Optimizing the Placement of Multimedia Objects on Disk Arrays

Harrick M. Vin, **Sriram S. Rao** and **Pawan Goyal**

Distributed Multimedia Computing Laboratory, Department of Computer Sciences
University of Texas at Austin, Austin, Texas 78712-1188
E-mail: {vin,sriram,pawang}@cs.utexas.edu, Telephone: (512) 471-9732, Fax: (512) 471-8885

Abstract

The immensity of the data transfer requirements of multimedia objects will require multimedia servers to be founded on disk arrays. To effectively utilize a disk array, and hence to maximize the number of clients that can be serviced simultaneously, a multimedia server must interleave the storage of each media stream among the disks in the array. In this paper we identify two placement policies for organizing storage of multimedia objects on disk arrays. We present methods for optimizing the performance of server for both the placement policies, and empirically validate and evaluate the methods through extensive trace driven simulation.

1 Introduction

1.1 Motivation

Recent advances in computer and communications technologies has laid a foundation for designing sophisticated multimedia information services in a wide range of application domains. The realization of such services, however, requires the development of high performance, scalable multimedia servers which can provide a range of services to clients over high speed networks. Due to the immensity of the sizes and the data transfer requirements of multimedia objects, such multimedia servers will undeniably be founded on large *disk arrays*. Disk arrays achieve high performance by servicing multiple I/O requests concurrently, as well as by utilizing several disks to service a single request in parallel. The performance of a disk array, however, is critically dependent on the distribution of the workload (i.e., the number of blocks to be retrieved from the array) among the disks. The higher the imbalance in the workload distribution, the lower is the throughput of the disk array.

In the recent past, several research projects have developed techniques for optimizing the throughput of disk arrays [2, 4, 6, 7, 8]. However, most of the analytical as well as simulation models developed for this purpose have presumed conventional workload. That is, retrieval requests do not have any real-time constraints, are aperiodic as well as independent of each other. On the contrary, retrieval of digital audio and video streams impose real-time constraints, and the data accesses are inherently sequential and periodic. Hence, the conventional approach for optimizing the performance of disk arrays is not directly applicable for multimedia storage servers. Techniques for designing multi-disk multimedia servers constitute the focus of this paper.

1.2 Multi-Disk Multimedia Servers

Digitization of audio yields a sequence of samples and that of video yields a sequence of frames. Since audio samples and video frames convey appropriate meaning only when presented continuously in time (unlike text in which spatial continuity is sufficient), a multimedia server must ensure that recording and playback of media streams to and from disks proceed at their real-time rates. Due to the periodic nature of media playback, a multimedia server can service multiple clients simultaneously by proceeding in *rounds*, the duration of which is governed by the response time requirements of clients. During each round, the server retrieves a fixed number of media units (e.g., video frames or audio samples) for each client. To ensure continuous playback, the number of media units accessed for each stream during a round must be large enough to meet their respective playback rate requirement. Furthermore, the service time (i.e., the total time spent in retrieving media units during a round) should not exceed the duration of the round [10, 11].

To effectively utilize a disk array, and hence to maximize the number of clients that can be serviced simultaneously, a multimedia server must interleave the storage of each media stream among the disks in the array. The unit of data interleaving, referred to as a *media bock*, denotes the maximum amount of logically contiguous data that is stored on a single disk (this has also been referred to as the *striping unit* in the literature [3]). Successive media blocks of a stream are placed on consecutive disks using a round-robin allocation algorithm.

Each media block may contain either a fixed number of media units or a fixed number of storage units (e.g., bytes) [1, 5]. If each media stream stored on the array is encoded using a variable bit rate (VBR) compression technique, the storage space requirement may vary from one media unit to another. Hence, a server that composes a media block by accumulating a fixed number of media units will be required to store variable-size media blocks on the array. On the other hand, if media blocks are assumed to be of fixed size, they may contain varying number of media units. Thus, depending on the placement policy, accessing a fixed number of media units during each round may require the server to retrieve either a fixed number of variable-size blocks or a variable number of fixed-size blocks from the array. Although several VBR compression algorithms have been proposed in the recent past [9], very little work has been done in developing techniques for efficient placement and retrieval of compressed multimedia objects from disk arrays.

