argument completes the proof. □

Note. The foregoing proof ignored overhead. However, striping data across multiple disk drives would cause each disk drive to participate in the playing of many more streams; with proper arm scheduling, this would result in smaller seek distances, so the proposition would still hold true.

To better exploit MZR, one could attempt to maximize the effective streaming capacity of the server by using a priori knowledge of the average relative viewing frequencies of the movies and placing the most frequently viewed ones in the outermost tracks. However, the relative average viewing frequencies provide a very limited indication of the relative viewing frequencies at any given time of day, day of week, etc. For example, the most frequently viewed movie might be viewed when overall system load is low, in which case it would clearly be unwise to place it in the outermost tracks. Consequently, streaming capacity with this approach may be erratic. One could also dynamically rearrange the movies on the server in anticipation of known viewing patterns or in response to changes, but rearrangement is quite costly and there is still no guarantee of success, except in special cases. Indeed, conservative vendors base their performance claims on the transfer rate of the innermost track, R_{min} [10]. To improve the claimed performance, the use of a substantial number of innermost tracks is sometimes disallowed. Rearrangement is much more beneficial, and even mandatory, when the server also comprises tertiary storage such as tape. Rearrangement resources are best spent on optimizing the content of the disks rather than on moving data within them.

We next present and analyze an altogether different approach: instead of trying to match data placement to the expected viewing frequency, we will make throughput independent of the material being viewed.

3 Track-Pairing

Let c_i, $i = 1, 2, ..., N_t$ denote the (arithmetic) sequence of track capacities (outermost to innermost), where N_t denotes the number of tracks. Clearly, $c_i + c_{N_t - i}$ is the same for all i. We denote this constant by C_p, and refer to track $N_t - i$ as the counterpart of track i. (To simplify notation, and since there are over 1000 tracks per surface, we allow the highest-number track to remain "unused".) We will refer to the $N_t/2$ outermost tracks as "outer tracks" and to the remaining ones — as "inner tracks".

3.1 Recording a movie

The process, described here for a single recording surface and depicted in Fig. 1, proceeds as follows:

1. Pick the desired net duration (reading time) of a time slice for the movie. For simplicity of exposition, let this correspond to an integral number of disk revolutions and hence tracks, denoted n_s.

2. Record the beginning of the movie on some n_s contiguous outer (or inner) free tracks.

3. Record the next portion of the movie on the n_s counterparts of the tracks of step 2.

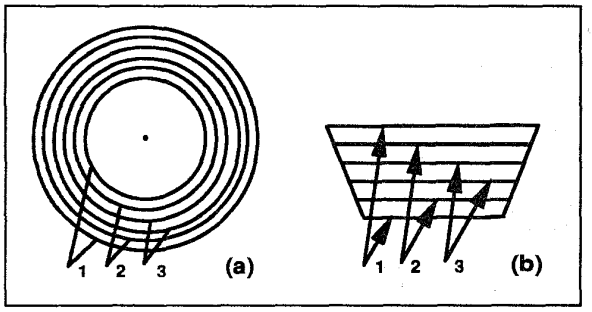

Figure 1: Track-pairing with multi-zone recording, showing a single track per zone. Track capacities are proportional to their lengths. (a) A single recording surface is depicted, along with the pairing of tracks. (b) A linear representation of the track lengths, also showing the pairing.

4. Repeat the previous two steps for the remainder of the movie, alternating between inner and outer tracks.

3.2 Scheduling

Without Track-Pairing, any given stream would typically occupy a contiguous set of tracks on the drive. Consequently, a seek-efficient arm-scheduling algorithm using a unidirectional scan (e.g. grant requests beginning with the outermost track and ending with the innermost, then jump back to the outermost) would cause the various streams to be accessed in the same order in every round.

With Track-Pairing, let us define a schedule round to include two time slices per stream, one on outer tracks and the other on inner tracks. For simplicity of exposition, descriptions will be as if a single track is read in each time slice. The extension to multiple tracks is straightforward.

Consider a single inward sweep of the disk arm, in which each stream is accessed once. Let t_i denote the number of the track that is read in the ith stop, and let i also identify the stream at track t_i. Numbering the tracks from outermost to innermost, it follows that $t_i > t_j$ if and only if $i > j$. According to the placement rule, the next chunk of data for stream i is in track $(N_t - t_i)$. So, the track-number order of the reading points for the streams is simply reversed. By allowing the arm to sweep back from the innermost to the outermost track, reading the next tracks for every stream in this process, the streams would thus be visited in the same order as during the inward sweep. Track-Pairing thus permits a bidirectional elevator schedule. Moreover, consecutive time slices for any given stream are equispaced in the reading sequence.

