Data Base Design Principles for Striping and Placement of Delay-Sensitive Data on Disks

Stavros Christodoulakis *

Laboratory of Distributed Multimedia Information Systems and Applications (MUSIC),

Technical University of Crete (TUC), P.O. Box 133, Chania 73100, Greece (http://www.music.tuc.gr)

stavros@ced.tuc.gr

Fenia A. Zioga*

Laboratory of Distributed Multimedia Information Systems and Applications (MUSIC),

Technical University of Crete (TUC), P.O. Box 133, Chania 73100, Greece (http://www.music.tuc.gr)

fenia@ced.tuc.gr

Abstract

We investigate design principles for placing striped delay-sensitive data on a number of disks in a distributed environment. The cost formulas allow us to calculate the maximum number of users that can be supported by n disks and the minimum number of disks needed to support k users, as well as to study the impact of other performance tuning options. Next, we examine the problem of optimal placement for striped data. We show that for fixed probabilities of accessing the delay-sensitive objects, partitioning the set of disks is always better than striping in all of the disks. Then, given a number n of disks and r distinct delay-sensitive objects with probabilities of access $p_1, p_2, \ldots, p_r$, that must be striped across r different disk partitions (i.e., non-overlapping subsets of the n disks), we use the Majorization theory and the theory of Schur functions in order to find what is the optimal number of disks that must be allocated to each partition. We analyze the problem of grouping the more and less popular delay-sensitive objects together in partitions when the partitions are less than the objects, so that the number of supported users is maximized. Finally, we present the trade-off of striping on all the disks versus partitioning the set of the disks when the access probabilities of the delay-sensitive objects change with time.

1 Introduction

A placement policy frequently applied in large scale storage systems is data striping across multiple disks. Such parallel placement schemes are also frequently employed in order to service without interruptions numerous concurrent delay-sensitive streams by exploiting the aggregate bandwidth of multiple disks. Recent empirical studies compare the trade-offs of alternative striping schemes as well as duplication policies and combine strategies to support large numbers of concurrent streams.

Several parameters such as the number of the disks on which striping is applied, the size of the striping unit, the consumption rate of each stream, and the performance characteristics of the disks such as the seek, the rotational delays and the disk transfer rate significantly influence the performance of the stream service. Some of these parameters, such as the size of the striping unit and the number of the disks used for striping vary among different placement strategies.

In this paper, we employ a model to investigate the data base design principles for striping and placement of delay-sensitive objects on disks. The paper is organized as follows: section (2) describes the relevant research that has been carried out and the contributions of this paper to the issue of striping. Section (3) presents the mathematical tools that will be used in the remainder of the paper. In section (4), we describe our basic model which is used so as to derive the maximum number of the users that can be supported by a given number of disks and the minimum number of the disks that are needed for supporting a given number of users. In section (5), given two delay-sensitive objects with different access probabilities that need to be striped we examine the problem of splitting (allocating) the system's disks into two partitions (one partition for each striped delay-sensitive object) and provide an optimal allocation algorithm that serves the maximum number of concurrent streams possible at all times. In section (6), we examine the generalized problem of disk allocation to r partitions ($r>2$), one for each of the r striped delay-sensitive objects. Section (7) includes a discussion of alternative schemes for grouping together a number of objects with different access probabilities into the same partition. In section (8), the optimal partitioned striping scheme is compared to the unpartitioned scheme when the probabilities of the delay-sensitive objects have a variance (i.e., the probabilities change in time). Finally, section (9) summarizes our results and contributions.

2 Related Work and Contributions

The notion of disk striping was introduced in the late 1980's by K. Salem and H. Garcia-Mollina, who presented the motivations, the benefits as well as the potential implications that the employment of striping policies might have. In their work ([16]), they evaluated striping in terms of improved efficiency due to the parallel I/O capability and in terms of

*Support for this work was provided by the European Community through the ESPRIT Long Term Research Project HERMES no. 9141.

reliability in the case of disk failures. In [9], the tradeoffs for supporting delay-sensitive data are studied analytically while most of the work in [10] considers the buffer size of the server to be a serious bottleneck. In [18], some useful concepts are presented for the utilization of disk arrays for striping and continuously retrieving large video files at a predefined rate. The disk access policy that is presented by the authors aims also at increasing the system's throughput (i.e., the number of the simultaneous video streams supported). Memory management and start-up latency issues as well as alternatives to improve efficiency in terms of memory management are addressed. In [1], an efficient striping policy (staggered striping) is presented, which is suitable for striping delay-sensitive objects with different consumption rates across overlapping subgroups of the initial set of disks. Continuity of retrieval is guaranteed while the bandwidth of all disks is fully exploited at all times. Chen and Patterson ([2]) accomplish maximization of the performance of a striped disk array with respect to the size of the striping unit (i.e., the amount of logically continuous data on each disk). The determination of the optimal stripe width depends on the system load at a given time and on the average time needed for retrieving the striped data (positioning and transfer delays). In [19], the optimal number of striping disks is estimated and is shown to be dependent on the multiprogramming level. Özden, Rastogi, and Silberschatz ([14]) present a placement scheme that stores video files across many disks in order to enable the parallel retrieval of each video file. An alternative placement scheme for variable-rate videos is developed and memory management issues are addressed.

Traditionally, data base design research has been based on rough deterministic models since the objective is to compare design decisions, not to accurately predict response times. In contrast, research in data base performance prediction considers probabilistic models. In [17] the authors use a probabilistic model for the cost of accessing data from the disk. Based on this model they express the cost of a service to be interrupted. This allows more users in the system than assuming conservative worst-case upper bounds for the disk cost. However, especially in a distributed environment the disk cost is not the only source of quality of service variations and many other factors like disk cache and caching policies, main memory allocation overheads, CPU congestion, and network congestion may result in such variations making the problem of accurate performance prediction very difficult. In this paper, we adopt a deterministic model since our objective is only to compare alternative strategies for design purposes and not to give an accurate prediction of performance in the presence of such variations. Once a design strategy has been selected the maximum throughput that can be supported by the system could be evaluated on the real system using experimentation.

The quality of service support of the overall system in a distributed environment is a problem which is examined by several research communities, not only the database community. Several engineers believe that the problems of variations in the quality of service will be solved in actual implementations with resource over-allocation after experimentation by the system designers.

Merchant and Yu ([12]) consider the issue of duplicating a striped file across the system disks not only for reliability but also for efficiency improvement reasons. They propose a scheme that utilizes different stripe widths for the two striped versions and they prove its superiority compared to applying the same width to both replicas. In [8], the relative access probabilities of the video files are taken into consideration in the determination of a replication policy of the "hottest parts" so that load-balancing is achieved.

In [15] Özden, Rastogi and Silberschatz present an evaluation of two alternative schemes. The first is called fine-grained striping and performs parallel retrieval of data from all the disks in each round of service in a manner similar to RAID-3. The second which is called coarse-grained striping and is similar to [1] phases out the retrieval so that in each round a different disk is involved as in the RAID-5 architecture. The authors show that coarse-grained striping can provide higher throughput than fine-grained striping.