1.3 Research Contributions of This Paper

In this paper, we analyze and evaluate fixed-size and variable-size block placement policies. For the fixed-size block placement policy, we present a model for characterizing the workload on a disk and deriving an optimal range of media block sizes. Using the model, we analyze the effects of varying number of clients, data rate requirements of clients, and disk array characteristics on the optimal media block sizes. For variable-size block placement policy, we present a method for distributing the workload amongst all the disks in the array, and analyze its effect on response time. To validate our analytical model as well as to compare the two placement policies, we have carried out extensive simulations. We present and analyze our simulation results.

The rest of this paper is organized as follows: In Section 2 and 3, we present and analyze the fixed-size and the variable-size block placement policies. Empirical evaluations of the policies are presented in Section 4, and finally, Section 5 summarizes our results.

2 Fixed-Size Block Placement Policy

In this policy, a multimedia server partitions video streams into fixed size blocks for storing them on the array. Thus, to access a fixed number of frames of VBR encoded video streams during each round, the server will be required to access varying number of blocks for each client. Since the set of disks accessed by a client may be unrelated to those accessed by other clients, the number of media blocks to be accessed during a round may vary from one disk to another. Due to this variation, the service time for the most heavily loaded disk may occasionally exceed the duration of a round, resulting in playback discontinuities for clients. To reduce the occurrence of such discontinuities, the server must select a media block size that minimizes the expected service time of the most heavily loaded disk, technique for which is described below.

2.1 Determining Media Block Size

Consider a multimedia server that is servicing n clients, each retrieving a video stream (say $S_1, S_2, ..., S_n$, respectively). Let $f_1, f_2, ..., f_n$ denote the number of frames of streams $S_1, S_2, ..., S_n$ retrieved during each round, respectively. Let each media stream be partitioned into blocks of size M bytes and stored on an array consisting of D disks. Thus, if Ψ_i denotes maximum data requirement of client i during a round (i.e., the maximum size of f_i frames of stream S_i), then in the worst case, the server will access:

$$a_i = \left\lceil \frac{\Psi_i}{M} \right\rceil$$

blocks of S_i from the array. Given that media blocks of each video stream have been stored on the array using the round-robin placement algorithm, client i will access at least:

$$c_i = \left\lfloor \frac{a_i}{D} \right\rfloor$$

blocks from each disk during that round. Furthermore, if $(a_i \bmod D) \neq 0$, then $(a_i \bmod D)$ disks of the array will be required to access an additional block of S_i. Let random variable $\mathcal{Y}_i^j$ denote the additional load imposed by client i on disk j, where

$$\mathcal{Y}_i^j = \begin{cases} 1 & \text{if client } i \text{ accesses an additional block} \\ & \text{from disk } j \\ 0 & \text{otherwise} \end{cases}$$

Clearly, $\mathcal{Y}_i^j = 1$ only if the first of the additional set of media blocks of stream S_i requested during the round is retrieved either from disk j or any one of the $((a_i \bmod D) - 1)$ previous disks. Since the first block is equally likely to be accessed from any of the disks in the array, we get:

$$P(\mathcal{Y}_i^j = 1) = p_i = \frac{(a_i \bmod D)}{D}$$

Thus, if random variable $\mathcal{X}^j$ denotes the total number of additional blocks accessed from disk j, then

$$\mathcal{X}^j = \sum_{i=1}^{n} \mathcal{Y}_i^j$$

Due to the variability in the frame sizes yielded by VBR compression techniques, even if the retrieval of multiple streams begin from the same disk, the set of disks accessed

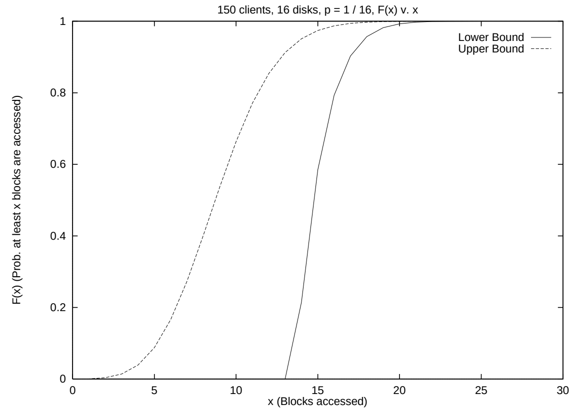

Figure 1 : Variation of $F_{\mathcal{X}_{\max}}(x)$ with x

by the streams become independent of each other within a small number of rounds (i.e., $\mathcal{Y}_i^j$'s are independent). Hence, we get:

$$Z(\mathcal{X}^j) = \prod_{i=1}^{n} Z(\mathcal{Y}_i^j)$$

where $Z(\mathcal{X}^j)$ and $Z(\mathcal{Y}_i^j)$ are the z-transforms of random variable $\mathcal{X}^j$ and $\mathcal{Y}_i^j$, respectively. Since $Z(\mathcal{Y}_i^j) = (1 - p_i) + p_i z$, we get $Z(\mathcal{X}^j) = \prod_{i=1}^{n}((1 - p_i) + p_i z)$. The expected number of additional blocks accessed from the most heavily loaded disk can be determined using the distribution function of $\mathcal{X}_{\max} = \max(\mathcal{X}^j)$. Since the distribution functions of $\mathcal{X}^j$'s yielded by the round-robin allocation algorithm are dependent, it can be shown that (see Appendix A):

$$\max(0, D * F_{\mathcal{X}^j}(x) - (D - 1)) \leq F_{\mathcal{X}_{\max}}(x) \leq F_{\mathcal{X}^j}(x)$$

Figure 1 illustrates the variation of the lower and the upper bound of $F_{\mathcal{X}_{\max}}(x)$ with x. As is evident from Figure 1, the lower bound of $F_{\mathcal{X}_{\max}}(x)$ yields an upper bound on $\mathcal{X}_{\max}$. Since the objective is to minimize transient overloads, in the rest of the paper we will use the lower bound of $F_{\mathcal{X}_{\max}}(x)$ to approximate $F_{\mathcal{X}_{\max}}(x)$.

Since the total number of media blocks to be accessed from disk j is given by $(\mathcal{X}^j + \sum_{i=1}^{n} c_i)$ where c_i's are constants, the expected number of blocks accessed from the most heavily loaded disk can be computed as:

$$\mathcal{B}_{\max} = E(\mathcal{X}_{\max}) + \sum_{i=1}^{n} c_i \qquad (1)$$

Assuming that the server employs SCAN disk scheduling algorithm, the service time for $\mathcal{B}_{\max}$ blocks can be modeled as:

$$\tau(\mathcal{B}_{\max}) = \mathcal{B}_{\max} * (t_{seek}^{exp} + t_{rot}^{exp} + t_{xfer}) \qquad (2)$$

where t_{seek}^{exp}, t_{rot}^{exp} and t_{xfer} denote the seek time, rotational latency, and data transfer time incurred while accessing a block from disk. Assuming that the $\mathcal{B}_{\max}$ blocks are equally spaced across the $\mathcal{C}$ cylinders of a disk, we define $t_{seek}^{exp} = t_{seek}\left(\lfloor \frac{\mathcal{C}}{\mathcal{B}_{\max}} \rfloor\right)$, where:

$$t_{seek}(x) = \begin{cases} 0 & \text{if } x = 0 \\ a\sqrt{x-1} + b(x-1) + c & \text{otherwise} \end{cases}$$

and a, b, and c are constants (determined using physical characteristics of a disk) [7]. The rotational latency t_{rot}^{exp}, is defined to be half of the maximum rotational latency. Thus, given the server configuration (i.e., disk array characteristics, number of clients, and their data rate requirements) and the size of media blocks, Equations (1) and (2) derive the expected service time for the most heavily loaded disk. Let $\tau_1, \tau_2, ..., \tau_k$ denote the service times yielded by repeating the above analysis for media block sizes $M_1, M_2, ..., M_k$. Let $\tau_{\min}$, given by:

$$\tau_{\min} = \min_{i \in [1,k]} \tau_i$$

denote the minimum of the service times yielded for k different values of media block sizes. *A media block size is said to belong to the optimal range if it yields a service time that is within 5% of* $\tau_{\min}$. We denote the media block size that yields the minimum service time as M_{opt}; and the lower and upper bounds of the optimal range as M_{low} and M_{high}, respectively. A multimedia server can reduce the occurrence of transient discontinuities in playback by selecting a media block size in the range $[M_{low}, M_{high}]$.

2.2 Discussion

Although selecting a media block size within the range $[M_{low}, M_{high}]$ reduces the occurrence of transient overloads, it does not eliminate them completely. In fact, the variation in the number of blocks to be accessed from the most heavily loaded disk coupled with the different relative placements of blocks may occasionally yield service times that exceed the duration of the round. We refer to rounds in which service time of the most heavily loaded disk is smaller than or larger than the duration of a round as *underflow* and *overflow* rounds, respectively.