Grouped accesses.

Track-Pairing places no restrictions on which of the

two locations of a movie is read in any given sweep; efficient access will be maintained so long as locations are alternated from sweep to sweep. It is, nevertheless, possible to impose such restrictions.

Proposition 3 *Grouping of reading locations such that only inner tracks are read during an outward sweep, and only outer ones — during an inward sweep (or vice versa) always reduces average access time.*

Proof. Consider N streams, numbered in ascending order of their current reading points from outer tracks. Let $1 \leq t_i \leq \frac{N_t}{2}$ be the reading point for stream i from an outer track.

With the suggested grouping, the reading order in an entire round is
$t_1, t_2, t_3, ..., t_i, t_{i+1}, ..., t_N, (N_t - t_1), (N_t - t_2), (N_t - t_3), ..., (N_t - t_i), (N_t - t_{i+1}), ..., (N_t - t_N)$.

Without grouping, the sweep direction may still change only once per round. Therefore, there is exactly one seek from an outer track to an inner track and one in the other direction. Without loss of generality, assume that the round begins with an inward sweep, and the switch from outer to inner tracks occurs after reading from t_i. The reading order is
$t_1, t_2, t_3, ..., t_i, (N_t - t_N), ..., (N_t - t_{i+1}), (N_t - t_1), (N_t - t_2), (N_t - t_3), ..., (N_t - t_i), N_n, ..., t_{i+1}$.
In both cases, the head moves between consecutive reading points, and then returns to t_1.

Comparing the sequences and assuming that seek time does not depend on its direction, there are only four seeks in each schedule that do not have identical counterparts in the other. The respective seeks are:
With grouping: $t_i \rightarrow t_{i+1}$, $t_N \rightarrow (N_t - t_1)$, $(N_t - t_i) \rightarrow (N_t - t_{i+1})$, $(N_t - t_N) \rightarrow t_1$.
These represent two different seek distances: $t_{i+1} - t_i$ and $N_t - t_1 - t_N$.
Without grouping: $t_i \rightarrow (N_t - t_N)$, $(N_t - t_{i+1}) \rightarrow (N_t - t_1)$, $(N_t - t_i) \rightarrow t_N$, $t_{i+1} \rightarrow t_1$.
Again, there are two different seek distances: $N_t - t_N - t_i$ and $t_{i+1} - t_1$.

Since $t_i \leq N_t/2$, the largest of the four distances is $N_t - t_1 - t_N$, and $t_{i+1} - t_i$ is the smallest. (The sums of the two distances in the two schedules are equal.) Since seek time per track is monotonically non-increasing with seek distance, the sum of the seek times for the smallest track distance and the largest one is less than the sum of the seek times for the two intermediate distances. Consequently, the schedule with grouping has a lower average seek time. □

Despite the advantage of grouping in terms of seek time, one may still be better off by not imposing this restriction: while the difference in average seek time is typically very small, especially when the schedule round is long (as is the case with arrays), this scheduling restriction could substantially increase the response time to new requests.

3.3 Guaranteed transfer rate

Consider an arbitrary pair of tracks, $(i, N_t - i)$, and denote the (fixed) time slice and the disk-revolution time by τ and T, respectively. The guaranteed transfer rate from this (or any other) pair of tracks is

$$\overline{R_{tp}} = \overline{R}(i, t_{N_t - i}) = \frac{\frac{c_i}{T} \cdot \tau + \frac{c_{N_t - i}}{T} \cdot \tau}{2 \cdot \tau} = \frac{C_p}{2 \cdot T}.$$

In contrast, the guaranteed transfer rate without Track-Pairing is that of the innermost track.

Using the pair (t_1, t_{N_t}), it follows that track-pairing increases the guaranteed (over viewing choices) transfer rate of an MZR disk drive by a factor of

$$\frac{\overline{R_{tp}}}{R_{mzr}} = \frac{c_{min} + c_{max}}{2 \cdot c_{min}}.$$

While this result seems intuitive, the reader is cautioned that it is only true because the pairing was based on equal reading times, not on equal amounts of data in the two members of a pair.

