Design of Multimedia Storage Systems for On-Demand Playback*

Yen-Jen Oyang, Meng-Huang Lee,
Chun-Hung Wen, and Chih-Yuan Cheng

Department of Computer Science
and Information Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Abstract

This paper presents a comprehensive procedure to design multimedia storage systems for on-demand playback. The design stresses effective utilization of disk bandwidth with minimal data buffer to minimize overall system costs. The design procedure is most distinctive in the following two aspects:

1. *It bases on a tight upper bound of the lumped disk seek time for the Scan disk scheduling algorithm to achieve effective utilization of disk bandwidth.*
2. *It starts with a general two-level hierarchical disk array structure to derive the optimal configuration for specific requirements.*

1 Introduction

In recent years, the design of mass storage systems for multimedia applications has become an active research topic [1, 2, 3, 4, 5, 13]. One of the most important applications of multimedia storage systems is on-demand playback of video or high-quality audio programs [3, 4, 6, 7, 8, 9, 10, 11, 12, 13]. In on-demand playback applications, the storage system supports concurrent retrieval of continuous video or audio programs requested by a large number of clients. Such applications impose two major challenges to mass storage system design:

1. High data retrieval bandwidth –
   The data retrieval bandwidth required by such applications is particularly high due to a large number of clients.
2. Real-time, continuous transfer of data –
   The system must guarantee uninterrupted service to each client.

In order to meet these two challenges with minimal system costs, the designer must develop appropriate data placement and retrieval strategies so that I/O bandwidth of the storage devices is effectively utilized with minimal amount of data buffer. Here, effective utilization of storage device bandwidth means that the number of storage devices required to provide sufficient I/O bandwidth is minimized. In addition to the number of storage devices, another factor that contributes to overall system costs is the size of data buffer. In on-demand playback applications, certain amount of data buffer is needed to guarantee real-time, uninterrupted transfer of data to each client. It is the designer's desire to minimize the amount of data buffer required.

The studies reported in [4, 9] represent the very first efforts to tackle the disk scheduling problem in multimedia storage design. The main deficiency of the early efforts is that utilization of disk bandwidth is extremely ineffective. The disk head needs to sweep across entire disk surface in order to read just one file block of data, or in another term, one retrieval unit of data. Yu, Chen and Kandlur[11], then, proposed the Grouped Sweeping Scheme (GSS) to improve disk bandwidth utilization. However, Yu, Chen, and Kandlur used a linear model for disk seek time. According to Ruemmler and Wilkes[14], a linear model could lead to a wide range of deviation. In multimedia storage system design, this means that disk bandwidth would not be effectively utilized since the designer would need to include a large margin to guarantee real-time constraints. Another important issue that has not been thoroughly studied in previous literature is how disk arrays can be exploited in multimedia storage system design.

This paper presents a comprehensive procedure to design multimedia storage systems for on-demand playback. The

*This research was sponsored in part by the National Science Council of R.O.C. under grant NSC 83-0408-E-002-002.

design stresses effective utilization of disk bandwidth with minimal amount of data buffer. The design procedure is most distinctive in the following two aspects:

1. It bases on a tight upper bound of the lumped disk seek time for the Scan disk scheduling algorithm to achieve effective utilization of disk bandwidth.

2. It starts with a general two-level hierarchical disk array structure to derive the optimal configuration for specific requirements.

In the following part of this paper, section 2 discusses the general organization and operations of the proposed system. Section 3 elaborates the design process that leads to an optimal solution under a given system specification. Section 4 presents two design examples and discusses the effects of various design alternatives. Finally, section 5 concludes the discussion of this paper.

2 General organization and operations

2.1 General organization

Figure 1 depicts the general disk system architecture which the multimedia storage system proposed in this paper is based on. The entire disk system consists of two levels of disk arrays. In the first level, the low level, disks are grouped to form fine-grain disk arrays, e.g. level-3 disk arrays [15]. The fine-grain disk array is then treated logically as an individual disk in the second level of the hierarchy to form a coarse-grain disk array structure, e.g. a level-4 or level-5 disk array [15]. Because playback operations invoke no writes to the disks, we will purposely omit the parity data when we refer to the disk array structure shown in Figure 1.

