Continuous Display Using Heterogeneous Disk-Subsystems[*]

Roger Zimmermann
Computer Science Department
University of Southern California
Los Angeles, California 90089
RZIMMERM@CS.USC.EDU

Shahram Ghandeharizadeh
Panasonic Technologies Inc.
2 Research Way
Princeton, New Jersey 08540
SHAHRAM@RESEARCH.PANASONIC.COM

Abstract

A number of recent technological trends have made data intensive applications such as continuous media (audio and video) servers a reality. These servers store and retrieve a large volume of data using magnetic disks. Servers consisting of heterogeneous disk drives have become a fact of life for several reasons. First, disks are mechanical devices that might fail. The failed disks are almost always replaced with new models. Second, the current technological trend for these devices is one of annual increase in both performance and storage capacity. Older disk models are discontinued because they cannot compete with the newer ones in the commercial arena. With a heterogeneous disk subsystem, the system should support continuous display while managing resources intelligently in order to maximize their utilization.

This study describes a taxonomy of techniques that ensure a continuous display of objects using a heterogeneous disk subsystem. This taxonomy consists of: (a) strategies that partition resources into homogeneous groups of disks and manage each independently, and (b) techniques that treat all disks uniformly, termed non-partitioning techniques. We introduce three non-partitioning techniques: disk merging, disk grouping, and staggered grouping. We investigate these techniques using analytical models. Our results demonstrate that disk merging is the most flexible scheme while providing among the lowest cost per simultaneous display. Finally, using an open simulation model, we compare disk merging with a partitioning technique. The obtained results demonstrate the superiority of disk merging.

Keywords

Heterogeneous disk subsystems; disk arrays; continuous media servers; continuous media storage systems.

[*]This research was supported in part by a Hewlett-Packard unrestricted cash/equipment gift, and the National Science Foundation under grants IRI-9203389 and IRI-9258362 (NYI award).

1 Introduction

Magnetic disks have established themselves as the mass storage device of choice for data intensive applications such as video-on-demand servers and multimedia applications in general. The current technological trend for disks is an annual increase in performance and storage capacity at a reduced cost, see Figure 1.

The performance improvements include a lower seek time, a reduced rotational delay, and a higher transfer rate. With this trend, for large servers consisting of many disks, a homogeneous disk subsystem might evolve over time to consist of a collection of heterogeneous disks. There are several reasons for this. First, existing disks might fail, forcing the operator to replace them with newer disk models. Although a single disk can be fairly reliable, with a large number of disks in a server, the aggregate rate of disk failures can be too high. At the time of this writing, the mean time to failure (MTTF) of a single disk is in the order of 600,000 hours (approximately 70 years). With a system consisting of 1000 such disks, the MTTF of some disk is on the order of 600 hours (approximately 25 days). Every time a disk fails, the operator might replace it with a newer disk model because either (1) this disk model costs less and provides a higher performance, or (2) the model of the failed disk has been discontinued and is no longer available.

Second, with a scalable server (e.g., Mitra [15]), a service provider might want to expand the server to meet its increased demand for both storage and bandwidth. If several years have passed since the original installation of the system, the service provider might be forced to purchase newer disk models, yielding a heterogeneous disk subsystem. Obviously, a technique that harnesses the full potential of new disks with no impact on system behavior is desirable.

With multimedia applications, display of continuous media (e.g., audio and video clips) is an important system characteristic. The main attribute of continuous media is their sustained bit rate requirement. If a system delivers a clip at a rate lower than its prespecified rate without special precautions (e.g., pre-fetching) then the user might observe frequent disruptions and delays with video and random noises with audio. These artifacts are collectively termed *hiccups*. A number of studies have addressed hiccup-free display of audio and video clips assuming a homogeneous disk subsystem [2]. In this paper, we investigate techniques that ensure continuous display of audio and video clips with heterogeneous disk drives. Figure 2 shows a taxonomy of the possible approaches. With the *partitioning* schemes, disks are grouped based on their model. To illustrate, assume a system that has evolved to consist of three types of disks: Quantum PD425S, Seagate ST31200W, Seagate ST32171W; see Table 1. With this approach, the system constructs three disk groups. Each group is managed independently. A frequently accessed (hot) clip might be replicated on different

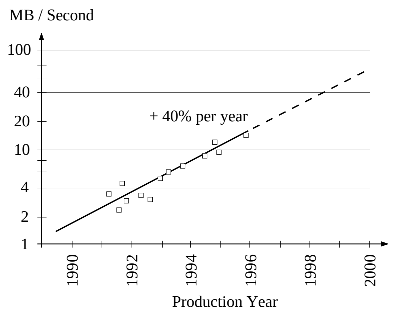


Figure 1a: (Media) data rate improvement [18]