In [3] Gemmel evaluates fine-grained striping, coarse-grained striping and replication of data, as well as combinations of the above in terms of system design and implementation in order to provide a system with high throughput and functionality in a distributed environment. He also observes that when main memory size is fixed, the coarse-grained striping can provide higher throughput. However, in a distributed environment where the clocks may not be synchronized the coarse-grained striping loses its main memory size advantage. In addition, he observes that for highly interactive applications which may stop and resume, or go forward or backward and resume the interactive response time of the system to requests can be immediate with the fine-grained striping, while with coarse-grained striping the start-up latency may be $d-1$ rounds, where d is the number of the striping disks. For environments where the interactive response time (start-up latency) is important, this is a major limitation of the coarse-grained striping. In these environments the balancing of disk loads for fast forward etc. is easier to guarantee with fine-grained striping. Gemmel also argues that combinations of the coarse-grained striping with fine-grained striping and/or with replication do not make much sense. However, combinations of fine-grained striping in a limited number of disks with replication may be profitable.

In this paper, we consider a model for striping delay-sensitive data in order to compare alternative designs and develop design principles for placing striped delay-sensitive data on a number of striping disks. Our model is based on fine-grained striping and is suited for interactive environments. In the distributed environment of our model each client (or sets of clients) has enough buffer space B (main and/or secondary memory) for accepting the data delivered by the server and the server is equipped with buffer space B_s to allow prefetching for the supported clients. In this setting the buffer size of the client is not necessarily the system bottleneck. Our approach thus is more related to the conceptual analysis of Gemmel. We develop an analytic formula that allows us to study the performance of striping. Based on the properties of the formula we show that the ratio of the length of the data B retrieved for a client per round of service of all the clients divided by the disk overhead for accessing the data (seek S and rotational delay R) is an important performance parameter. When this ratio is large the number of the streams that can be supported is proportional to the number of the disks used in striping. However, when the ratio $\frac{B}{(S+R)}$ is finite then as the number of the striping disks increases the number of streams that can be supported is bounded by $\frac{B}{C(S+R)}$ where C is the consumption rate of each stream. We study various methods for increasing the number of users that can be supported including the use of secondary storage for prefetching in the client stations. Our model allows the system designer to understand the various methods that can be used to improve the performance of the system.

We then study the problem of partitioning the system disks in order to stripe a number of delay-sensitive objects across each partition so that the maximum number of users possible is supported. The delay-sensitive objects may have different access probabilities (and therefore a different number of disks should be allocated to each object) and/or they may have different consumption rates. We show that a good partitioning of the disks can support more users than striping all the delay-sensitive objects in all the disks. It is clear that in order to maximize the number of concurrent streams in the system at any point in time, more disks should be allocated to the highest probability objects. Finding however the optimal number of disks that have to be allocated to each delay-sensitive object is a combinatorial problem. We prove that there is a simple algorithm to find the optimal number of striping disks for each different delay-sensitive object. A closed form formula is derived for the case that the number of the incoming requests is not too high and hence all the user requests can be served by a partitioning scheme. For the cases that the system is overloaded and not all the requests can be served optimal placement algorithms are provided that support the maximum number of users possible. The optimality proofs are based on properties of Schur concave functions.

A different problem is then examined in which the number of the partitions is less than the number of the delay-sensitive objects and therefore more than one object must be striped per partition. The choice of the particular objects that must be grouped together in the same partition depends on their access probabilities and we prove by using properties of Schur concave functions that the optimal solution must guarantee that the aggregate partition probabilities are as uniform as possible in the Majorization sense. We also derive an allocation algorithm for the case that the delay-sensitive objects have different probabilities and different consumption rates.

Finally, we compare striping different objects in different partitions of the set of disks versus striping all the objects across all the disks in the case that the probabilities of the delay-sensitive objects have a variance and we describe the tradeoffs involved.

3 Mathematical Tools

Majorization Theory and Schur Functions

Let two decreasing probability vectors $\vec{\mathbf{p}} = (p_1, \ldots, p_n)$ and $\vec{\mathbf{q}} = (q_1, \ldots, q_n)$. We say that $\vec{\mathbf{p}}$ majorizes $\vec{\mathbf{q}}$, (symbolized as $\vec{\mathbf{q}} \prec \vec{\mathbf{p}}$), if $\sum_{i=1}^{j} p_i \geq \sum_{i=1}^{j} q_i$ for all $j < n$ and $\sum_{i=1}^{n} p_i = \sum_{i=1}^{n} q_i$. Intuitively, $\vec{\mathbf{p}}$ majorizes $\vec{\mathbf{q}}$ if $\vec{\mathbf{p}}$ is "more skewed vector" than $\vec{\mathbf{q}}$.

Theorem 1 (Schur)
If a function $\phi: R^n \rightarrow R$ is **Schur convex** *(or Schur* **increasing***) and $\vec{\mathbf{p}}$ majorizes $\vec{\mathbf{q}}$, then $\phi(\vec{\mathbf{p}}) \geq \phi(\vec{\mathbf{q}})$. If a function $\phi: R^n \rightarrow R$ is* **Schur concave** *(or Schur* **decreasing***) and $\vec{\mathbf{p}}$ majorizes $\vec{\mathbf{q}}$, then $\phi(\vec{\mathbf{p}}) \leq \phi(\vec{\mathbf{q}})$.*

A necessary and sufficient condition for a continuously differentiable function $\phi : R^n \rightarrow R$ to be Schur convex (concave) is that for any vector $\vec{\mathbf{x}}$ of R^n and for all $i \neq j$:

$$(x_i - x_j)\left(\frac{\partial \phi}{\partial x_i} - \frac{\partial \phi}{\partial x_j}\right) \geq (\leq) 0$$

where each x_i is a component of the probability density vector $\vec{\mathbf{x}} = (x_1, \ldots, x_n)$.

The proof of this theorem and many of its applications are described in ([11]). Using this theorem we can compare the values of (Schur) functions of vectors for more uniform or more skewed vectors.

4 The Basic Model

In this section we describe a simple model which will allow us to study the impact and compare the design alternatives when striping delay-sensitive data across a number of secondary storage devices. We choose this simple model because our purpose is the comparison of the design alternatives in certain common, well-understood environments and the derivation of design principles based on the parameters incorporated in the model. Our purpose is not the derivation of an analytic performance prediction model which would probably be very difficult to achieve due to the complexity of the actual systems as well as the numerous and complex interdependencies of the parameters.

We assume that we have a server environment with a number of secondary storage devices (disks) which have the same performance characteristics (data transfer rate, worst case seek time, worst case rotational delay). We stripe delay-sensitive data on these disks, in order to support multiple concurrent streams of the striped data. The stream service proceeds with a series of successive service rounds. A **service round** is the time interval during which all k concurrent streams are serviced one time in a round-robin fashion by retrieving a total of $k\,B$ bytes of data where B denotes the buffering capacity of each client. The amount of data that is fetched for a particular stream during a service round is called the **round block size** for that stream and the data itself is called the **round block** of the stream. The **default round block size** is the buffer capacity B at each client's workstation. The data are striped across the disks so that for any stream the associated load during a service round (i.e., the amount of data that is retrieved for that stream during a service round) is balanced among all disks. Moreover, the buses on which the secondary storage devices are placed have enough bandwidth and produce no queuing delays.