3.4 Temporal overhead and buffer size

Let us compare track-pairing on an MZR drive with a non-MZR, fixed-angular-density drive. This "canonical" drive is assumed to have the same number of tracks as the MZR drive, and its transfer rate is $\overline{R_{tp}}$. (It would thus be more expensive.) Temporal overhead as well as buffering requirements will be compared under two sets of assumptions.

Equal amounts of data per schedule round. The schedule for the track-paired drive comprises two accesses per stream whereas that for the canonical drive comprises a single one. Therefore, the net time slice for the canonical drive is twice as long as for the track-paired drive. The larger number of accesses per unit of data for the track-paired drive nearly doubles its temporal overhead. The effect of the larger overhead depends on the chosen working point. Letting ρ denote the ratio of the net duration of a time slice to the sum of that and the seek time in the canonical system,

$$\frac{N_{streams}^{tp}}{N_{streams}^{canonical}} = \frac{2 \cdot \rho - 1}{\rho}.$$

If, as is likely to be the case in throughput-limited video servers, the time slices are large relative to seek times (large ρ), the degradation in streaming capacity will not be severe.

The reward for reduced efficiency is a reduction in required buffer size. Ignoring various spares, the maximum buffer size per stream is approximately equal to the largest chunk of data read on behalf of a stream in a single time slice. Recalling the two-fold larger time slices for the canonical drive and that its transfer

rate equals that of the middle track in the track-paired disk, the buffer-size ratio is bounded from above by

$$\frac{B_{tp}}{B_{canonical}} = \frac{c_{max}/2}{(c_{min}+c_{max})/2} = \frac{c_{max}}{c_{min}+c_{max}}.$$

The value of this bound for the HP 2247 and 2490A is 0.65.

Equal time slices. Here, the schedule round is approximately twice as long for the track-paired disk than for the canonical one, and the amount of data retrieved is twice as large. The number of seeks per unit time (or data) is the same, so temporal overhead is similar as well.

The required buffer size with track-pairing, however, is higher by a factor of up to

$$\frac{B_{tp}}{B_{canonical}} = \frac{2 \cdot c_{max}}{c_{max}+c_{min}}.$$

For an HP C2490A drive, $(c_{max}/c_{min}) \approx 1.8$, so the maximum factor is 1.3.

In both cases, the bounds are only reached for streams read from the innermost and outermost tracks. When reading from other tracks, Track-Pairing does better.

3.5 Further details

Writing- and Reading-order. One might be tempted to read the data during a time slice in accordance with the direction of the sweep. This, however, is wrong. Consecutive tracks are often shifted in position (skewed) to avoid the loss of an entire revolution when crossing a track boundary in the course of reading contiguous sectors; this would be lost if tracks were read in reverse order. Also, since a likely size of a time slice is on the order of the time to read a single cylinder, the potential reduction in seek distance is negligible. Finally, unless the grouping restriction is imposed, any data chunk could be read during an outward sweep or an inward one, so there is no "natural" order. The order in which data for any given time slice is recorded may conform either to the reading order or to the pairing of tracks, provided that it is placed correctly in the buffer when read.

Quantization problems. Although the capacity of a track-pair with MZR comes within a few percent of a constant, the cumulative excess or deficiency in the amount of data retrieved for a stream could create a problem. This is best overcome at recording time, by partitioning the movie into chunks of the nominal size, and placing fractions of each chunk in the outer- and inner-track writing points so as to consume the same angular span in both areas (to within one sector). At reading time, the difference between MZR and fixed linear density will cause a small variability in the duration of time slices, on the order of the variability in rotational latency.

Control information. This is a matter of implementation. Due to the large chunks of data read in each disk access, it presents no problem in performance or storage.

3.6 Extensions

We have so far discussed Track-Pairing on an entire single recording surface. We next present several extensions.

Multiple recording surfaces. Track-pairing can be extended to multiple recording surfaces in the same way as any other data organization, i.e., by treating tracks within a cylinder as adjacent.

Disk arrays. A disk array with synchronized access, whereby an entire stripe is read concurrently, is equivalent to a single disk drive with higher storage capacity and throughput. Consequently, track-pairing extends trivially. Since the duration of a time slice must not be changed in order to operate the disks efficiently, however, the required buffer size per stream increases linearly with the size of the array, which is unacceptable. One solution is to stagger the access schedules to the different disks. This solves the problem but doesn't work in fault-tolerant systems, since the parity group is no longer available for reconstruction.