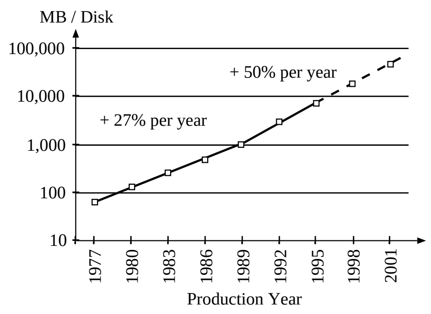


Figure 1b: Capacity improvement [23]

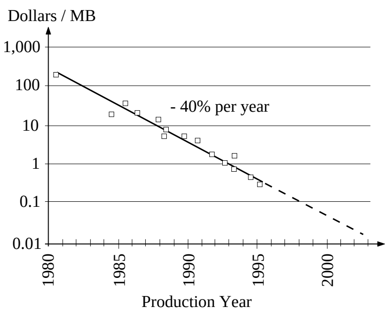


Figure 1c: Cost per megabyte decline [17]

Figure 1: Technological trends for magnetic disk drives

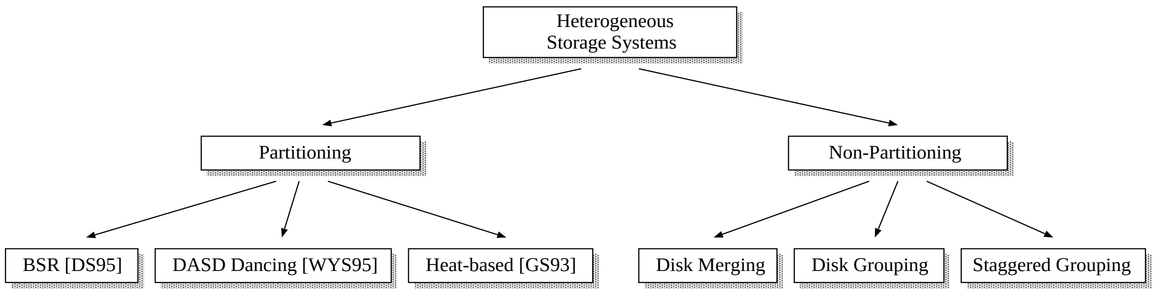


Figure 2: Taxonomy of techniques

Series Model Manufacturer	ProDrive PD425S Quantum	Hawk 1LP ST31200W Seagate	Barracuda 4LP ST32171W Seagate
Capacity C	0.406 GB	1.006 GB	2.061 GB
Avg. transfer rate R_D	1.64 MB/s	3.47 MB/s	7.96 MB/s
Spindle speed	3600 rpm	5400 rpm	7200 rpm
Avg. rot. latency	8.34 msec	5.56 msec	4.17 msec
Max. seek time	27 msec	21 msec	19 msec

Table 1: Parameters for three commercial disk drives

groups in order to avoid formation of hot spots and bottlenecks [7, 13, 30]. With the *non-partitioning* schemes, the system constructs a logical representation of the physical disks. This logical abstraction provides the illusion of a homogeneous disk subsystem to those software layers that ensure a continuous display.

In general, the non-partitioning schemes are superior because the resources (i.e., bandwidth and storage space) are combined into a unified pool and no local bottlenecks that involve only part of the resources (i.e., hot spots) can form. Hence, they are sensitive to neither the frequency of access to objects nor the distribution of requests as a function of time. The partitioning schemes, on the other hand, are detective because they must recognize hot objects and replicate them. The period spent recognizing hot objects might result in formation of bottlenecks that degrade the overall system performance. Moreover, scheduling of resources is simple with non-partitioning schemes. With the partitioning techniques, the system must monitor the load on each disk partition when activating a request in order to balance the load across partitions evenly. This becomes a difficult task when all partitions are almost completely utilized. The disadvantage of non-partitioning techniques is as follows. First, the design and implementation of availability techniques that guarantee a continuous display in the presence of disk failures becomes somewhat complicated. Second, deciding on the configuration parameters of a system with a non-partitioning technique is not a trivial task. The contribution of this study is the design and analysis of non-partitioning techniques.