Users are accessing the data from workstations across a network. The communication bandwidth is large enough to pass all the bandwidth that the disks of the server deliver (otherwise the clients can not be supported with this communication infrastructure). In addition no delays are observed in the server when the data is sent to the workstations. We assume that every client has a large buffer of size B (bytes) for accepting all the data that is delivered to it during a service round. For the buffer allocation in the server two cases are considered. First, we consider the case that the server is equipped with the minimum buffer space that suffices to support the retrieval of B bytes per stream. Secondly, we examine the alternative that extra buffer space may be allocated in the server for prefetching purposes (see for example [13]), under the assumption that the server offers high enough transfer rate to allow prefetching. During the consumption of the data that has been sent to a client the server is serving other clients. No two clients require the same piece of data at precisely the same time. The server is finishing the production of data for all the users in less time than the time needed for the consumption of data in one user station. The consumption rate of the delay-sensitive data in the client is constant and equal among the clients of the system. Moreover, there are no requests that increase the bandwidth demand from a single user (like fast forward). Such requests may be served like simple retrieval requests if there is enough available bandwidth to do so, or they can be aided by supporting structures that exist in different disks as is done for example in the KYDONIA multimedia information server ([6]). In some cases the workstations of the clients are equipped with secondary storage space so that data can be prefetched there before the playback starts, and the retrieval requirements in real time are less.

This simple model can be generalized to capture the principles of striping in several different environments. Some generalizations of the model are introduced later on in the paper. Further generalizations and study of parameter variances and interdependencies in actual systems (such as limited or variable communication bandwidth) are currently under experimental investigation in a testbed environment supported by KYDONIA ([6]).
Table 1 describes the parameters of the problem.

X	a random round block size (bytes).
n	number of disks that are used for striping.
C	consumption rate of the delay-sensitive objects in the system (bytes/sec).
T	transfer rate of the disks (bytes/sec).
S	worst case delay due to seek to a desired disk location (sec).
R	worst case delay for the disk to rotate to a desired disk sector (sec).
k	number of concurrent streams in the system.
B	round block size (bytes) for maximum number of users and no prefetching.

Table 1: The parameters of our system.

The following is a basic equation which expresses the fact that the consumption time for the data in a user's buffer has to be larger than the time needed for a disk to produce the data for all concurrent users when no prefetching has taken place, so that no user experiences hiccups:

$$\frac{X}{C} \geq k \cdot (S+R) + k \cdot \frac{X}{nT}$$

where X is the round block size for each stream. The left hand side of this equation is the time needed by a user to consume the data delivered by the server in one service round. The right hand side is the time needed by the server to produce all the data of a service round i.e., to produce the round block for all the streams. From the above equation, it follows that the number of users that can be concurrently serviced is:

$$k \leq \frac{X}{C \cdot (S+R+\frac{X}{nT})}$$

The round block size can be increased in order to maximize the number of streams k that can be supported concurrently by the system. However, it can not exceed B, i.e., the buffering capacity in each client's workstation. Therefore, the *capacity $S(n)$ of the partition which consists of the n striping disks* i.e., the maximum number of streams that can be supported by striping across the n disks is:

$$\begin{aligned} S(n) = max(k) &= \frac{B}{C} \cdot \frac{1}{S+R+\frac{B}{nT}} \\ &= \frac{1}{C} \cdot \left(\frac{B}{S+R}\right) \cdot \frac{1}{1+\left(\frac{B}{S+R}\right)\frac{1}{nT}} \quad (1) \end{aligned}$$

In the following we study the impact that various striping parameters have on its performance. The parameters that we examine are: the number of the striping disks, the existence of prefetching in the server or in the client, the buffer space in the client and the bandwidth of the striping disks. In the following we provide a short description of this study. The complete analysis and the intuitive explanation of the impact of each parameter are omitted due to space reasons but can be found in [5].

- **The Impact of the Number of the Striping Disks.** From equation (1) it can be seen that when the number of the striping disks increases but the ratio $\frac{B}{S+R}$ is finite (which happens when there is a finite round block size or finite length of buffer at each client's workstation), then the system capacity $S(n)$ converges to $\frac{B}{C(S+R)}$:
$$\lim_{n\to\infty} S(n) = \frac{1}{C} \cdot \frac{B}{S+R}$$
That is, for given disk characteristics, stream consumption rate and finite round block size B for a stream there is an upper bound of the system capacity that can be achieved, regardless of the number of the striping disks.

- **The Impact of Prefetching in the Server.** We assume that the server is equipped with main memory for prefetching purposes and high enough transfer rate in order to allow data prefetching for some or all of the supported streams, i.e., allow these streams to retrieve more than B bytes which is the default round block size. Essentially, prefetching for some streams means that less streams than k_1 will be concurrently supported. However, the total time that the striping disks are occupied in a service round is smaller with prefetching. This means that after the end of a service round a free time period intervenes until the beginning of the next service round. This free time per round can be used to support small non delay-sensitive requests.

- **The Impact of the Buffer Size of the Client.** From equation (1) it follows that
$$\lim_{B/(S+R)\to\infty} S(n) = \frac{nT}{C}$$
This implies that when the round block size (or the size of the client buffer) is very large in comparison to the positioning overhead, then the number of the streams that can be supported is proportional to the number of the striping disks. Equation (1) can be written as:
$$S(n) = \frac{nT}{C} \cdot \frac{B/T}{B/T + n(S+R)}$$
The maximum number of streams that the system can support is the percentage of the useful time per stream during a service round multiplied by the maximum number of the streams that the system could support if there were no positioning overheads.

- **The Impact of the Bandwidth of the Striping Disks.** When $nT >> \frac{B}{(S+R)}$ i.e., the aggregate bandwidth of all striping disks is significantly bigger than the ratio of the round block size for each stream over the associated positioning delays ($\frac{B}{S+R}$), then the system capacity converges to $\frac{B}{C(S+R)}$, i.e.,

$$\lim_{T \to \infty} (S(n)) = \frac{1}{C} \frac{B}{S+R}$$

- **Data Prefetching in the Client.** An alternative system design which can increase the number of the streams supported involves prefetching of data at the workstations of the clients, provided that the clients are equipped with sufficient secondary storage for storing the prefetched data. The prefetched data may even reside in main memory if the size of the main memory of the client is significant in comparison to the length of the delay-sensitive object so that a significant proportion of the object is prefetched.

  The retrieval of a striped delay-sensitive object with prefetching at the client proceeds as follows: Firstly, the system does an initial prefetching in order to fill the prefetching buffer at the client before the stream playback begins. This initialization phase introduces a delay in the service of the stream which is perceived as start-up delay from the user's point of view. The longer the prefetching size the more concurrent clients the system can support but the longer the start-up delay will be. When the initialization interval has completed the stream playback begins. At the same time that the prefetched data are retrieved from the client disks in order to be consumed by the client new data are delivered from the server and placed on the client disks. These two actions that have to be performed concurrently by the clients' disks raise the bandwidth demand at the client.