Observe that the sequential nature of video playback enables a multimedia server to precompute the set of blocks to be accessed from each disk for round $(r+1)$ while accessing media blocks during round r. Consequently, if the pattern of block access in round $(r + 1)$ is such that the service time of disk d will exceed the duration of the round (i.e., $(r + 1)$ is an overflow round), then the server can schedule the retrieval of some of the blocks to be accessed from disk d in round $(r + 1)$ during round r as long as the resulting service time for round r remains within the duration of the round. Selection of the set of blocks to be read-ahead during round r may be governed by the increase in service time resulting from their retrieval, and the service requirements of the clients [12]. The reduction in the transient playback discontinuities (and the corresponding improvement in the quality of service being provided to clients) yielded by such read-ahead policies are at the expense of increase in buffer space requirement. However, by judiciously choosing the set of blocks to be read-ahead in underflow rounds, the increase in the buffer space requirement can be minimized [11].

3 Variable-Size Block Placement Policy

Consider a multimedia server that interleaves the storage of video streams on a disk array in terms of variable-size blocks, each containing fixed number of frames. Hence, retrieving f_i frames for each client during a round will require the server to access a fixed number of media blocks from disks. Clearly, to minimize the seek and rotational latency incurred while retrieving f_i frames, each block should contain f_i frames. This placement policy enables the server to access large chunks of data each time a diskhead is repositioned, thereby improving the throughput of the disks.

The most appealing feature of this placement policy is that, regardless of the playback rate requirements, the retrieval of each individual video stream from disk proceeds in *lock-step*. That is, each client accesses exactly one disk during each round, and that consecutive disks in the array will be accessed by the same set of clients during successive rounds.

Since the retrieval of the streams proceed in lock-step, the server can partition the set of clients into D logical *groups* (denoted by $g_0, g_1, ..., g_{D-1}$, respectively). During each round, all clients within a group access the same disk. Assuming that all the clients of group g_i access disk i during round 0, due to the lock-step nature of retrieval, they will access disk $((i+r) \bmod D)$ in round r. If a client requests the retrieval to begin from disk d in round r, then the server attempts to assign the client to the group that is accessing disk d during round r (say g_{new}). Once assigned, the client remains a member of group g_{new} until it departs. To avoid the occurrence of transient overload on any disk in the array, the server must employ admission control algorithm on a group basis [12]. That is, if the addition of the client to group g_{new} would lead to the continuity violations of the clients, then group g_{new} is said to be full and the server must delay admitting the client until a suitable group is found.

As the number of clients being serviced by the server increases, the delay in admitting the clients also increases. To quantify the increase in waiting time, assume that a server attempts to assign the new client to groups in a round-robin manner starting with g_{new}. Thus, a client will be delayed by i rounds if i consecutive groups are full while the $(i+1)^{th}$ group is not. In general, if k out of the D groups are full and if the retrieval of blocks for a new client is equally likely to begin from any disk in the array, then the probability that a client has to wait for i rounds before being admitted is given by:

$$W(i,k) = \frac{k!(D-i)!}{(k-i)!D!} * \left(1 - \frac{k-i}{D-i}\right)$$

Thus, the expected number of rounds that a client gets delayed when k groups are full is given by: $\sum_{i=1}^{i=k} i * W(i,k)$.

In addition to the increase in waiting time at high system load, the key limitations of the variable-size block placement policy include: (1) the inherent complexity of allocating and deallocating variable-size media blocks, and (2) the higher implementation overheads. Thus, although the variable-size block placement policy is highly attractive for designing multimedia storage servers for predominantly read-only environments (e.g., video on-demand), it may not be viable for the design of integrated multimedia file systems (in which multimedia documents are created, edited, and destroyed very frequently).

4 Analysis and Evaluation

In this paper, we have presented fixed- and variable-size block placement policies, and have proposed a model for deriving a range of media block sizes which reduce the occurrence of transient playback discontinuities. This model can also be utilized to characterize the effects of varying system configuration parameters (such as number of clients, data requirements of each client, number of disks in the array, and the disk performance characteristics) on the optimal range of media block sizes. In what follows, we analyze their dependence and then empirically evaluate our model.