Two studies have investigated non-partitioning techniques in support of single-user environments that retrieve the data at a high bit rate [28, 5]. Their target application is an image server that contains aerial photographs of an area of interest for the purpose of terrain visualization by a single user (approximate bit rate requirement is 300-400 megabits per second). These pioneering studies are orthogonal to our study. We start by describing the differences between this study and [28, 5]. Next, we describe how these studies are orthogonal to each other.

The main differences between this study and [28, 5] are as follows. First, our study focuses on multiple users displaying different audio and video clips at the same time where the bandwidth requirement of each clip can differ (and might be in the order of hundreds of megabits per second). Our techniques *guarantee* a continuous display on behalf of each display. Second, in our target application, a display typically retrieves the blocks of a clip sequentially, facilitating design of intelligent techniques based on a deterministic schedule of data retrieval. With the terrain visualization, a display might navigate different areas arbitrarily and the system has no advance knowledge of which block might be retrieved in the future.

These studies are orthogonal to each other due to our focus on multimedia applications. The techniques proposed by [5] can be easily extended to maximize the performance of a heterogeneous disk subsystem in support of multiple users retrieving traditional data types that have no real-time constraints, e.g., text, records, objects, etc. To maximize system performance, a multimedia server can employ our techniques for continuous media and the techniques proposed by [5] for traditional data types.

The rest of this paper is organized as follows. Section 2 describes three different non-partitioning techniques. An analytical analysis of these techniques demonstrates that one of them, namely disk merging, is the most flexible while at the same time providing among the lowest cost per stream. Section 3 compares disk merging with a non-partitioning technique to quantify the superiority of disk merging. We conclude with the future research directions in Section 4.

Term	Definition
R_{D_i}	Transfer rate of physical disk i
R_C	Display bandwidth requirement (consumption rate)
X	Continuous media object
X_i	Logical block i of object X
$X_{i.j}$	Fragment j of logical block i (of object X)
d_i	Physical disk drive i
ld_i	Logical disk drive i
m	Number of logical disks
K	Number of physical disks that constitute a logical disk
$\mathcal{N}$	Maximum number of simultaneous displays supported by a single logical disk
$\mathcal{N}_{Tot}$	Maximum number of simultaneous displays supported by the system (throughput)
$\mathcal{B}_i$	Size of a fragment assigned to physical disk i
$\mathcal{B}^l$	Size of a logical block assigned to a logical disk
M	Total amount of memory needed to support $\mathcal{N}_{Tot}$ streams
L_{max}	Maximum latency (without queuing)
T_p	Duration of a time period (also round or interval)
S_p	Duration of a sub-period
$T_{Seek_y}(x)$	Seek time to traverse x cylinders ($0 < x \leq \#cyl_y$) with seek model of disk type y ($\#cyl_y$ denotes the total number of cylinders of disk type y); e.g., Figure 4b

Table 2: List of parameters used repeatedly in this paper and their respective definitions

2 Three Non-partitioning Techniques

This section describes three non-partitioning techniques that guarantee a hiccup-free display of audio and video clips. These techniques differ on how they place data and schedule disk bandwidth. These decisions impact how much of the available disk bandwidth and storage is wasted, and how much memory is required to support a fixed number of simultaneous displays.

To illustrate the alternative techniques, we assume a system that consists of six disks, two of each disk type listed in Table 1. We assume a constant average transfer rate for each disk even though they are multi-zone disks with each zone providing a different transfer rate. We made this simplifying assumption in order to describe the non-partitioning techniques. It is eliminated in Section 2.4. We focus on constant bit rate media data types whose bandwidth requirement (R_C)[1] remains fixed as a function of time for the entire display time of a clip. To guarantee a hiccup-free display, a clip must be produced at a rate of R_C. This is accomplished using the concept of time period (T_p) and logical blocks ($\mathcal{B}^l$) [29, 24, 1]. Both, data placement and retrieval are performed in a round-robin manner. Once a request arrives referencing object X, the system produces a logical block of X per time period starting with X_0. The display time of a block is equivalent to the duration of a time period. A client initiates the display of the first block of X (X_0) at the end of the time period that retrieved this block. This enables the server to retrieve the blocks of the next time period using an elevator algorithm to minimize the impact of seeks [31]. A detailed description of our target architecture is contained in [15].

Below, we describe the organization of logical blocks and the concept of time period with each technique. The objective of this section is to identify the system parameters. To provide a focused presentation (and due to lack of space), we describe neither optimization techniques nor a configuration planner for computing optimal values for the different system parameters. The first two techniques are neither flexible nor as efficient as disk merging. By flexibility we imply that they cannot support an arbitrary number of disks with different physical characteristics. They are described due to their simplicity. In addition, they serve as a foundation for disk merging.