  Let L be the length of the delay-sensitive object (in bytes) and L' be the length of the secondary storage buffer at the client available for prefetching (i.e., L' is the amount of data prefetched before the playback begins). If C' denotes the data production rate that the server is required to provide at real time in order to guarantee hiccup-free stream playback, then:

$$C' \geq C \frac{L - L'}{L} \qquad (2)$$

  Equation (2) shows that when secondary storage is available in the client for prefetching, i.e., $L' > 0$, then the minimum bandwidth demand C' imposed on the server is smaller in comparison to the original consumption rate C of the stream. Thus, more users can be serviced by the system concurrently at the expense of the buffer cost at the client as well as the start-up delay that the users experience. The precise number of concurrent users that can be supported by the system is:

$$S(n) = \frac{B}{C(\frac{L-L'}{L})} \frac{1}{S + R + \frac{B}{nT}} \qquad (3)$$

  Clearly, when there is little (or no) prefetching at the client (i.e., $L' \to 0$) then no reduction of the server's minimum data production rate is achieved (i.e. $C' = C$) and users experience zero start-up latency. On the contrary, when the secondary storage buffer has size approximately equal to the size L of the delay-sensitive object (i.e., $L' \to L$), then the bandwidth demand imposed on the server, is approximately zero ($C' = 0$)

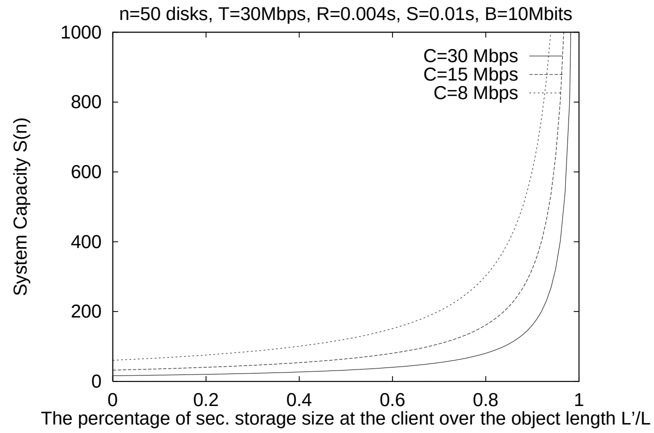

Figure 1: $S(n)$ against the ratio of the prefetching size L' divided by the object size L.

  which results in maximum system capacity while the start-up latency is maximized. The above figure shows the impact that the ratio $\frac{L'}{L}$ of the size of the secondary storage buffer divided by the size of the delay-sensitive object has on the system capacity $S(n)$.

  Formulas that give the start-up delay overhead as well as the bandwidth overhead at the client can be found in [5]. The formulas allow the system designer to fine tune the amount of client prefetching based on the maximum tolerable start-up delay and the available client disks.

- **Estimating the Minimum Number of Disks for Delay-Sensitive Support.** In the case of systems which provide a number of disks dedicated to the service of delay-sensitive requests, a lower bound of the number of disks that are required to support a given request load can be derived from equation (1) by solving for n: $n \geq \frac{kBC}{TB - Tk(S+R)C}$. The minimum number $MIND(k)$ of disks that must be used in order to service exactly k concurrent streams is:

$$MIND(k) = \left\lceil \frac{kBC}{TB - Tk(S+R)C} \right\rceil$$

5 Optimal Striping for Two Delay-Sensitive Objects across Non-Overlapping Disk Partitions

In this section we study the problem of optimal utilization of the system disks when two different delay-sensitive objects must be striped across them. The optimal utilization of the available disks is defined as the one which supports the maximum number of requests for the two striped objects. Essentially, the set of the disks is divided into two non-overlapping groups (partitions) of possibly different lengths and the two delay-sensitive objects are allocated to the two partitions (one object in each partition). However, different partition sizes may result in different retrieval performance. We solve the problem of determining the partition sizes which maximize the number of the user requests for the two streams that can be supported. We also show that such an optimal allocation to the two partitions (whenever feasible) is always better than striping both objects to the set of the n disks.

5.1 Striping in Disjoint Partitions versus Striping in one Partition

The following theorem states that when the probabilities of the two delay-sensitive objects are constant, then striping in disjoint partitions of the disks (partitioned striping) is superior to striping both objects in the whole set of disks (unpartitioned striping).

Theorem 2: *Let two delay-sensitive objects with the same performance characteristics and with fixed access probabilities p_1, p_2 and let n disks which may be used for striping the two objects. Then there is a partition (n_1^*, n_2^*) of the n disks into two sets of disks such that if the first object is striped across n_1 disks and the second object is striped across n_2 disks, then the maximum number of requests k^* that this allocation scheme (optimal partitioned striping) can support is higher than the maximum number of requests that can be supported by striping both objects in the n disks (unpartitioned striping).*

The proof of theorem 2 is based on the fact that $S(n)$ is monotonically increasing and concave with respect to n ([5]).

5.2 Optimal Sizes of Striping Partitions

Consider a set of n disks, each having the same characteristics and two delay-sensitive objects with equal size, but different consumption rates C_1, C_2 and different probabilities of access. The first delay-sensitive object has access probability p_1, while the second delay-sensitive object has access probability $p_2 = 1 - p_1$, bigger than that of the first object. The *probability vector* of the two objects is then: $\vec{p} = (p_1, p_2)$, $p_1 \leq p_2$. Hence, when k requests arrive in the system for the two objects $p_1 k$ of them request the first delay-sensitive object, while $p_2 k$ request the second delay-sensitive object.

The two objects must be striped across the system's disks. We are interested in determining the optimal *allocation vector* $\vec{n} = (n_1, n_2)$ of the n disks into two partitions r_1 and r_2 containing n_1 and n_2 disks respectively. For the allocation vector it must hold that: $n = n_1 + n_2$. We define the *service ratio of the system* to be the fraction of the maximum number of delay-sensitive objects that can be serviced over the total number of requests submitted to the system. Then, the optimal allocation scheme must be such that when the first delay-sensitive object is striped across the n_1 disks of partition r_1 and the second delay-sensitive object is striped across the n_2 disks of partition r_2, a maximum service ratio is accomplished by the system.

Recall that the system's capacity $S(x)$ (where $x > 0$ is the number of the striping disks) denotes the maximum number of the users that can be serviced by the x disks of the system when all requests retrieve the same delay-sensitive object with consumption rate C_i, $i = 1,2$, and assuming that the disk transfer, seek and rotational delays are T, S, R respectively, and the round block size is B_i, $i = 1,2$:

$$S(x) = \frac{B_i}{C_i} \frac{1}{S + R + \frac{B_i}{xT}}$$

Let $S^{(2)}(\vec{n})$ be the maximum number of users that can be serviced by the system when the allocation vector is $\vec{n} = (n_1, n_2)$. Then:

$$\begin{aligned} S^{(2)}(\vec{n}) &= S(n_1) + S(n_2) \\ &= \frac{B_1}{C_1} \frac{1}{S + R + \frac{B_1}{n_1 T}} + \frac{B_2}{C_2} \frac{1}{S + R + \frac{B_2}{n_2 T}} \end{aligned} \quad (4)$$

For a given allocation vector $\vec{n}$, we define the *service ratio of the i^{th} partition* $\lambda_i(n_i)$, to be the fraction of the capacity of the partition over the total number of incoming users which request a delay-sensitive object from the i^{th} partition: $\lambda_i(n_i) = \frac{S(n_i)}{p_i k}$, $i = 1, 2$. When $\lambda_i(n_i)$ is less than one, some users will not be served by the system because the system can not provide the required bandwidth. For a given system load k (i.e., number of requests for the delay-sensitive objects), the objects can be classified as either being *non-congested* or *congested*, if their service ratio is bigger or less than 1, respectively. The partition where an object is striped inherits the congested or non-congested property of the object.