4.1 Dependence of Media Block Size on System Parameters

- *Dependence on number of clients*: For small number of clients, to minimize load imbalance, media block size should be chosen such that each client accesses almost all the disks in the array during every round. However, at small media block sizes, a significant fraction of the service time may be due to seek time and rotational latency overheads. Hence, to service a larger number of clients, the server should minimize the fractional overhead due to seek time and rotational

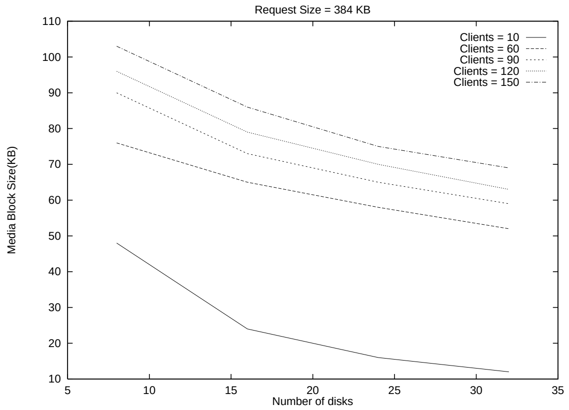

Figure 4 : Effect of variation in number of disks in the array on M_{low}

Disk capacity	2 GBytes
Number of disks in the array	16
Bytes per sector	512 KB
Sector per track	99
Tracks per cylinder	21
Cylinders per disk	2627
Minimum seek time	1.7 ms
Maximum seek time	22.5 ms
Maximum rotational latency	11.1 ms

Table 1 : Disk Parameters of Seagate-Elite3 disk used in the paper

latency by selecting larger media block sizes. Figure 2 depicts the effect of increasing the number of clients on the media block size.

- *Dependence on data rate requirement*: With increase in the data rate requirements of the clients, to effectively utilize the disk array, the server can: (1) increase media block size while maintaining the number of disks accessed by clients during each round at a constant value, or (2) increase the number of disks accessed during each round, or both. The optimal choice, however, may depend on the relative gains in performance (measured in terms of the service time of the most heavily loaded disk) yielded by above two approaches. Figure 3 illustrates this variation.

- *Dependence on disk array characteristics*: The choice of media block size may depend upon the number of disks in the array as well as the performance characteristics of each disk.

  1. Effects of increasing number of disks in the array: To utilize the increase of data transfer rate yielded by an increase in the number of disks in the array, the server must partition media streams into smaller size media blocks. Figure 4 demonstrates this trend.
  2. Effects of enhancing disk performance characteristics: The choice of media block size depends on the ratio of seek time and rotational latency overhead incurred while accessing a block to the actual data transfer time. Thus, when the track density is increased, the server can access larger amount of data for each client from disks without affecting the service times. Hence, increase in track density is directly reflected as an increase in the media block sizes. On the other hand, increasing rotational speed decreases the rotational latency (and hence, the overhead per block) as well as the data transfer time. Hence, the corresponding increase in the ratio of overhead to data transfer time is smaller than the scenario in which track density is increased. Hence, the rate of increase in media block size is smaller. Figure 5 depicts this variation.

Thus, in addition to enabling a server designer to determine a media block size that is best suited for a given environment, the proposed model makes it feasible for the designer to appropriately configure the server (i.e., determine the size and performance of the disk array) so as to meet the requirements imposed by the target environment.

4.2 Empirical Evaluation

To validate our models as well as to compare the two placement policies, we carried out extensive trace-driven simulations in an environment consisting of an asynchronous disk array with 16 disks. Each media stream is assumed to be interleaved across the entire array. The characteristics of each disk are shown in Table 1. Clients are assumed to arrive at the beginning of the simulation and remain in the system till the end of the simulation. The video stream accessed by clients are assumed to be independent and encoded using a Variable Bit Rate (VBR) compression technique. The conventional SCAN disk scheduling algorithm is employed for retrieving media blocks from a disk during each round. The playback rate of each video stream is assumed to be 30 frames/sec with an average data rate requirement of 1.2 Mb/sec.