The amount of data that is needed eventually for a video stream is very large. Unlike other situations in which a computer requests a large amount of data, however, in VOD there is no use for data until it is needed for playing. Data that is read prematurely must be buffered or dumped (and read again).

Track-Pairing can be used with such fault-tolerant arrangements in either one of two schemes: 1) constructing an entire parity group from contiguous data, i.e., treating the entire array as a single disk drive in terms of data layout, and 2) alternating between inner and outer tracks in successive disk drives. It is also possible to partition the array into sub-arrays, trading storage space for RAM buffer size. A detailed discussion of multi-disk arrangements is beyond the scope of this paper, and the interested reader is referred to [11] and [12]. Nonetheless, it is interesting to observe that in certain such arrangements, buffering may also be needed for storing stream data that was read out of order. This is the case when layout scheme 2) is employed and an entire parity group must be read, e.g., when a disk fails.

Paired disk drives. Track-pairing can be applied to a pair of disk drives: track i, $i = 1, 2, ..., N_t$ of one drive is paired with track $N_t - i$ of the other. Unlike with a single disk drive, in which efficient reading re-

quires an up-front decision on the duration of a time slice, the interleaving in the case of a pair of drives is carried out with a granularity of a single track. This permits the efficient reading of any amount of contiguous data. If a track and its counterpart are read concurrently, the two drives appear as a "canonical" drive whose transfer rate is twice as high as the rate of a single track-paired disk and is constant even within a schedule round. The actual reading order must again take track-skewing into account, and the data read from the two drives must, of course, be merged correctly. (Alternatively, one of the disks could be formatted with a reversed skew.) The scheduling of each disk would be in a unidirectional scan order. Also, the efficient reading of arbitrary chunk sizes does not extend to a larger number of disks, except to an array with synchronized access, but this has major problems with buffer sizes [12]. Finally, we note that a similar idea can be applied to a single disk drive by using two arms.

Partial track-pairing. This entails the use of parts of the disk in track-paired mode and others in conventional mode. Partial Track-Pairing can be used in interesting ways. For example, a band of outermost tracks can be used for the "hottest" items, such as a just-released movie. Similarly, non-paired innermost tracks may be used for a backup copy of important data. As another example, one may exclude track-pairs in the center of the disks from the pairing, and use these tracks for latency-critical data.

To use part of the disk in track-paired mode and the rest in any other way, proceed as follows: Mark a contiguous set of tracks, beginning with the outermost one, as non-paired; similarly, mark a set of tracks beginning with the innermost track as non-paired (the two sets needn't be of equal cardinalities); pair the remaining tracks as if they constituted the entire disk; mark all tracks of any desirable subset of pairs as non-paired; treat the paired tracks as a track-paired disk, and use the others in any desirable way.

3.7 An alternative: "Logical tracks" [14]

By this scheme, which is most appropriate when every zone has the same number of tracks or an integer multiple of some fixed number, same-numbered tracks in all zones are grouped to form fixed-size "logical" tracks. The physical-track order within a logical track is by zone number. (One could similarly group cylinders or any number of contiguous tracks.)

The primary objective in [14] was to emulate disk drives with a fixed track size using MZR drives, one important reason being to retain compatibility with existing operating systems. Consequently, the focus there is on reading all constituents of a logical track either concurrently or in immediate succession by zone order, typically from a number of disk drives that equals the number of zones.

In the remainder of this section, we compare the buffer requirements of LT with those of TP in the context of VOD.

Buffer requirements with Logical Tracks. Given a disk with N_z zones and reading the components of a logical track in zone-number order, it follows that the amount of data that has to be buffered per stream reaches a maximum after the reading of $\frac{N_z}{2}$ chunks, one from each of the $\frac{N_z}{2}$ outermost zones. We next derive the peak buffer occupancy per stream under the following assumptions: an initially-empty buffer, equispaced readings from single zones with constant time slices of a single disk revolution, an equal number of tracks per zone, and track capacities of consecutive zones forming an arithmetic sequence.