Our placement problem can then be formulated as follows: given the probability vector $\vec{p}(p_1, p_2)$ of two delay-sensitive objects, the total number of the disks n in the system, the disk performance characteristics T, R, S, the consumption rates C_1, C_2 of the two streams, the number (k) of the incoming requests for both objects, and assuming that the round block sizes for each stream are B_1 or B_2 (depending on whether the user requests for the first or the second delay-sensitive object respectively), determine the *optimal allocation vector* $\vec{n}_{opt} = (n_{1,opt}, n_{2,opt})$, for which $n_{1,opt} + n_{2,opt} = n$ holds, i.e., the vector that maximizes the system capacity $S^{(2)}(\vec{n})$. We require the maximum $max\left(S^{(2)}(\vec{n})\right) = max\left(S(n_1) + S(n_2)\right)$ over all possible combinations of n_1, n_2 under the constraint that $n_1 + n_2 = n$.

Figure (2) shows the percentage of serviced requests $\frac{S^{(2)}(\vec{n})}{k}$ for given values of n_1, n_2. We see three distinct sections of the system's service ratio curve. The leftmost section is horizontal and corresponds to the values of k that are small enough so that all $p_1 k$ and $p_2 k$ requests can be served from the n_1 and n_2 disks respectively, and none of the incoming requests is lost. The middle section of the curve, where the percentage of the serviced requests starts dropping below one hundred percent corresponds to the case where one of the two partitions is not big enough to support all requests coming to it, so that some of its requests are not satisfied. Finally, the rightmost section of the curve corresponds to the case where both partitions are not big enough to support all the incoming requests.

- **Determining the Optimal Partitions for Striping Non-congested Delay Sensitive Objects.**

  The maximum number of streams k_1 that can be supported without rejecting any of the incoming requests for streams, is achieved when n_1 and n_2 are such that:

$$\begin{aligned} \mathrm{p}_1 \mathrm{k}_1 &= S(n_1) \\ p_2 k_1 &= S(n_2) \\ n_1 &= n - n_2 \\ p_1 &= 1 - p_2 \end{aligned} \quad (5)$$

  where $S(n_1)$ and $S(n_2)$ are obtained from equation (4). Solving the system (5) we obtain the allocation vector $\vec{n}_{opt} = (n_{1,opt}, n_{2,opt}) = (n_{1,opt}, n - n_{1,opt})$ that must be used to construct the optimal partitions where $n_{1,opt}$ is:

$$n_{1,opt} = \frac{k_1 p_1 C_1 B_1}{T(B_1 - p_1\, k_1\, C_1\, (S + R))} \quad (6)$$

  The solution for k_1 can be found in the Appendix.

  In [5] we provide the optimal placement for the special case of objects with equal consumption rates and

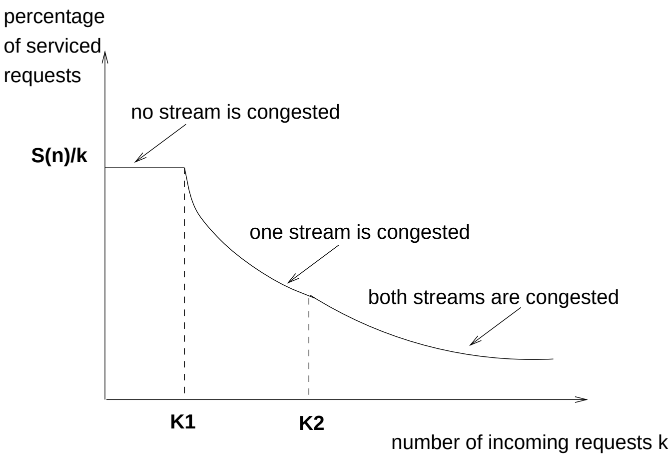

Figure 2: The general shape of the curves representing the service ratio of the system as a function of the number of incoming requests for both delay-sensitive objects, when the two striping partitions are constructed based on a random allocation vector $\vec{n} = (n_1, n_2)$.

retrieval lengths. Furthermore, in the case that the round block sizes B_1, B_2 can be variable, we derive the values of B_1, B_2 that give the maximum system capacity.

- **Determining the Optimal Partitions for Striping Congested Delay Sensitive Objects**

  We have derived the optimal partition sizes when the delay-sensitive objects are congested under the assumption that the delay-sensitive objects are retrieved by the clients at a fixed consumption rate C and that the round block retrieved in a service round for all the clients has the same size B in order to accommodate the playback of any of the two objects. It is clear that an allocation with which the first partition is not congested but the second is i.e., $\lambda_1(n_1) > 1$ and $\lambda_2(n_2) < 1$, can not be optimal since if we transfer some disks from the first partition to the second, then the first partition can remain uncongested (i.e., $\lambda_1({n_1}') \geq 1$, where ${n_1}' < n_1$) while at the same time some of the users of the second partition which could not be served before due to congestion, can be served now by the $n_2+(n_1-{n_1}')$ disks that are allocated to this partition after the transfer of the $n_1-{n_1}'$ disks from the first partition to the second. Therefore, we will only examine allocations in which both partitions are congested i.e., $\lambda_1(n_1) < 1$ and $\lambda_2(n_2) < 1$. In this case, both partitions serve the maximum number of streams that they can support i.e., $S^{(2)}(\vec{n}) = \frac{B}{C}\frac{1}{S+R+\frac{B}{n_1 T}} + \frac{B}{C}\frac{1}{S+R+\frac{B}{n_2 T}}$ where $n_1+n_2=n$. However, one of the two partitions, or both, may not be able to accommodate some of the arriving requests since they are more than the system capacity $S^{(2)}(\vec{n})$. In order to maximize the number of serviced streams $S^{(2)}(\vec{n})$ we use the theory of Majorization and Schur functions and we prove the theorem 3 about the optimization of the capacity $S^{(2)}(\vec{n})$ of the system:

  Theorem 3: *The function of the capacity* $S^{(2)}(\vec{n}) = \frac{B}{C}\frac{1}{S+R+\frac{B}{n_1 T}} + \frac{B}{C}\frac{1}{S+R+\frac{B}{n_2 T}}$ *of a striping system which consists of n disks and is partitioned in two partitions according to the allocation vector $\vec{n} = (n_1, n_2)$ is Schur-concave with respect to $\vec{n} = (n_1, n_2)$.*

  As a consequence of theorem 3, maximization of the number of streams that can be serviced is accomplished by constructing the two partitions according to that particular allocation vector $\vec{n} = (n_{1,opt}, n_{2,opt})$ which is majorized by all other vectors $\vec{n}$, subject to the constraints $p_1 k > S(n_1)$ and $p_2 k > S(n_2)$.