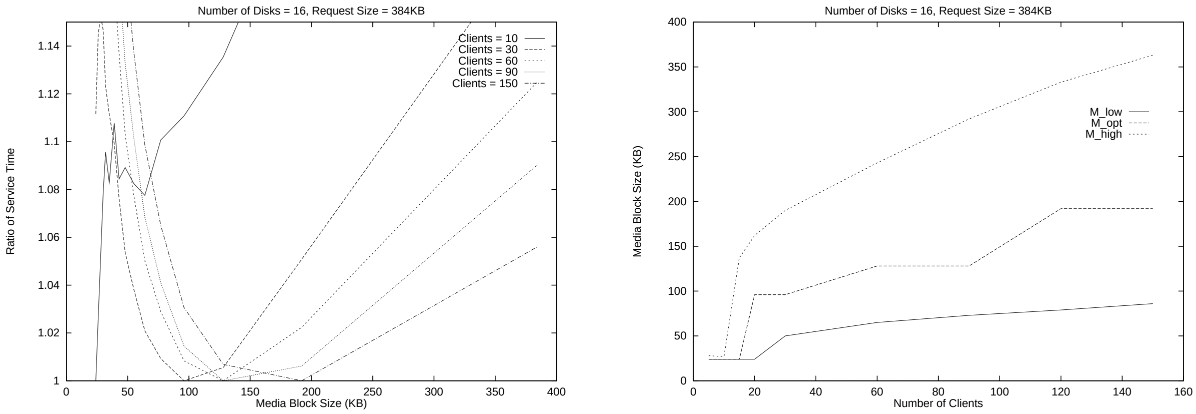

Figure 2 : Effect of varying the number of clients on M_{low} , M_{opt} , M_{high}