Figure 2 shows the general rule to place the file blocks, or retrieval units in another term, of a video/audio program in the disk system. In Figure 2, each drawn disk can be a single disk or a low-level fine-grain disk array. Since the discussion here is about how file blocks are interleaved in the high-level of the disk array hierarchy, a low-level fine-grain disk array can be treated logically as an individual disk in this regard.

Figure 2 shows that each disk is evenly partitioned into several regions. Each region is composed of a number of physically consecutive tracks. Disk partition is not mandatory. If partition is not performed, then file blocks are placed in the disk system just like the normal case of a coarse-grain disk array. If disk partition is performed, then file blocks from each individual video/audio program are then interleaved in the disk system according to the following rule:

file blocks with indices
$2RMI + 2jR + k$ and $2RMI + 2jR + (2R - k - 1)$
are placed in region k of disk j, where M is the number of disks or low-level fine-grain disk arrays in the high-level coarse-grain disk array structure, R is the number of regions into which the disk is partitioned, I is any integer number larger than or equal to 0, j is the index of the disk and runs from 0 to $M - 1$, and k is the index of the region and runs from 0 to $R - 1$.

The idea behind the development of the 2-level disk array architecture is to provide various design alternatives. As will be shown later in the paper, two systems with the same number of disks but different organizations will have different characteristics. The main distinctions are the amount of data buffer required and the maximum start-up latency, which is the maximum amount of time a new client needs to wait before the service begins. It is up to the designer's decision to select one design alternative that best fits his/her needs.

The reason behind performing disk partition is to reduce average seek time of disk accesses. The reason to round file block placement at one end of the disk is to optimize disk head movement. As will be shown later, when carrying out data retrieval, the disk head iteratively scans across disk surface in both directions to read file blocks from the active video/audio programs. Therefore, it makes sense to round file block placement at one end of the disk so that the data to be read next are immediately available when the disk head turns around.

In the proposed multimedia storage system design, the basic storage unit is a disk track. That is, a file block(or retrieval unit) comprises one or more disk tracks from each of the disks in a fine-grain disk array. The reason behind adopting this practice is to eliminate rotation latency during disk accesses, which is one of the major overheads of disk accesses. If the disk features on-arrival read-ahead [14] and file blocks always start and end at disk track boundaries, then disk rotation latency can be completely eliminated.

2.2 Disk operations

The multimedia storage system proposed in this paper performs service by dividing the streams into a number of groups and making these groups access disks in an interleaved and synchronized manner. Let M denotes the number of low-level fine-grain disk arrays in the disk system. Then, the system divides the streams into M groups with no group exceeding a predetermined ceiling of number of streams. The ceiling is imposed to guarantee uninterrupted service to each client, i.e. to meet real time requirements. If there are more clients than the system can serve at one time, then the late comers must wait until some slots in

