Independent Access Array for Continuous Media Storage Management (Extended Abstract*)

Masatoshi Saito†
masa@shasta.stanford.edu

Hector Garcia-Molina
hector@cs.stanford.edu

Department of Computer Science
Stanford University
Stanford, CA 94305

Abstract

It is important for a continuous media server to meet strict timing constraints at high bandwidth. Most continuous media servers use disk scheduling schemes, careful data allocation, disk arrays, and/or memory buffers to support simultaneous users within the limitation of the available computer resources. In this paper we analyze buffer requirements in several scenarios. In particular, we show that an independent disk array has significantly lower buffer requirements as compared to a parallel array. We also show that a layered disk array is a good choice for a continuous media server.

Keywords: *Condinuous Media, Buffer, Disk Array, Layered Disk Array, Disk Scheduling.*

1 Introduction

The characteristics of Continuous Media (**CM**), such as motion video and audio, are different from those of conventional data. In particular, it is important to provide multiusers with guaranteed simultaneous data transfers. To meet these guarantees, efficient disk scheduling schemes, optimal data allocation, disk arrays and/or large memory buffers can be used. In this paper we study how efficient disk scheduling techniques can be used with disk arrays to meet application timing constraints. We evaluate the performance of the schemes as well as their buffer requirements.

For our evaluation, we consider three disk scheduling schemes. To support simultaneous multiusers, a *fixed order round-robin scheme* (**FIX**) was proposed in [1]. A stream[1] is divided into segments[2], and during a *service cycle*, each segment in a buffer is consumed (at a constant rate) by the user requesting that stream. If the number of users is N, during a service cycle N I/Os must be performed. Assuming that segments from different streams are scattered on the disk, a CM storage manager needs to perform N full cylinder seeks during a service cycle. Thus, the fraction of the seek operations in a service cycle is considerable and the segment size should be large to amortize the cost of the seeks. With this scheme, we need one memory buffer to hold the current segment being consumed by each user. In addition, a single *staging buffer* is used to hold the segment being read from disk. Thus, the total buffer size used is $(N+1)$ times the segment size.

An *elevator-type scheduling scheme* (**SCN**), introduced by [2], reads segments in the order they are stored on the disk so that the total seek distance in a service cycle is always equal to 1 full cylinder. Thus, SCN can minimize the seek time overhead and the segment size, allowing more concurrent users than a FIX scheme. However, since transfers are not in the order users need them, each stream needs a read buffer and a staging buffer, for a total buffer size of $2N$ times the segment size.

A *grouped sweeping scheme* (**GSS**), which reduces the buffer requirement and the total seek distance, was proposed in [3]. Users are divided into G *groups*; in each group streams are handled in the same way as SCN, so that the total seek distance for GSS in a service cycle is more than that for SCN by a factor of G. However, the required segment size lies between that for FIX and for SCN. In this case, during a disk scan we need one staging buffer for each segment in the group. In addition we need one read buffer per user. The total memory requirement is $(N + \lceil N/G \rceil)$ times the segment size. A storage manager can select a good G value to minimize the total buffer requirement.

Our goal is to apply these disk scheduling and buffering strategies to disk arrays. We consider two types of arrays [4]. In an *independent access array* (**I-Array**) each segment is stored contiguously on one of the member disks[3]. The segments for a given stream could go on the same member or could be striped across the members in a variety of ways. Each array member disk has an independent

*The full paper will be in anonymous ftp: db.stanford.edu, under directory: /pub/saito/1994.

†On leave as a Visiting Scholar from Oki Electric Industry Co., Ltd.

[1] A CM file which is played back without jitter for a user.

[2] The part of a stream retrieved by a single I/O operation.

[3] A disk drive which is a component part of a disk array.

rotational and actuator position. In a *parallel access array* (**P-Array**) a segment is divided by byte or bit across the members. Every segment request to a P-Array always causes all members to seek and read. So the actuators for all member disks move in unison and are positioned on the same cylinder of each disk. In a P-Array, a segment is spread across all member disks, so it can be read in at a faster rate. In an I-Array, on the other hand, if segments are stored on different members, they can be read concurrently.

2 Issues

A key issue for I-Arrays is that of data allocation and load balancing. If we can choose a suitable segment size for a system and balance loads to all member disks evenly, the total performance of an I-Array can be maximized. In practice, particularly in conventional environments, there are mixed types of access patterns from several applications, and it is difficult to select an optimum segment size to balance loads.