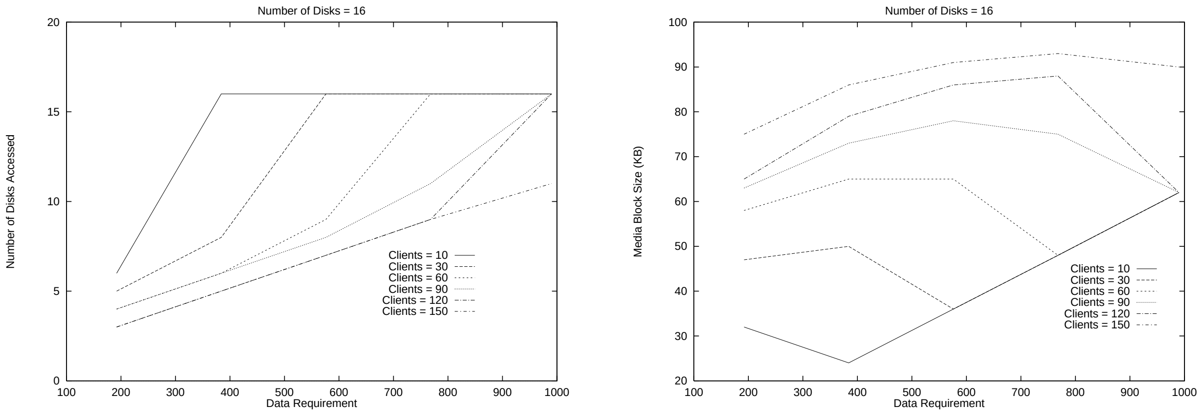

Figure 3 : Effect of variation in data requirement on M_{low}

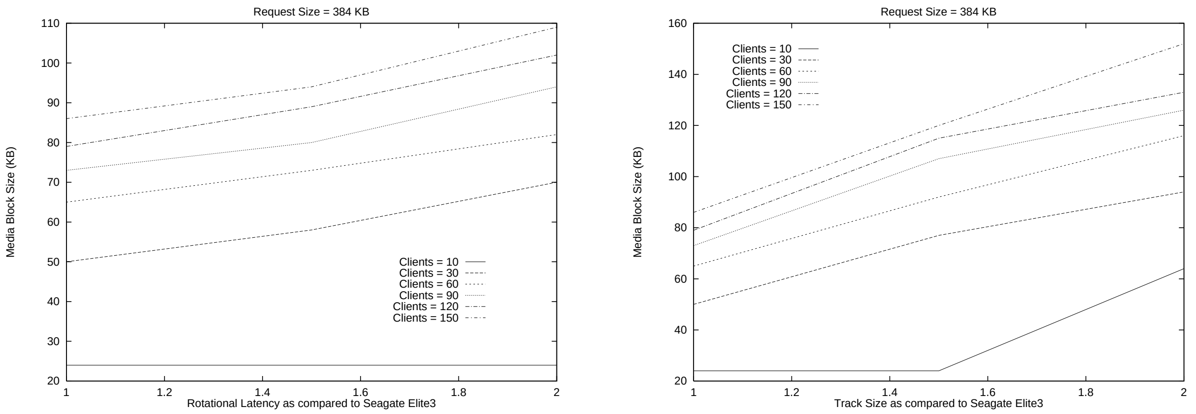

Figure 5 : Effect of disk characteristics on M_{low}

Parameter	Lower	Optimal	Upper
Number of clients	$< 5\%$	$< 2\%$	5-7%
Data requirement	$5 - 7\%$	$< 2\%$	1-7%
Disk characteristics	$4 - 7\%$	$< 2\%$	0 - 4%

Table 2 : Comparison of the service times yielded by M_{low}, M_{opt} and M_{high} media block sizes with empirically observed minimum value

4.2.1 Validation of Analytical Model

As a first step towards validating the analytical model for determining media block size, we empirically measured: (1) service time as a function of the number of blocks being accessed, and (2) the number of blocks accessed from the most heavily loaded disk during each round. Our experiments demonstrate that the empirically derived service times closely match the analytical prediction. Furthermore, since the model employs worst-case assumptions regarding the number of blocks requested by each client, the empirically observed values for the number of blocks accessed from the most heavily loaded disk are slightly smaller (within 15%) than their analytical predictions.

We have also empirically evaluated the effectiveness of the range of media block sizes predicted by the model. To do so, we measured service times yielded by the most heavily loaded disks over a large number of rounds for a wide range of media block sizes. That is, for each media block size M_k, we measured the service times (denoted by $\widehat{\tau^i_{\max}}(M_k), i \in [1, R]$) of the most heavily loaded disk for R rounds, and computed:

$$\widehat{\tau^{avg}_{\max}}(M_k) = \frac{\sum_{i=1}^{R} \widehat{\tau^i_{\max}}(M_k)}{R}$$

The above procedure was repeated for various media block sizes including M_{low}, M_{opt} and M_{high}, which denote the analytically derived lower bound, optimal and upper bound on media block sizes. Table 2 demonstrates that the values of $\widehat{\tau^{avg}_{\max}}$ yielded by M_{low}, M_{opt} and M_{high} are within 2%, 7% and 7% of the minimum of the $\widehat{\tau^{avg}_{\max}}$ derived for various values of M_k.

4.2.2 Comparison of Fixed and Variable-size Block Placement Policies

We compared the performance of fixed- and variable-size block placement policies in terms of: (1) the maximum number of clients that can be supported without violating the continuity requirements of any client, and (2) their average buffer space requirement. Table 3 illustrates that, due to the variability (induced by VBR compression techniques) in the number of blocks accessed by a client, the fixed-size block placement policy without any read-ahead (see Section 2.2) can support a much smaller number of clients as compared to the variable-size block placement policy. However, exploiting the sequentiality of video playback and reading ahead media blocks enables the server to smooth out transient overloads. Consequently, a server employing fixed-size block placement policy in conjunction with controlled read-ahead techniques can service as many clients as a server that uses variable-size block placement policy. With respect to buffer space, fixed-size block placement policy requires slightly larger buffer space as compared to its counterpart. This is because, the set of fixed-size media blocks accessed during a round generally contain slightly larger amount of data than what is consumed during that round. Hence, the server must provide sufficient buffer space to maintain such additional data across round boundaries.

Policy	Number of Clients supported
Fixed, No Read Ahead	180
Fixed, Read Ahead	240
Variable	240

Table 3 : Number of Clients supported

5 Concluding Remarks

In this paper, we have presented fixed-size and variable-size block placement policies for storing multimedia objects on disk arrays. For the fixed-size block placement policy, we have presented an analytical model for determining an optimal range of media block sizes for a given set of system configuration parameters. Using the model, we have characterized the effects of varying various system parameters (such as number of clients, data rate requirement of clients, as well as disk array characteristics) on the media block sizes. For variable-size block placement policy, we have presented a method for distributing the workload amongst all the disks in the array. The analytical predictions were validated using extensive trace driven simulation.