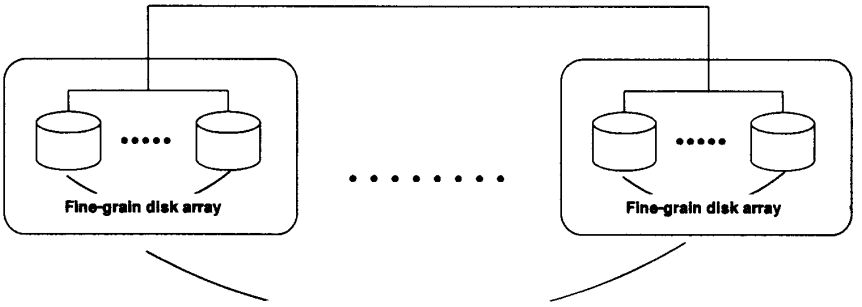

Figure 1: General disk architecture in the proposed multimedia storage system

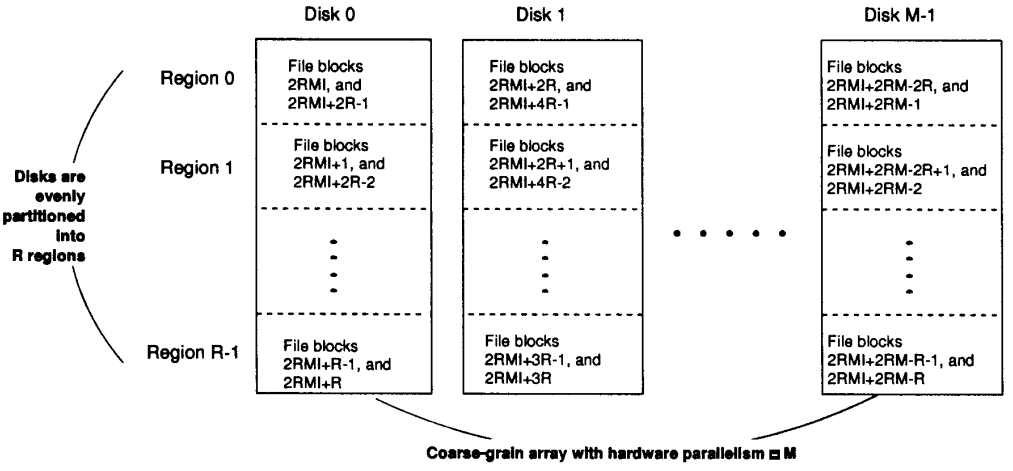

Figure 2: File block placement strategy

these groups become vacant, i.e. some clients terminate their accesses.

The system divides the streams into M groups according to their starting times so that the M groups of streams access the M low-level disk arrays in an interleaved and rotatory manner. Two streams in the same group were admitted to the system either at the same time or at different times but with overlapped shifts. In the later case, these two streams access file blocks with an index offset equal to a multiple of $(2 * R * M)$ simultaneously, where R is the number of regions into which the disk is partitioned into. On the other hand, two groups of streams never access the same disk at any given time.

The M groups of streams access disks in a synchronized manner. That is, the M groups of streams access the same partition region in different low-level disk arrays at the same time. When the disk head scans across one partition region, the system retrieves one file block for each stream. Once the disk head has made a round trip across the disk surface, these M groups of streams rotate and start a new round of access.

Figure 3 demonstrates a simple case of the disk access operation. In Figure 3, there are 4 streams and 2 disks. Each disk has two partition regions. According to the placement policy illustrated in Figure 2, file blocks with indices
$8I, 8I+1, 8I+2, 8I+3,$
are stored in Disk 0 and file blocks with indices
$8I+4, 8I+5, 8I+6, 8I+7,$
are stored in Disk 1, where I is an integer larger than or equal to 0. The 4 streams are divided into 2 groups so that while streams W and X are accessing Disk 0, streams Y and Z are accessing Disk 1, and vice versa. Furthermore, accesses performed by these two groups are synchronized. While stream W is accessing file block $8i + m$ from one

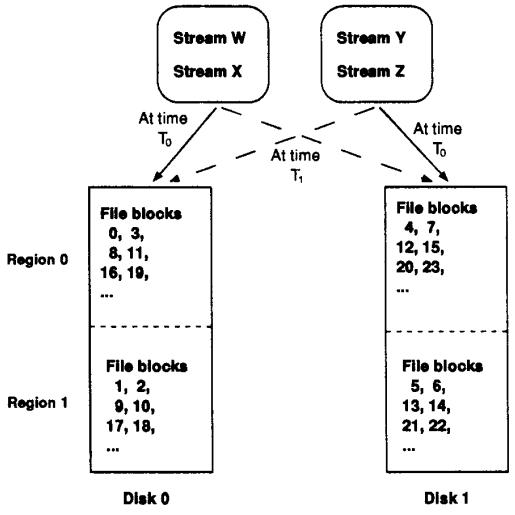

Figure 3: An illustration of the disk access operation

program and stream X is accessing file block $8j + m$ from another program, streams Y and Z are accessing file blocks $(8k + m + 4)$ *modulo* 8 and $(8l + m + 4)$ *mod* 8 from the other two programs, respectively.