2.1 Disk Grouping

As implied by its name, this technique groups physical disks into logical ones and assumes a uniform characteristic for all logical disks. To illustrate, the six physical disks of Figure 3 are grouped to construct two homogeneous logical disks. In this figure, a larger physical disk denotes a newer disk model that provides both a higher storage capacity and a higher performance[2]. The blocks of a movie X are assigned to the logical disks in a round-robin manner to distribute the load of a display evenly across the available resources [26, 2, 21, 8]. A logical block is declustered [12] across the participating physical disks. Each piece is termed a fragment (e.g., $X_{0.0}$ in Figure 3). The size of each fragment is determined such that the service time ($T_{Service}$) of all physical disks is identical. The disk service time is composed of the transfer time (desirable) and the seek time (undesirable):

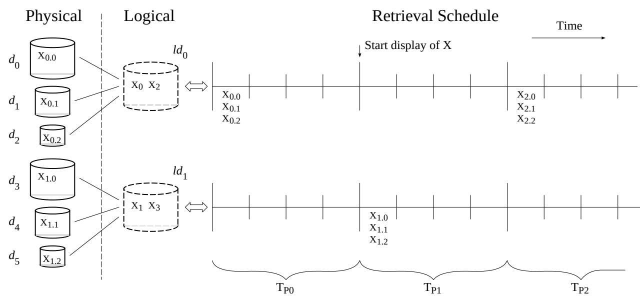

Physical
Logical
Retrieval Schedule
Time
Start display of X
d_0 d_1 d_2 d_3 d_4 d_5
ld_0 ld_1
$X_{0.0}$ $X_{0.1}$ $X_{0.2}$ $X_{1.0}$ $X_{1.1}$ $X_{1.2}$
X_0 X_2
X_1 X_3
$X_{2.0}$ $X_{2.1}$ $X_{2.2}$
T_{P0} T_{P1} T_{P2}

Figure 3: Disk Grouping

$$T_{Service} = T_{Transfer} + T_{Seek} \quad (1)$$

The transfer time ($T_{Transfer}$) is a function of the amount of data retrieved and the data transfer rate of the disk: $T_{Transfer} = \frac{\mathcal{B}}{R_D}$. The seek time ($T_{Seek}$) is a non-linear function of the number of cylinders traversed by the disk heads to locate the proper data sectors. One approach (see [25, 14]) to model such a seek profile is to approximate it with a combination of a square-root and linear function as follows:

$$T_{Seek}(d) = \begin{cases} c_1 + (c_2 \times \sqrt{d}) & \text{if } d < z \text{ cyl.} \\ c_3 + (c_4 \times d) & \text{if } d \geq z \text{ cyl.} \end{cases} \quad (2)$$

where d is the seek distance in cylinders. In addition, the disk heads need to wait on average half a platter rotation—once they have reached the correct cylinder—for the data to move underneath

[1] The definitions of the parameters used to describe the different techniques are summarized in Table 2.

[2] While this conceptualization might appear reasonable at first glance, it is in contradiction with the current trend in the physical dimensions of disks that calls for an annual reduction, e.g., the PD425S is bulkier in appearance when compared with the ST31200W.

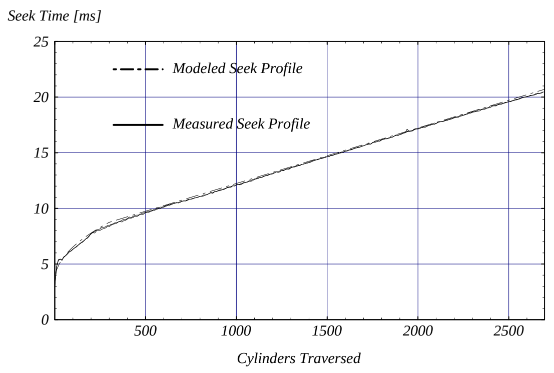


Figure 4a: Measured and modeled seek profile

ST31200W Seek Parameters	
Seek constant c_1	3.5 + 5.56[a] msec
Seek constant c_2	0.303068 msec
Seek constant c_3	7.2535 + 5.56[a] msec
Seek constant c_4	0.004986 msec
Switch-over point z	300 cylinders
Total size $\#cyl$	2697 cylinders

[a] Average rotational latency.