For CM, however, we can take advantage of the isochronous nature of data in order to balance loads to member disks. To illustrate, consider an I-Array with four member disks. Assume that for a given segment size, a member disk can perform 15 I/Os in the time a user consumes its segment. Assume further that the segments that make up a stream are always stored on the same member disk. Suppose now that 60 users concurrently request streams that are all located on Member 0. In this worst case, the system can only support 15 of the 60 users, i.e., 45 users will have to wait until the first 15 fully playback their streams. Member 0 is fully utilized performing 15 I/Os each service cycle, while the remaining three members are idle. This is one of unbalanced situations.

However, performance can improve dramatically if the segments for a stream are placed on different members, as suggested by the top part of Figure 1. (Stream A consists of segments $A0, A1, A2, ...$; stream B consists of $B0, B1, ...$ and so on.) Using this allocation we can now service all 60 users. For example, suppose that all 60 requests arrive at the same time, and all are for streams that start on Member 0. In the first service cycle, 15 users get their first segments, while 45 wait. During the second cycle, the first 15 users read their second segments from Member 1, while another 15 read from Member 0. After three cycles all members become fully utilized. In this case, the utilization of each member disk is 100% and each load to a member is even. Of course, in practice we may not achieve this full utilization if streams are short (relative to the startup time).

The bottom portion of Figure 1 illustrates the allocation for a P-Array. With a P-Array, the segment transfer rate is higher than with an I-Array. However, the seek cost for an I/O is higher, since we have to do seeks on all member disks. Thus, if our disk scheduling policy and available buffer space force us to use small segments, then the higher transfer rate may not pay off and a P-Array may be counter productive.

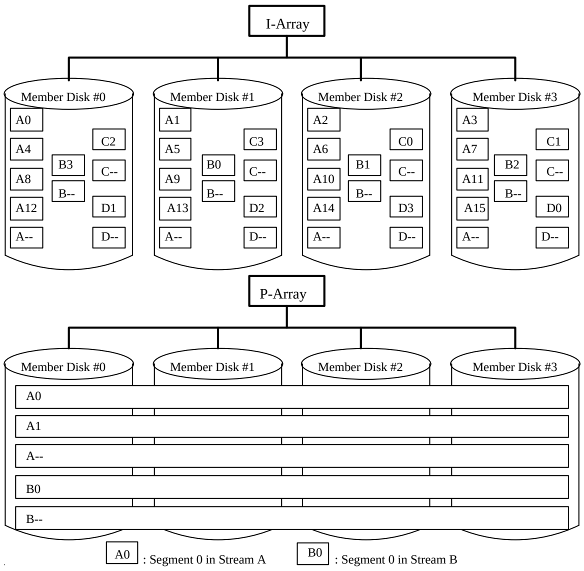

Figure 1: Data allocations for I-Array and P-Array.

As one can see, there are a number of interesting tradeoffs and interactions between the scheduling policy, the buffer space available, the array structure, the segment allocation policy, and the particular parameters of the hardware. In the full paper we will study these issues in detail. In particular, the contributions will be:

- An analysis of the buffer requirement for I-Arrays and P-Arrays in CM applications, under various scheduling strategies;
- An evaluation of data allocation strategies for I-Arrays;
- A sensitivity analysis of disk hardware parameters, exploring their impact on buffer requirements and throughput;
- An analysis of the buffer requirement for layered arrays[4] in CM applications.

3 Sample Results

In the rest of this extended abstract, we illustrate some of the results by describing three of the graphs we have obtained:

- Buffer requirements for different disk array structures and disk scheduling schemes;
- Buffer requirement sensitivity to member disk parameters;
- Buffer requirements for layered arrays.

For the analysis, we use an example disk based on the characteristics of a state-of-the-art 3.5 inch form factor disk

[4] In a layered array, I-Arrays and P-Arrays are combined in a hierarchical fashion.

drive, which is 16.5 and 0.6 msec maximum and minimum seek time respectively and 7200 rpm rotational speed, and use four example disks to construct a disk array. The seek cost depends on the square root of the seek distance. Each stream is assumed to be MPEG I with a 1.5 Mbit/sec playback rate. Data allocation is as in Figure 1; an optimal G value, which minimizes the total buffer size, is used for GSS. We also assume that all users issue their stream requests at the same time, starting with an idle system. First we calculate the minimum segment size at a given number of streams from the constraint that ensures users playback CM streams without jitter. Next, we compute the buffer size from the segment size and one of the three buffer requirement equations defined by the disk scheduling strategies.