During the access operation, the system retrieves and buffers one file block for each stream when the disk head scans across one partition region. Later in time, when the disk head moves to next region, the system transmits the buffered data to the clients and buffers the incoming data in another area of the data buffer. With this practice, it is quite straightforward to figure out that the data buffer must be of size

$$2 * (\text{the size of a file block}) * (\text{the maximum number of streams allowed})$$

The admission control mechanism in the proposed multimedia storage system is quite straightforward. A new client can be admitted only when vacant slots are available in one or more groups of streams. That is, some groups contain less streams than the permitted ceiling. However, even if vacant slots are available, a new client can not start its access until a group with vacant slots has rotated to the low-level disk array that contains file block 0.

3 The design process

This section elaborates a comprehensive procedure to determine the optimal disk system configuration for meeting a given specification. The primary optimization criteria are the effectiveness of disk bandwidth utilization and size of data buffer required. Here, let us use the symbols listed below in the subsequent discussion.

- N: denotes the total number of streams that the system can admit. Assume N is given by the system specification.
- R: denotes the number of regions into which the disk is partitioned into.
- L: denotes the number of disks that each of the low-level, fine-grain disk arrays in the disk system hierarchy contains.
- M: denotes the number of lower-level, fine-grain disk arrays or single disks that the high-level, coarse-grain disk array structure contains.
- G: denotes the maximum number of streams in each group. G is equal to N divided by M.
- S: denotes the number of bytes in a disk track.
- U: denotes the number of disk tracks that a file block contains from each disk. The size of a file block is equal to U times S times L.
- α: denotes the ratio of disk bandwidth utilization that the designer wants to achieve.
- T_o: denotes the worst-case overhead when the disk head scans across one disk partition region and makes G disk accesses. T_o contains two components. The first component is the worst-case lumped sum of seek

time. The second component is due to other system overheads for making G disk accesses such as command issuing, bus initialization, and data transfer. It is assumed here that the disk features on-arrival read-ahead and file blocks always start and end at disk track boundaries. As a result, disk rotation latency is completed eliminated.

- T_r: denotes the time of one disk revolution.
- T_s: denotes the time that the disk head takes to switch from one track to an adjacent track. T_s is often called track switching time.
- B_d: denotes the sustained bandwidth of the disk. It is the bandwidth observed when the disk continues to read data out from consecutive tracks.
- B_s: denotes the data bandwidth required by each stream.

The system design is based on the disk system architecture depicted in Figure 1. The basic idea behind the system design is to achieve effective utilization of disk bandwidth by minimizing the percentage of time the access overhead, T_o defined above, accounts for. In other words, we should make the disk system spend most of time in retrieving data rather than moving disk heads around.

The design process starts with given N, α, B_d, and B_s. The first question that the designer needs to answer is at least how many disks are needed. An approximate answer to this question can be easily obtained by solving the following inequality:

$$D * B_d \geq N * B_s, \tag{1}$$

where D is the number of disks required.

The next issue is to figure out how the disks should be configured and organized. This issue concerns (1) how many regions into which the disk is to be partitioned, i.e. R defined above, and (2) how these disks should be grouped to form a disk array structure as depicted in Figure 1. The design procedure presented in the following proceeds with a pre-determined value of R. The designer may need to go through the design procedure a few times with various values of R in order to determine the optimal configuration that meets the system requirements, i.e. simultaneously serving N streams with each requiring B_s data bandwidth.

With the value of R pre-determined, the designer can derive a formula for computing T_o defined above. T_o contains two components: (1) the maximum lumped sum of seek time and (2) the overhead due to other system operations such as command issuing, bus initialization, and data transfer. The overhead due to other system operations for making one disk access can be modeled by a fixed worst-case lumped sum and is denoted by T_1 here. The maximum lumped sum of seek time is a function of the number of cylinders in one disk partition and the value of G defined above.