Letting $k = c_{max}/c_{min}$, $C_{nom} = (c_{max} + c_{min})/2$, and $\Delta = c_i - c_{i+1}$, and defining R_{nom} to be the streaming capacity of a track with capacity C_{nom},

$$c_{max} = \frac{2k}{k+1} \cdot C_{nom}; \quad c_{min} = \frac{2}{k+1} \cdot C_{nom},$$

and

$$\Delta = \frac{(k-1)c_{min}}{N_z - 1} = \frac{2(k-1)}{(k+1)(N_z-1)} C_{nom}.$$

The amount of data read from the $N_z/2$ outermost zones is

$$\sum_{i=0}^{N_z/2-1} c_i = \left(2k - \frac{N_z - 2}{N_z - 1} \cdot \frac{k-1}{2}\right) \cdot \frac{N_z}{2(k+1)} \cdot C_{nom},$$

and the amount consumed (played out) is

$$\left(\frac{N_z}{2} - 1 + \frac{R_v}{R_{nom}}\right) \cdot C_{nom}.$$

The last additive term represents the amount of data played during the reading of the last chunk (a single disk revolution). With current disk drives and video rates, this is at most $0.1 \cdot C_{nom}$.

The maximum buffer occupancy, which is equal to the amount of data read minus the amount consumed, is given by

$$B_{max}^{lt} = \left(\left(2k - \frac{N_z - 2}{N_z - 1} \cdot \frac{k-1}{2}\right) \cdot \frac{N_z}{2(k+1)} - \frac{N_z}{2} + 1 - \frac{R_v}{R_{nom}}\right) \cdot C_{nom}.$$

With Track-Pairing, the maximum buffer size would be

$$\left(1 - \frac{R_v}{R_{nom}}\right) C_{nom} \leq B^{tp}_{max} \leq \left(\frac{2k}{k+1} - \frac{R_v}{R_{nom}}\right) C_{nom},$$

depending on the pair. The following table depicts the relative maximum buffer occupancies (LT/TP) for several values of N_z and k. An additional amount of buffer space may be required in both cases to compensate for scheduling glitches. Also, the small last term was neglected in both expressions. Note that, as one would expect, the two schemes require the same amount of buffer space when $N_z = 2$.

	Maximum buffer occupancy (LT/TP)			
Nz	$k = 1.3$	$k = 1.5$	$k = 1.75$	$k = 2.0$
2	1.00	1.00	1.00	1.00
4	1.04	1.06	1.07	1.08
6	1.09	1.13	1.17	1.20
8	1.15	1.21	1.28	1.32
10	1.21	1.30	1.38	1.44
12	1.26	1.38	1.49	1.57
14	1.32	1.46	1.59	1.69
16	1.38	1.54	1.70	1.82
18	1.43	1.63	1.81	1.94
20	1.49	1.71	1.91	2.07
40	2.07	2.54	2.98	3.31
100	3.80	5.04	6.20	7.06
200	6.68	9.21	11.55	13.31

Table 1: Maximum buffer occupancy per stream with Logical Tracks relative to the worst case with Track-Pairing.

Track-Pairing within logical tracks. Instead of organizing the physical tracks within a logical track in zone number order [14], one could alternate between the outermost and innermost unused physical tracks comprising the logical track. During normal operation, this would essentially equate the required buffer size with that for Track-Pairing. However, the size difference when operating a disk array in degraded mode, which is the larger one, would unfortunately remain unchanged [11][12].

3.8 Implementation

The simplicity of the Track-Pairing scheme, combined with the favorable operating point (large chunks and low overhead), suggest that it should deliver the promised disk performance. The primary implementation challenge on a single disk drive is the battle against the operating system and the disk drive's "intelligence".

We have successfully implemented Track-Pairing on a single HP C2247A disk drive under Microsoft's Windows NT operating system. The entire disk drive was defined to the file system as a single file, and the mapping of locations within the file to locations on the disk drive was kept separately. Also, to permit demonstration using any conventional video-playing application, a circular RAM buffer was created. Data was "pushed" into the buffer, and application calls to the file were redirected to it. Flow control was achieved using read- and write-pointers. Implementation of the scheme on a pair of disk drives is under way. When reading the data and dumping it, we were able to obtain the full performance of the disk drive. Otherwise, we lost some 20 percent, but this appears to be due to the use of synchronous reads rather than to a true bottleneck. For more details, see [15] and [13]

4 Conclusion

The use of multi-zone recording with fixed rotation speed creates a dependence of streaming capacity on viewing choices. Track-Pairing was proposed as a static layout scheme that gets rid of this dependence and maximizes the guaranteed streaming capacity. For disk drives such as the HP C2247 and C2490, with a 1.8:1 ratio of outermost to innermost track capacities, guaranteed streaming capacity is increased by some 40 percent, and is only 22 percent below the streaming capacity of the outermost track! Track-Pairing permits efficient scheduling, and is similar to conventional layouts in terms of both buffering requirements and temporal overhead.