  In [5] we extend these results for delay-sensitive objects with different round block sizes and/or different consumption rates.

Below we present our placement algorithm, which computes the optimal allocation $\vec{n}_{opt}$, given the probability vector of the streams, the number of the striping disks n in the system and the capacity function $S(x)$.

Algorithm 1: Optimal Disk Allocation for Two Delay-Sensitive Objects with Equal Consumption Rates

Input
p_1, p_2: access probabilities of the two delay-sensitive objects, such that $p_1 < p_2$
k : number of requests that have arrived in the system
n : number of disks available for striping

Output
n_1, n_2: the optimal partition sizes for striping

begin
  If $S^{-1}(p_1 k)+S^{-1}(p_2 k)<n$ **then**
    allocate $n_1 = S^{-1}(p_1 k)$ *and* $n_2 = S^{-1}(p_2 k)$
  else
    if $S^{-1}(p_1 k) \geq n/2$ **then**
      allocate $n_1 = n/2$ *and* $n_2 = n/2$
    else
      allocate $n_1 = S^{-1}(p_1 k)$ *and* $n_2 = n - n_1$
end

6 Optimal Striping for R Delay-Sensitive Objects with Equal Consumption Rates across Non-Overlapping Disk Partitions

In this section we extend the placement policy which was presented in section (5) to determine the partitioning of the n striping disks into r partitions, so that r delay-sensitive objects can be striped in the system, one delay-sensitive object per partition. We derive the following optimal placement algorithm which determines the optimal number of disks n_i $(i = 1, \ldots, r)$ that must be allocated to the i^{th} partition, so that when the i^{th} delay-sensitive object is striped across the i^{th} partition $(i = 1, \ldots, r)$, the system capacity is maximized.

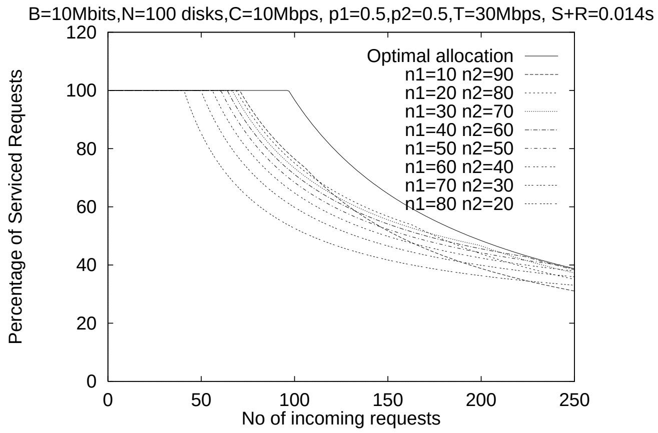

Figure 3: The capacity curve of the optimal partitioning algorithm for two delay-sensitive objects with probabilities 0.5 and 0.5 is drawn together with the capacity curve of various random allocations of disks to the two partitions.

Algorithm 2: Optimal Disk Allocation for Striping r Delay-sensitive Objects with Equal Consumption Rates.

Input
$p_1, \ldots, p_r$: access probabilities of the r objects such that $p_1 < p_2 \ldots < p_r$
k : number of requests that have arrived in the system
n : number of disks available for striping

Output
$n_1, \ldots, n_r$: the optimal partition sizes for striping

begin
if $\sum_{i=1}^{r} p_i k < n$ **then**
allocate $n_i = S^{-1}(p_i k)$ *disks* $(i=1,\ldots,r)$ *to each partition*
else
for $i=1$ **to** r **do**
if $S^{-1}(p_i k) \leq \frac{n}{r}$ **then**
allocate $n_i = S^{-1}(p_i k)$ *disks to the* i^{th} *partition*
else
allocate $n_k = \frac{\left(n - \sum_{j=1}^{i-1} n_j\right)}{r-i+1}$ *disks* $(k=i,\ldots,r)$ *to the remaining partitions*
end

7 Grouping Delay-Sensitive Objects into Partitions

Although striping different delay-sensitive objects across non-overlapping disk partitions was shown to provide higher throughput than unpartitioned striping when the probabilities of the objects are constant it may be desirable to group objects in the same partition in the case that the object probabilities have a variance or when temporal variations are observed during a day or week (see section 8). In this section we study the problem of grouping delay-sensitive data objects in the same partition so that the number of supported streams is maximum. Two cases are considered: in the first case all delay-sensitive objects have the same consumption rate and are classified into two categories with respect to their access probability. In the second case the delay-sensitive objects have different consumption rates and access probabilities.

- **Grouping Delay-Sensitive Objects with Equal Consumption Rates.** Let two groups of delay-sensitive objects with access probabilities p_1, p_2, which will be striped in two disk partitions of n total disks. All delay-sensitive objects have the same consumption rate C. We are seeking the precise grouping scheme of the delay-sensitive objects, so that the total number of serviced users is maximized. The maximum number of concurrent streams $k_1(\vec{p})$ supported can be obtained from the expression for k_1 which is provided in the Appendix by setting $B_1 = B_2 = B$ and $C_1 = C_2 = C$:

$$k_1(\vec{p}) = \frac{nTB(S+R)C + B^2C - \sqrt{const}}{2\,p_1\,p_2\,nT(S+R)^2C^2 + 4\,p_1\,p_2\,B(S+R)C^2}$$

where

$$\begin{aligned} const \;=\; & n^2T^2B^2(S+R)^2C^2 + 2\,nTB^3(S+R)C^2 \\ & + B^4C^2 - 4\,n^2T^2B^2p_1\,p_2\,(S+R)^2C^2 \\ & - 8\,nTB^3p_1\,p_2\,(S+R)C^2 \end{aligned}$$

and $\vec{p} = (p_1, p_2)$ is the vector of probabilities of the two groups of objects in this case. The following theorem holds (the proof of the theorem can be found in [5]):

Theorem 4: *The function* $k_1(\vec{p}) = \frac{nTB(S+R)C+B^2C-\sqrt{const}}{2\,p_1\,p_2\,nT(S+R)^2C^2+4\,p_1\,p_2\,B(S+R)C^2}$ *which gives the maximum number of the streams that can be supported by two partitions of the n system disks in which two groups of objects with probabilities* p_1, p_2 *are striped (one group of objects striped per partition) is Schur concave with respect to* p_1, p_2.

Since $k_1(\vec{p})$ is Schur concave it is maximized when the vector $\vec{p}_{opt} = (p_{1,opt}, p_{2,opt})$ of the access probabilities of the two groups of delay-sensitive objects that are striped in each of the partitions is the one that is majorized by any other vector $\vec{p} = (p_1, p_2)$. Practically, this means that the maximum number of streams is achieved when the grouping scheme of the delay-sensitive objects into the partitions, results in the two partitions having as equal cumulative probabilities as possible.

- **Grouping Delay-Sensitive Objects with Different Consumption Rates.** In this case, different delay-sensitive objects have different consumption rates and access probabilities. In particular, the i^{th} delay-sensitive object $(i = 1, \ldots, r)$ has access probability p_i and consumption rate C_i such that there exists a j, $j \neq i$ for which $C_i \neq C_j$ and $p_i \neq p_j$ hold.