According to an article by Ruemmler and Wilkes [14], the disk seek time can be accurately modeled by

$$\text{seek time} = \begin{cases} c_1 + c_2\sqrt{d} & \text{if } d \leq C \\ c_3 + c_4 d & \text{if } d > C, \end{cases} \tag{2}$$

where C is a constant defining the boundary of the two formulas above and d is the distance of disk head movement in number of cylinders. Mathematically, it can be proved that the maximum lumped seek time occurs when the G stops during a scan are evenly apart (See Appendix A). Accordingly, the designer derives the following formula for computing T_o:

$$T_o = \begin{cases} (G+1) * (c_1 + c_2\sqrt{d_0}) + G * T_1 & \text{if } d_0 \leq C, \\ (G+1) * (c_3 + c_4 d_0) + G * T_1 & \text{if } d_0 > C, \end{cases} \tag{3}$$

where d_0 is the number of cylinders in a partition region divided by $G+1$. Note here that, when the disk head scans across a partition region and carries out G disk accesses, the disk actually makes $G+1$ seeks.

Given equation (3), the designer now needs to work out combinations of G, L, and U that satisfy the following two inequalities:

$$T_o \leq (T_o + G * (U * T_r + (U-1) * T_s)) * (1-\alpha) \tag{4}$$

$$B_s * (T_o + G * (U * T_r + (U-1) * T_s)) \leq L * U * S \tag{5}$$

Inequality (4) guarantees the desired level of disk bandwidth utilization is achieved. Meanwhile, inequality (5) guarantees uninterrupted service of video/audio programs.

Altogether, there are three variables G, L, and U in equation (3) and inequalities (4) and (5). The designer can write a computer program to figure out possible combinations of G, L, and U values that satisfy inequalities (4) and (5) and select the one that best matches his/her needs. A note here is that the program should automatically exclude some combinations with unreasonable large values of L and U since the size of data buffer required is

$$2 * M * G * L * U * S.$$

For each combination of G, L, and U values derived above, the designer can easily conclude the number of the lower-level, fine-grain disk array groups needed by computing the minimum value of M that makes the following inequality satisfied

$$M * G \geq N$$

The designer also can conclude that the total number of disks needed is

$$L * M$$

and the size of data buffer required is

$$2 * M * G * L * U * S.$$

In addition to the number of disks and the size of data buffer, another important issue is the maximum start-up latency. In the worst case, a new client needs to wait until a stream group with vacant slots rotates to the low-level fine-grain disk array that contains file block 0. Therefore, the maximum start-up latency is

$$2 * M * R * (T_0 + G * (U * T_r + (U - 1) * T_s)).$$

The discussion so far is under a given value of R. The designer may repeat the design procedure with various values of R and then select an alternative that best fits his/her demands.

4 Design examples and evaluation

This section presents two design examples to illustrate the design process and discusses the effects of different design alternatives. In the first example, the designer is requested to design a system that can handle 40 clients with each requiring 200 Kbytes of data per second. Assume the designer wants to achieve disk bandwidth utilization of more than 80%.

Table 1 gives the specification of the hard disk. The parameters listed in Table 1 are from a calibration of the HP 97560 hard disk carried out by Ruemmler and Wilkes [14]. The seek time of the disk is modeled by

$$\text{seek time} = \begin{cases} 3.24 + 0.400\sqrt{d} & \text{if } d \leq 383 \\ 8.00 + 0.008d & \text{if } d > 383, \end{cases} \quad (6)$$

where d is the distance of disk head movement in number of cylinders and the unit of time is millisecond. The system overhead, i.e. T_1 addressed above, is modeled by a fixed amount of 2 milliseconds

Table 2 summaries the resources required and maximum start-up latency for various design alternatives. Accordingly, the designer can select an alternative that best fits his/her demands. An observation is that, by increasing the R value, the designer can reduce the size of data buffer at the price of increasing the maximum start-up latency. Also, by increasing the L value, the designer can reduce the maximum start-up latency at the price of increasing the data buffer size. In short, it is the designer's choice to select a configuration that most matches his/her needs.

Rotation speed	4000rpm
No. of Cylinders	1962
No. of sectors per track	72
No. of bytes per sector	512 Bytes
Track switching time	1.6ms
Maximum sustained disk bandwidth	2.17 MBytes/sec.

Table 1: Hard disk Specification.

Finally, the designer can evaluate the design by comparing the number of disks needed with the theoretical lower bound. By calculating

$$\frac{40 * 200KBytes}{\text{2.17MBytes per second}},$$

the designer figures out that at least 4 disks are needed.