When compared with the tailoring of data placement to viewing frequency, Track-Pairing gives away the promise of the best possible streaming capacity in exchange for avoiding the worst case. It is nonetheless important to note that the best case (all streams coming from movies in the outermost zone) can seldom be relied upon, whereas any performance guarantees must take into account the possibility that all streams are read from inner tracks. Additionally, the most advantageous situations for tailored placement are ones in which a very small fraction of data is viewed nearly all the time. Partial Track-Pairing can be used in such cases to exploit reliable knowledge while dramatically increasing guaranteed performance. Finally, Track-pairing obviates the need for data reorganization to match changing viewing patterns.

When compared with the Logical Tracks scheme, which also maximizes the guaranteed streaming capacity, Track-Pairing requires less buffer space. Its advantages are even more pronounced in the setting of a fault-tolerant array of disk drives, as discussed elsewhere.

We conclude that track-pairing warrants serious consideration as a layout scheme for VOD servers that use disk drives with multi-zone recording, and have demonstrated it.

Acknowledgments. Thought-provoking discussions with Manu Thapar, John Wilkes, David Coggins and It-

tai Tadmor on these and related issues are gratefully acknowledged. The implementation was carried out by Ittai Tadmor, Edan Almog and Noam Kogan in the Parallel Systems Lab. of the Electrical Engineering Dept. at the Technion under the author's supervision. The implementation employed equipment and software donated by Intel, IBM and Microsoft.

References

[1] J. Gray, R. Horst and M. Walker, "Parity striping of disc arrays: low-cost reliable storage with acceptable throughput", Proc. 16th Intnl. Conf. on Very Large Databases, Brisbane, Australia, pp. 148-159, Aug. 1990.

[2] C. Ruemmler, J. Wilkes, "Disk shuffling", Hewlett Packard Technical report HPL-91-156, Oct. 1991.

[3] K. Muller and J. Pasquale, "A high-performance multi-structured file system design," Proc. 13th ACM Symp. on Operating System Principles (SOSP), Asilomar, CA, October 1991, pp. 56–67.

[4] Roger L. Haskin, "The Shark Continuous Media Server", Digest of Papers of IEEE Spring Compcon 1993, San Fransisco, CA, Feb. 1993, pp. 12–16.

[5] Roger L. Haskin and Frank L. Stein, "A System for the Delivery of Interactive Television Programming", Digest of Papers of IEEE Spring Compcon 1995, San Fransisco, CA, Mar. 1995.

[6] F.A. Tobagi and J. Pang, "StarWorks - a video applications server," Digest of Papers of IEEE Spring Compcon 1993, San Fransisco, CA, Feb. 1993, pp. 4–11.

[7] F.A. Tobagi, J. Pang, R. Baird and M. Gang, "Streaming RAID — A disk array management system for video files", Proc. 1st ACM Int'l Conf. on Multimedia, Aug. 1–6, 1993, Anaheim CA.

[8] Hewlett Packard Company, C2486A/88A/90A SCSI-2 Disk Drives – Technical Reference Manual, 1st ed., Sep. 1992.

[9] Hewlett Packard Company, C2240 SCSI-2 disk drive – Technical Reference Manual, 2nd ed., P/N 5960-8346, April 1992.

[10] MicroNet Technology Inc., product information and personal communication.

[11] Y. Birk, "Track-Pairing: a novel data layout for scaled 913VOD storage servers," Hewlett Packard Technical report HPL-95-xxx, to appear, 1995.

[12] Y. Birk, "Deterministic load-balancing schemes for disk-based video-on-demand storage servers", IEEE Int'l Sump. on Mass Storage Sys, Monterey, CA, Sep. 11, 1995 (to appear).

[13] E. Almog, N. Kogan and I. Tadmor, "Video file server prototype," project report, Parallel Sys. Lab, EE Dept., Technion, Haifa, Israel, May 1994.

[14] S.R. Heltzer, J.M. Menon and M.F. Mitoma, "Logical data tracks extending among a plurality of of zones of physical tracks of one or more disk devices", U.S. Patent No. 5,202,799, April 1993.

[15] I. Tadmor, "Disk performance optimization in a video server", M.Sc. Dissertation, EE Dept., Technion, Haifa, Israel, 1995.