Figure 2 shows the impact of the two different disk array structures and the three disk scheduling schemes on buffer requirements. In all cases, as the number of streams increases, more buffer space is needed. The more users the storage manager supports, the more I/O operations are needed to read the corresponding segments. This makes a service cycle longer. Each stream then needs a bigger segment size in order that a user has enough data to read during the longer cycle. This in turn leads to the larger buffer space requirement.

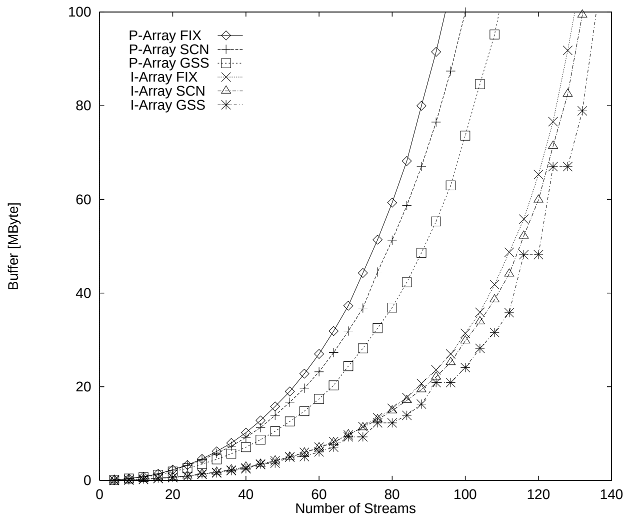

Figure 2: Buffer requirements for different disk array structures and disk scheduling schemes.

As pointed out in [3], GSS has the smallest buffer requirement among the three schemes. In all cases, an I-Array can reduce the buffer requirement significantly, compared with a P-Array. For example, to support 60 streams with SCN, an I-Array needs only a 7.2 MByte buffer, while a P-Array needs a 23.8 MByte buffer. This is because each member of the P-Array has to serve 60 users, while each I-Array member is serving only 15 users. In other words, an I-Array can support more streams than a P-Array given the same buffer space.

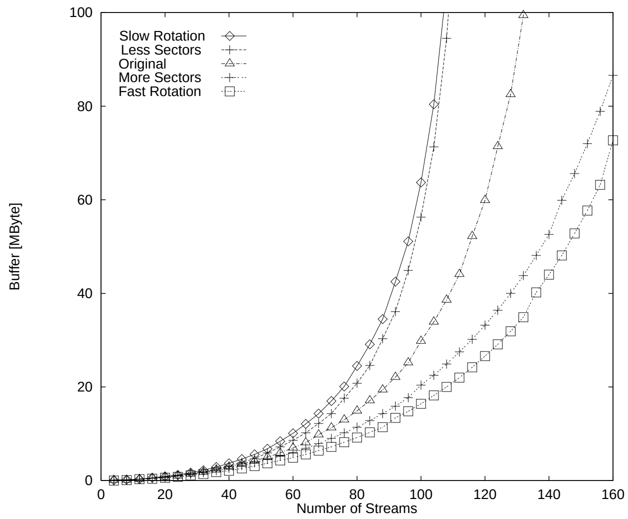

Figure 3: Buffer requirement sensitivity to different specifications of a member disk, I-Array /w SCN.

Next, we examine the sensitivity of the results to variations in the disk hardware parameters. We vary the following parameters (one at a time) by plus or minus 30% from the base case: seek time, rotational speed, head switching time and the number of sectors in a track. Figure 3 shows the results for an I-Array with a SCN scheduling scheme. For clarity, the curves for faster/slower seek time and head switching time are not shown, and are similar to that of the original. It is important to note that seek time does not significantly affect the buffer requirement because the fraction of seek time has been already reduced by the efficient SCN scheduling. What has the strongest impact on the buffer requirement is the rotational speed. For example, with 30% faster rotational speed, a 50 MByte buffer can support 140 streams, instead of the original 110 streams. Unfortunately, the 7200 rpm rotational speed for the original specification is at the edge of current disk technology. However, increasing the number of sectors in a track, which derives from higher areal density, can also improve performance. Disk areal density has recently improved 60% per year. The difference between the rotational speed and the number of sectors in a track is that the former can reduce both transfer time and rotational latency, while the latter can reduce only transfer time. So the rotational speed has greater impact than the other. The disk with faster rotational speed or with higher areal density improves performance of an I-Array more than the one with a shorter seek time.