  Let n_1 and n_2 $(n_1+n_2 = n)$ be the fixed number of disks that are allocated to partitions r_1 and r_2 respectively. Given n_1 and n_2 we determine which delay-sensitive objects must be striped across partition r_1 and which of them must be striped across r_2 so that the total capacity of the two partitions is maximized. In order to maximize performance the assignment of delay-sensitive objects to the two partitions must be such that the sum of the products p_jC_j over all the delay-sensitive objects that are striped across partition r_1 is equal to $\sum_{j=1}^{r} p_j C_j \left(\frac{\sqrt{\frac{n_1}{n_2}}}{1+\sqrt{\frac{n_1}{n_2}}}\right)$ while for the objects that are striped across partition r_2 the corresponding sum must be equal to $\sum_{j=1}^{r} p_j C_j \left(\frac{1}{1+\sqrt{\frac{n_1}{n_2}}}\right)$ (see [5] for the proof).

8 Optimal Striping of Objects when the Access Probabilities have a Variance

According to the theorem 2 of section (5) when n disks are available for striping two delay-sensitive objects with different access probabilities p_1^*, p_2^* then it is always optimal to create two different partitions of the n disks and to stripe each of the delay-sensitive objects separately in each partition. Moreover, we proved in section 5.2, that among all the possible partitioning schemes of the n disks (in two partitions) there exists one optimal allocation vector $\vec{n^*} = (n_1^*, n_2^*)$ for the two partitions which accomplishes the maximum number of supported streams. In other words, the maximum number of concurrent streams that the partitioned configuration with fixed partitions n_1^*, n_2^* can support is higher than the maximum number of the concurrent streams that can be supported by the unpartitioned configuration.

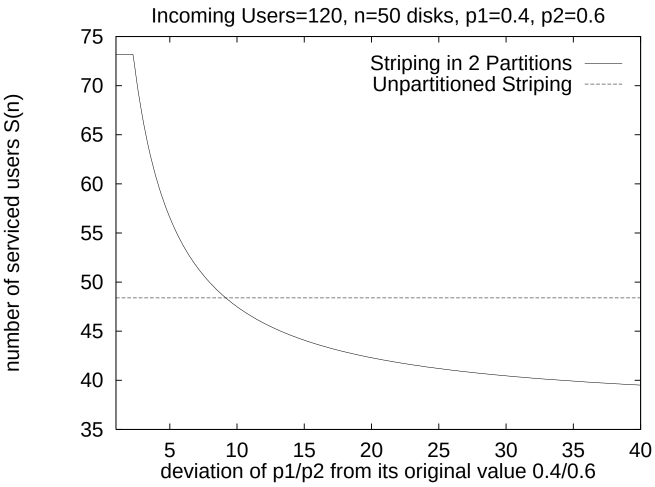

Figure 4: The number of serviced streams against the increases of the probability ratio $\frac{p_1}{p_2}$ when the number of incoming requests is 120. The system parameters were C=10 Mbps, T=30 Mbps, B=10 Mbits, S=0.01 s, and R=0.004 s.

However, if the request arrival rates for the two delay-sensitive objects have a variance, then the partitioned scheme becomes less efficient. Assuming that the partitions have fixed sizes n_1^*, n_2^* then as the ratio $\frac{p_1}{p_2}$ of the object probabilities increases or decreases, the number of concurrent users that can be serviced by the partitioned configuration drops and it may become even smaller than the capacity of the unpartitioned configuration. Whether the probability distribution of p_1, p_2 is more flat or more skewed will impact on the maximum number of the users supported by the partitioned configuration.

In figure (4) the number of concurrent clients supported by the partitioned and the unpartitioned striping scheme has been drawn against the variance of the probability ratio $\frac{p_1}{p_2}$ of the two objects. The horizontal line represents the number of streams supported by the unpartitioned striping scheme. The decreasing curve corresponds to the partitioned striping scheme. The partitioned configuration was constructed using the optimal allocation (n_1^*, n_2^*) when the probability ratio of the two objects was $\frac{p_1^*}{p_2^*} = \frac{0.4}{0.6}$ and the number of incoming requests in the system was 120. Then the ratio $\frac{p_1}{p_2}$ was increased gradually and the number of serviced users started to drop. The portion of the curve that lies below the horizontal line denotes that for the corresponding values of p_1, p_2 the performance of the partitioned striping is worse than the performance of the unpartitioned striping.

9 Summary-Conclusions

In this paper we have analyzed some fundamental issues in the performance of striping for delay-sensitive data.

We have provided a framework for the analytic evaluation of striping in a distributed environment. We derived a closed-form formula that gives the minimum number of disks required to do striping in order to accommodate a given number of users, and alternatively the number of users that can be supported in parallel from a given set of disks.

In the case that more than one delay-sensitive streams have to be supported, we analyzed the performance trade-offs for the placement of delay-sensitive objects with different access probabilities in a set of n disks. We derived a simple optimal placement strategy for a partitioned allocation. If the delay-sensitive objects are to be striped and several objects are to be placed on the same partition of disks we proved that an optimal strategy is to partition the delay-sensitive objects such that the total access probability of each partition is the same.

Although the principles established with the analytic model in this paper can help the designers to understand better the principles of striping and delay-sensitive data placement on disks, in real environments other factors may play a role as well. Such factors include communication bandwidth limits or traffic variations in the communications, CPU overhead for various real-time tasks, variations in the consumption rates, variations in the buffer size and the secondary storage size of the client workstation etc. Thus, these principles should always be seen in the larger system framework. To understand better the other system parameters we are performing system implementation in large scale and we are also implementing and experimenting with benchmarks. Some of the benchmarking results for the delay-sensitive data in the distributed framework of KYDONIA are reported in ([7]).

References

[1] Berson S., Muntz R., Ghandeharizadeh S. and Ju X., *"Staggered Striping in Multimedia Information Systems"*, Proceedings of ACM SIGMOD Conference, May 1994, pp. 79-90.

[2] Chen P.M. and Patterson D.A., *"Maximizing performance in a striped disk array"*, Proc. 17th Int'l Symp. on Computer Architecture, Seattle, Wash., pp.322-331, May 1990.

[3] D.J. Gemmell, *"Disk Scheduling for Continuous Media"*, in "Multimedia Information Storage and Manage-

ment", edited by Soon M. Chung, Kluwer Publications, 1996.

[4] Christodoulakis S. and Triantafillou P., *"Research and Development Issues for Large-Scale Storage Servers"*, Information Systems Journal, 20, 6, pp. 443-445, 1995.

[5] Christodoulakis S. and Zioga F., *"Principles of Striping and Placement of Delay-Sensitive Data on Disks"* MUSIC Technical Report, No 9., 1997.

[6] Christodoulakis S., Pappas N., Kyriakaki G., Maragoudakis Y., Mavraganis Y. and Tsinaraki C., *"The KYDONIA Multimedia Information Server"*, In Proceedings of ECMAST 97.

[7] Christodoulakis S., Magoulioti D. and Triantafillou P., *"Benchmarking and Performance Tuning of Multimedia Servers"*, In Proceedings of EuroPAR'97.