In the second example, the designer is requested to design a system that can handle 25 clients with each requiring 100 Kbytes of data per second. Assume the designer wants to achieve disk bandwidth utilization of higher than 80%..

In this example, Magneto-Optical(MO) disks, instead of conventional hard disks, are used. Table 3 gives the specification of the MO disk. The system overhead, i.e. T_1 addressed above, is modeled by a fixed amount of 2 milliseconds. The seek time of the MO disk is modeled by

$$\text{seek time} = \begin{cases} 21.9 + 0.0076d & \text{if } d \leq 2500 \\ 30.9 + 0.0040d & \text{if } d > 2500, \end{cases} \quad (7)$$

where d is the distance of disk head movement in number of tracks and the unit of time is millisecond. In this example, piecewise linear equations, rather than the general form used in the first example, are used to model seek time. Since the first order difference function of the seek time modeling function is a monotonically decreasing function, Theorem 1 of Appendix A applies.

Rotation speed	3600rpm
No. of tracks	9953
No. of sectors per track	24
No. of bytes per sector	512 Bytes
Track switching time	1ms
Maximum sustained disk bandwidth	679 KBytes/sec.

Table 3: MO disk Specification.

Table 4 summaries the resources required and maximum start-up latency for various design alternatives. In this

Alternative Values of R and L	Value of G	Value of U	No. of disks required	Size of data buffer required	Maximum start-up latency
R=1, L=1	10	8	4	23,040 KBytes	11.41 seconds
R=1, L=2	20	6	4	34,560 KBytes	8.60 seconds
R=1, L=4	40	5	4	57,600 KBytes	7.16 seconds
R=2, L=1	10	6	4	17,280 KBytes	17.24 seconds
R=2, L=2	20	5	4	28,800 KBytes	14.35 seconds
R=2, L=4	40	5	4	57,600 KBytes	14.20 seconds
R=4, L=1	10	5	4	14,400 KBytes	28.77 seconds
R=4, L=2	20	5	4	28,800 KBytes	28.43 seconds
R=4, L=4	40	4	4	46,080 KBytes	22.89 seconds

Table 2: Design alternatives of the first design example.

examples, the three alternatives with $L = 2$ consistently give inferior designs, in terms of hardware costs and the maximum start-up latency, to the three alternatives with $L = 4$. Therefore, the designer can exclude the three alternatives with $L = 2$. For the remaining alternatives, if the designer wants to minimize the number of disks, then he/she should choose from the three alternatives with $L = 4$. If the designer wants to have a small data buffer, then he/she should choose from the three alternatives with $L = 1$. For the three alternatives with the same L value, the tradeoff is between the size of data buffer and the maximum start-up latency.

The designer finally can evaluate the design by comparing the number of disks needed with the theoretical lower bound. By calculating

$$\frac{25 * 100KBytes}{679\text{KBytes per second}},$$

the designer figures out that at least 4 disks are needed.

5 Conclusion

This paper discusses the design of multimedia storage systems for on-demand playback. The design stresses effective utilization of disk bandwidth with minimal data buffer to minimize overall system costs. The design procedure is most distinctive in the following two aspects:

1. It bases on a tight upper bound of the lumped disk seek time for the Scan disk scheduling algorithm to achieve effective utilization of disk bandwidth.

2. It starts with a general two-level hierarchical disk array structure to derive the optimal configuration for specific requirements.

The design procedure is simple and effective in the sense that the designer can easily work out a few design alternatives and then select the one that best fits his/her requirements according to the hardware resources required and other concerns such as maximum start-up latency.

Acknowledgments

The authors wish to thank Dr. Jan-Ming Ho of Academia Sinica of Taiwan and Dr. John Wilkes of HP Laboratories for many valuable suggestions.