To improve availability and reliability of disk arrays, several types of redundancy have been suggested. Here we consider two types, where arrays are layered: RAID level53 [4] and RAID level10 [4] (see Figure 4). The former consists of several P-Arrays, each with its own parity disk, as a substitute for an ordinary member disk drive. The latter uses several mirrored arrays. Since they both have redundant data, they are able to continue to function even when one of the member disks fails. For our analysis we assume each *logical* disk has the same capacity as the disks in our original non-redundant arrays. This means that for RAID level10, the physical disks must have the same capacity as the original physical disks. However, for RAID level53, data is not fully duplicated, so physical disks can be smaller. In particular, for RAID level53 we assume smaller 2.5 inch disks (which have poorer performance as compared to the original 3.5 inch disks). We also assume the *SCN* scheduling policy is used.

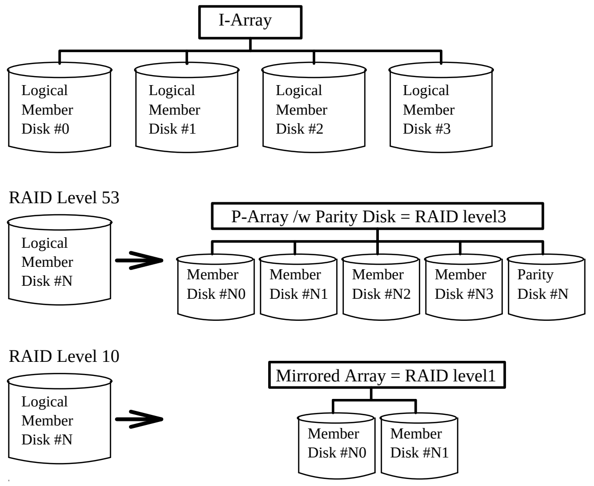

Figure 4: Structures of layered arrays.

Figure 5 shows the buffer requirements for the two layered arrays (each with four logical disks) under normal operation. RAID level10 has the smallest buffer requirement for most values of the number-of-streams parameter. Since it has twice as many disk drives as the original I-Array, under normal operation it behaves as a simple I-Array with eight member disks. However, once one of the member disks fails, RAID level10 has the same buffer requirement as the original 4-disk I-Array has. Thus, the key features of RAID level10 are its 100% space overhead and the different buffer requirements between normal and failure mode.

RAID level53, on the other hand, can function at the same buffer requirement even in failure mode. Moreover, the space overhead is less than that of RAID level10. Figure 5 shows the buffer requirements for three types of RAID level53, where the number of member disks in each logical member disk are 2, 4 and 8. The array with 2 member disks (plus parity disk) per logical disk needs a larger buffer than the original I-Array. This is because it cannot amortize the inferior performance of member disks. The other two types, however, need smaller buffers. Thus, we see there is an interesting tradeoff between the characteristics of a member disk and the number of member disks in RAID level53.

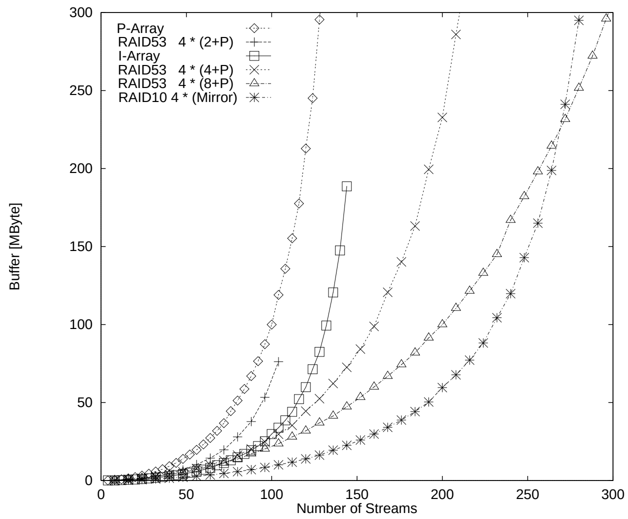

Figure 5: Buffer requirements for layered arrays /w SCN.

4 Conclusion

In this paper we have analyzed the buffer requirements (for continuous media) of several types of disk arrays, under a variety of disk scheduling policies. We showed that an independent disk array is more effective than a parallel array, in terms of its buffer requirements. We have also shown that a layered disk array, especially RAID level53, is a good choice for a continuous media server.

References

[1] P.V. Rangan and H.M. Vin, "Design file systems for digital video and audio", *Proc. of 12th ACM Symposium on Operating Systems*, 1991.

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

[3] P.S. Yu, M-S. Chen and D.D. Kandlur, "Design and analysis of a grouped sweeping scheme for multimedia storage management", *IBM Research Report RC18376(80350)*, Sep, 1992.

[4] "The RAID book 1st edition", *The RAID Advisory Board*, Jun, 1993.