Our experiments have demonstrated that a multimedia server that employs fixed-size block placement policy in conjunction with read-ahead techniques can service as many clients as a server employing the variable-size block placement policy. Whereas the variable-size block placement increases the complexity of storage space management, the fixed-size policy increases the buffer space requirements. Therefore, variable-size block placement policy is attractive for designing multimedia storage servers for predominantly read-only environments where storage space management complexity is not an issue (e.g., video on-demand). However, environments which involve frequent creation, modification, and deletion of multimedia objects (e.g., integrated multimedia file system) favor fixed-size block placement policy. A multimedia server prototype based on these placement policies is currently being implemented at the Distributed Multimedia Computing Laboratory at UT Austin.

REFERENCES

[1] E. Chang and A. Zakhor. Scalable Video Placement on Parallel Disk Arrays. In *Proceedings of IS&T/SPIE International Symposium on Electronic Imaging: Science and Technology, San Jose*, February 1994.

[2] P. Chen and D. Patterson. Maximizing Performance in a Striped Disk Array. *Proceedings of ACM SIGARCH Conference on Computer Architecture, Seattle, WA*, pages 322–331, May 1990.

[3] H. Garcia-Molina and K. Salem. Disk Stripping. *International Conference on Data Engineering*, pages 336–342, February 1986.

[4] R.H. Katz, G. Gibson, and D. Patterson. Disk System Architectures for High Performance Computing. *Proceedings of the IEEE*, 77(12):1842–1858, December 1989.

[5] K. Keeton and R. Katz. The Evaluation of Video Layout Strategies on a High-Bandwidth File Server. In *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'93), Lancaster, UK*, November 1993.

[6] M. Kim. Synchronized Disk Interleaving. *IEEE Transactions on Computers*, C-35(11):978–988, November 1986.

[7] E.K. Lee and R.H. Katz. An Analytic Performance Model for Disk Arrays. In *Proceedings of the 1993 ACM SIGMETRICS*, pages 98–109, May 1993.

[8] M. Linvy, S. Khoshafian, and H. Boral. Multi-Disk Management Algorithms. *Proceedings of Eleventh Symposium on Operating Systems Principles*, pages 69–76, November 1987.

[9] M. Rabbani and P. Jones. Digital Image Compression Techniques. *SPIE Optical Engineering Press*, 1991.

[10] F.A. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming RAID: A Disk Storage System for Video and Audio Files. In *Proceedings of ACM Multimedia'93, Anaheim, CA*, pages 393–400, August 1993.

[11] H. M. Vin, A. Goyal, and P. Goyal. Algorithms for Designing Large-Scale Multimedia Servers. *To appear in Computer Communications*, March 1995.

[12] H. M. Vin, P. Goyal, A. Goyal, and A. Goyal. A Statistical Admission Control Algorithm for Multimedia Servers. In *Proceedings of the ACM Multimedia'94, San Francisco*, pages 33–40, October 1994.

A Derivation of bound on $F_{X_{\max}}(x)$

Consider two dependent random variables X, Y and define $Z = \max(X, Y)$. Our objective is to determine $P(Z > t)$. Note that $P(Z > t)$ is at least the probability that $X > t$ or $Y > t$ and is at most the sum of the individual probabilities. In other words,

$$\max(P(X > t), P(Y > t)) \leq P(Z > t)$$
$$\leq \min(1, P(X > t) + P(Y > t))$$

Rewriting,

$$\max(1 - P(X \leq t), 1 - P(Y \leq t)) \leq 1 - P(Z \leq t)$$
$$\leq \min(1, 2 - P(X \leq t) - P(Y \leq t))$$

Equivalently,

$$1 - \min(P(X \leq t), P(Y \leq t)) \leq 1 - P(Z \leq t)$$
$$\leq \min(1, 2 - P(X \leq t) - P(Y \leq t))$$

Subtracting 1 and rewriting,

$$\min(P(X \leq t), P(Y \leq t)) \geq P(Z \leq t)$$
$$\geq \max(0, P(X \leq t) + P(Y \leq t) - 1)$$

Alternately,

$$\max(0, F_X(t) + F_Y(t) - 1) \leq F_Z(t)$$
$$\leq \min(F_X(t), F_Y(t))$$

In our case, we have D such random variables (1 corresponding to each disk) and since all are identical, we get the desired result.

$$\max(0, D * F_{X_j}(x) - (D-1)) \quad \leq F_{X_{\max}}(x) \leq \quad F_{X_j}(x)$$