Appendix A

Here, we will present a tight upper bound of lumped disk seek time for the Scan disk scheduling algorithm based on an accurate disk seek time model[14]. The general form of the model is

$$\text{seek time} = \begin{cases} c_1 + c_2\sqrt{d} & \text{if } d \leq D \\ c_3 + c_4 d & \text{if } d > D \end{cases} \quad (8)$$

where D is a constant defining the boundary of the two formulas above and d is the distance of the seek in number of cylinders. A complete mathematical proof can be found in [16]. Here, we just present the basic idea of the proof. In the subsequent discussion, N^+ denotes the set of all positive integers and R denotes the set of all real numbers.

Lemma 1 *Let $h : N^+ \rightarrow R$ be a function with $h(i) \geq h(j)$ for all $i, j \in N^+$ and $i < j$. Then, for an $n_0 \in N^+$ and two series of positive integers $l_1, l_2, \ldots, l_p$ and $m_1, m_2, \ldots, m_q$,we have*

$$\sum_{i=1}^{p}\sum_{k=1}^{l_i} h(n_0 - k) \geq \sum_{j=1}^{q}\sum_{k=1}^{m_j} h(n_0 + k - 1)$$

Alternative Values of R and L	Value of G	Value of U	No. of disks required	Size of data buffer required	Maximum start-up latency
R=1, L=1	5	7	5	4,200 KBytes	8.30 seconds
R=1, L=2	13	39	4	48,672 KBytes	37.42 seconds
R=1, L=4	25	18	4	43,200 KBytes	17.24 seconds
R=2, L=1	5	6	5	3,600 KBytes	14.09 seconds
R=2, L=2	13	35	4	43,680 KBytes	67.18 seconds
R=2, L=4	25	17	4	40,800 KBytes	32.57 seconds
R=4, L=1	5	6	5	3,600 KBytes	27.62 seconds
R=4, L=2	13	33	4	41,184 KBytes	126.7 seconds
R=4, L=4	25	17	4	40,800 KBytes	64.98 seconds

Table 4: Design alternatives of the second design example.

if $\sum_{i=1}^{p} l_i = \sum_{j=1}^{q} m_j$ *and* $n_0 > l_i$ *for all* $1 \le i \le p$.

Proof: Since $h(i) \ge h(j)$ *for all* $i, j \in N^+$ *and* $i < j$, *we have*

$$\sum_{i=1}^{p}\sum_{k=1}^{l_i} h(n_0 - k) \ge \sum_{i=1}^{p} l_i h(n_0)$$

$$= h(n_0)\sum_{i=1}^{p} l_i = h(n_0)\sum_{j=1}^{q} m_j$$

$$= \sum_{j=1}^{q} m_j h(n_0) \ge \sum_{j=1}^{q}\sum_{k=1}^{m_j} h(n_0 + k - 1)$$

□

Lemma 2 *Let* $f : N^+ \to R$ *be a function and* $f'(n) = f(n+1) - f(n)$ *for all* $n \in N^+$. *If* $f'(i) \ge f'(j)$ *for all* $i < j$, *then for a positive integer* n_0 *and a series of integers* $d_1, d_2, \ldots, d_s$ *with* $\sum_{i=1}^{s} d_i = 0$ *and* $n_0 + d_i \ge 1$ *for all* d_i, *we have*

$$s f(n_0) \ge \sum_{i=1}^{s} f(n_0 + d_i)$$

Hint: Without loss of generality, assume $d_1, d_2, \ldots, d_p < 0$ *and* $d_{p+1}, d_{p+2}, \ldots, d_s \ge 0$. *Let*

$$q = s - p,$$
$$l_i = -d_i \quad \text{for} \quad 1 \le i \le p,$$
$$m_j = d_{p+j} \quad \text{for} \quad 1 \le j \le q.$$

Then, apply Lemma 1.
Please see [16] for a complete proof.

Lemma 3 *Let* $f : N^+ \to R$ *defined by*

$$f(n) = \begin{cases} c_1 + c_2\sqrt{n} & \text{if} \quad n \le D_0 \\ c_3 + c_4 n & \text{if} \quad n > D_0 \end{cases}$$

be a monotonically increasing function used to model hard disk seek time, where n *is the distance of the seek in number of cylinders, and* D_0 *is a positive integer. Then, we have*

$$f'(i) \ge f'(j) \quad \text{for all} \quad i < j,$$

$$\text{where } f'(n) = f(n+1) - f(n)$$

Proof: Please see [16].