[8] Dan A., Kienzle M. and Sitaram D., *"A dynamic policy of segment replication for load-balancing in video-on-demand servers"*, Multimedia Systems, 1995, 3: pp.93-103.

[9] Gemmell J. and Christodoulakis S., *"Principles of Storage and Retrieval of Multimedia Data"*, ACM Transactions on Information Systems (ACM TOIS), 10, 1, Jan 92.

[10] Chang E., Garcia-Molina H., *"Effective Memory Use in a Media Server"*, Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB 97), Athens, August 97.

[11] Marshall A. and Olkin I., *"Inequalities: Theory of Majorization and its Applications"*, Academic Press, University of Southern California, 1979.

[12] Merchant A. and Yu P., Fellow, IEEE, *"Analytic Modeling and Comparisons of Striping Strategies for Replicated Disk Arrays"*, IEEE Transactions on Computers, 1995, 44: pp. 419-433.

[13] Ng R. and Yang J., *"An Analysis of Buffer Sharing and Prefetching Techniques for Multimedia Systems"*, The Department of Computer Science, University of British Columbia, Technical Report TR-94-20 (http://www.cs.ubc.ca/cgi-bin/tr/index).

[14] Özden B., Rastogi R., Silberschatz A., *"On the design of a low-cost video-on-demand storage system"*, Multimedia Systems, 1996, 4: pp. 40-54.

[15] B. Özden, R. Rastogi, A. Silberschatz, *"Disk striping in video server environments"*, IEEE International Conference on Multimedia Computing and Systems, June 1996.

[16] Salem K. and Garcia-Mollina H., *"Disk Striping"*, Proceedings of 4^{th} IEEE International Conference on Data Engineering, 1986, pp. 336-342.

[17] Scheuermann P., Weikum G., Zabback P., *"Data Partitioning and Load Balancing in Parallel Disk Systems"*, Technical Report A/02/96, Department of Computer Science, University of Saarland, April 1996.

[18] Tobagi F., et al., *"Streaming RAID - A Disk Array Management System for Video Files"*, Proceedings of ACM Multimedia, 1993, pp. 393-400.

[19] Triantafillou P. and Faloutsos C., *"Overlay Striping and Optimal Parallel I/O in Modern Applications"*, Parallel Computing Journal, Special Issue on Parallel Data Servers and Applications, 1998, (also available from http://www.music.tuc.gr/Research/Projects.html).

Appendix

The maximum number of supported streams is $k_1 = \frac{\sigma_1}{\sigma_2}$ where

$$\begin{aligned}
\sigma_1 \;=\;& -p_1 C_1 B_1 B_2 - C_2 B_2 B_1 - B_2 N p_1 C_1 R - C_2 R N B_1 \\
& -B_2 N p_1 C_1 S + C_2 p_1 B_2 B_1 + C_2 p_1 R N B_1 \\
& -C_2 S N B_1 + C_2 p_1 S N B_1 + \sqrt{\Omega}
\end{aligned}$$

$$\begin{aligned}
\sigma_2 \;=\;& -2 p_1 C_1 B_1 C_2 S - 2 p_1 C_1 B_1 C_2 R + 2 {p_1}^2 C_1 B_1 C_2 S \\
& +2 {p_1}^2 C_1 B_1 C_2 R - 2 C_2 S^2 N p_1 C_1 - 4 C_2 S N p_1 C_1 R \\
& -2 C_2 R^2 N p_1 C_1 - 2 C_2 B_2 p_1 C_1 S - 2 C_2 B_2 p_1 C_1 R \\
& +2 C_2 {p_1}^2 S^2 N C_1 + 4 C_2 {p_1}^2 S N C_1 R + 2 C_2 {p_1}^2 R^2 N C_1 \\
& +2 C_2 {p_1}^2 B_2 C_1 S + 2 C_2 {p_1}^2 B_2 C_1 R
\end{aligned}$$

$$\begin{aligned}
\Omega \;=\;& {C_2}^2 {B_2}^2 {B_1}^2 + {p_1}^2 {C_1}^2 {B_1}^2 {B_2}^2 + 2 p_1 C_1 {B_1}^2 {B_2}^2 C_2 \\
& +2 {p_1}^2 {C_1}^2 B_1 {B_2}^2 N R - 2 p_1 C_1 {B_1}^2 B_2 C_2 R N \\
& +2 {p_1}^2 {C_1}^2 B_1 {B_2}^2 N S - 2 {p_1}^2 C_1 {B_1}^2 {B_2}^2 C_2 \\
& +2 {p_1}^2 C_1 {B_1}^2 B_2 C_2 R N - 2 p_1 C_1 {B_1}^2 B_2 C_2 S N \\
& +2 {p_1}^2 C_1 {B_1}^2 B_2 C_2 S N - 2 C_2 {B_2}^2 B_1 N p_1 C_1 R \\
& +2 {C_2}^2 B_2 {B_1}^2 R N - 2 C_2 {B_2}^2 B_1 N p_1 C_1 S \\
& -2 {C_2}^2 {B_2}^2 {B_1}^2 p_1 - 4 {C_2}^2 B_2 {B_1}^2 p_1 R N \\
& +2 {C_2}^2 B_2 {B_1}^2 S N - 4 {C_2}^2 B_2 {B_1}^2 p_1 S N \\
& +{B_2}^2 N^2 {p_1}^2 {C_1}^2 R^2 - 2 B_2 N^2 p_1 C_1 R^2 C_2 B_1 \\
& +2 {B_2}^2 N^2 {p_1}^2 {C_1}^2 R S + 2 {B_2}^2 N {p_1}^2 C_1 R C_2 B_1 \\
& +2 B_2 N^2 {p_1}^2 C_1 R^2 C_2 B_1 - 4 B_2 N^2 p_1 C_1 R C_2 S B_1 \\
& +4 B_2 N^2 {p_1}^2 C_1 R C_2 S B_1 + {C_2}^2 R^2 N^2 {B_1}^2 \\
& -2 {C_2}^2 R^2 N^2 {B_1}^2 p_1 + 2 {C_2}^2 R N^2 {B_1}^2 S - 4 {C_2}^2 R N^2 {B_1}^2 p_1 S \\
& +{B_2}^2 N^2 {p_1}^2 {C_1}^2 S^2 + 2 {B_2}^2 N {p_1}^2 C_1 S C_2 B_1 \\
& -2 B_2 N^2 p_1 C_1 S^2 C_2 B_1 + 2 B_2 N^2 {p_1}^2 C_1 S^2 C_2 B_1 \\
& +{C_2}^2 {p_1}^2 {B_2}^2 {B_1}^2 + 2 {C_2}^2 {p_1}^2 B_2 {B_1}^2 R N \\
& +2 {C_2}^2 {p_1}^2 B_2 {B_1}^2 S N + {C_2}^2 {p_1}^2 R^2 N^2 {B_1}^2 \\
& +2 {C_2}^2 {p_1}^2 R N^2 {B_1}^2 S + {C_2}^2 S^2 N^2 {B_1}^2 \\
& -2 {C_2}^2 S^2 N^2 {B_1}^2 p_1 + {C_2}^2 {p_1}^2 S^2 N^2 {B_1}^2
\end{aligned}$$