Theorem 1 *If a disk head scans across a region of* C *cylinders and makes* $(s-1)$ *stops, then the lumped seek time is maximized when the* $(s-1)$ *stops are evenly apart by* C/s. *Here, it is assumed that* C *is a multiple of* s.

Proof: Let $n_0 = C/s$ *and* $d_1, d_2, \ldots, d_s$ *be a series of integer with* $\sum_{i=1}^{s} d_i = 0$ *and* $n_0 + d_i \ge 1$ *for all* $1 \le i \le s$. *Let* $f : N^+ \to R$ *be a function of the general form shown in Lemma 3 that models the disk seek time. Then, by Lemmas 2 and 3, we have*

$$s f(n_0) \ge \sum_{i=1}^{s} f(n_0 + d_i).$$

This means the lumped seek time is maximized when the $(s-1)$ *stops are evenly apart by* C/s. □

In Theorem 1 above, it is assumed that C is a multiple of t. If it is not the case, then the designer can use $\lceil C/t \rceil$ instead to calculate the maximum lumped seek time.

References

[1] C. Yu, W. Sun, and D. Bitton. Efficient placement of audio on optical disks for real-time applications. *Communication ACM*, July 1989.

[2] D. D. Kandlur, M. S. Chen, and Z. Y. Shae. Design of a multimedia storage server. In *IBM Research Report*, June 1991.

[3] Shahram Ghandeharizadeh and Luis Ramos. Continuous retrieval of multimedia data using parallelism. *IEEE Transactions on Knowledge and Data Engineering*, Vol.5, No.4, August 1993.

[4] P. Lougher and D. Shepherd. The design of a storage server for continuous media. *The Computer Journal*, Vol.36, No.1, 1993.

[5] A.L. Narasimha Reddy and James C. Wyllie. I/O issues in a multimedia system. *IEEE Computer*, March 1994.

[6] Sun Microsystem. Multimedia file system. Technical report, Sun Microsystem, August, 1989. Software Release.

[7] David P. Anderson and Goerge Homsy. A continuous media I/O server and its synchronization mechanism. *IEEE Computer*, October 1991.

[8] D. Anderson, Y. Osawa, and R. Govindan. Real-time disk storage and retrieval of digital audio and video. Technical report, U.C. Berkeley, September 1991. UCB Technical Report.

[9] P. Venkat Rangan and Harrick M. Vin. Designing file systems for digital video and audio. In *Proc. of 13th ACM Symp. on Operation Syst. Principles*, pp. 81-94, October 1991.

[10] Jim Gemmell and Stavros Christodoulakis. Principles of delay-sensitive multimedia data storage and retrieval. *ACM Transactions on Information Systems*, January 1992.

[11] Philip S. Yu, Mon-Song Chen, and Dilip D. Kandlur. Design and analysis of a grouped sweeping scheme for multimedia storage management. In *Proceedings of Third International Workshop on Network and Operating System Support for Audio and Video*, November 1992.

[12] Fouad A. Tobagi, Joseph Pang, Randall Baird, and Mark Gang. Streaming RAID - a disk array management system for video files. In *Proceedings of First ACM International Conference on Multimedia*, August 1993.

[13] Craig Federighi and Lawrence A. Rowe. A Distributed Hierarchical Storage Manager for a Video-on-Demand System. In *Proceedings of IS&T/SPIE Symp. on Elec. Imaging Sci. & Tech.*, February 1994.

[14] Chris Ruemmler and John Wilkes. An introduction to disk drive modeling. *IEEE Computer*, Vol.27, No.3, pp. 17-28, March 1994.

[15] Robert Y. Hou Gregory R. Ganger, Bruce L. Worthington and Yale N. Patt. Disk arrays: High-performance, high reliability storage subsystems. *IEEE Computer*, March 1994.

[16] Yen-Jen Oyang, Meng-Huang Lee, Chun-Hung Wen and Chih-Yuan Cheng. Design of Multimedia Storage Systems for On-Demand Playback. Technical report, NTUCSIE 94-03, September 1994.