Table 4b: Modeling parameters

Figure 4: Seek profile of a Seagate ST31200W disk drive

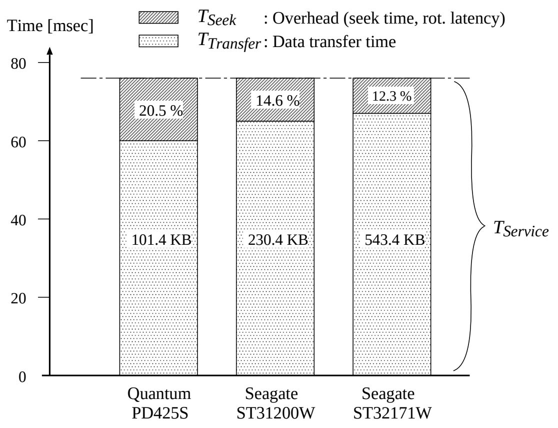


Figure 5: Seek and rotational delay overhead when retrieving blocks from different disk models

the heads. This time is termed rotational latency and we include it in our model of T_{Seek} (see c_1 and c_3 in Table 4b). Every disk type has its own distinct seek profile. Figure 4a illustrates the measured and modeled seek profile for a Seagate ST31200W disk drive (shown without the rotational latency) and Table 4b lists the corresponding constants used in Equation 2. Figure 5 illustrates how the fragment size can be adjusted to account for the difference in seek time and data transfer rate to obtain identical service times for three disk models. Throughout this paper we will use subscripts (e.g., R_{D_i}, T_{Seek_i}) to denote the parameters of a specific disk type (e.g., Quantum PD425S).

To guarantee a hiccup-free continuous display, the worst case seek time must be assumed for each disk type, i.e., the seek distance must equal the total number of cylinders, $d = \#cyl$. However, if multiple displays ($\mathcal{N}$) are supported simultaneously, then $\mathcal{N}$ fragments need to be retrieved from each disk during a time period. In the worst case scenario, the $\mathcal{N}$ requests might be evenly scattered across the disk surface. By ordering the fragments according to their location on the disk surface (i.e., employing the elevator or SCAN algorithm), an optimized seek distance of $d = \frac{\#cyl}{\mathcal{N}}$ can be obtained, reducing both the seek and the service time. We subsequently employ this optimization for all techniques described in this study.

With a fixed storage capacity and fix-sized fragments for each physical disk, one can compute the number of fragments that can be assigned to a physical disk. The storage capacity of a logical disk (number of logical blocks it can store) is constrained by the physical disk that can store the fewest fragments. For example, to support 120 simultaneous displays of MPEG-1 streams ($R_C = 1.5$ Mb/s), the fragment sizes[3] of Table 3 enable each logical disk to store only 3,883 blocks. The remaining space of the other two physical disks (ST31200W and PD425S) can be used to store traditional data types, e.g., text, records, etc. During off-peak times, the idle bandwidth of these disks can be employed to service requests that reference this data type.

When the system retrieves a logical block into memory on behalf of a client, all physical disks are activated simultaneously to stage their fragments into memory. This block is then transmitted to the client. We now derive equations that specify the size of a block, the size of a fragment, the duration of a time period, the amount of memory required by this technique, and the maximum startup latency a client observes with this technique.

To guarantee a hiccup-free display, the display time of a logical block must be equivalent to the duration of a time period:

$$T_p = \frac{\mathcal{B}^l}{R_C} \tag{3}$$

Moreover, if a logical disk services $\mathcal{N}$ simultaneous displays then the duration of each time period must be greater or equal to the service time to read $\mathcal{N}$ fragments from every physical disk i that is part of the logical disk. Thus, the following constraint must be satisfied:

$$T_p \geq \mathcal{N} \times T_{Service_i} = \mathcal{N} \times \left(\frac{\mathcal{B}_i}{R_{D_i}} + T_{Seek_i}\left(\frac{\#cyl_i}{\mathcal{N}}\right) \right) \tag{4}$$

By using Equation 3 to substitute T_p in Equation 4 plus the additional constraint that the time period must be equal for all physical disk drives (K) that constitute a logical disk, we obtain the following equation system (note: we use T_{Seek_i} as an abbreviated notation for $T_{Seek_i}(\frac{\#cyl_i}{\mathcal{N}})$):

$$\mathcal{B}^l = \sum_{i=0}^{K-1} \mathcal{B}_i \tag{5}$$

$$\frac{\mathcal{B}^l}{R_C \times \mathcal{N}} = \frac{\mathcal{B}_0}{R_{D_0}} + T_{Seek_0} = \ldots = \frac{\mathcal{B}_{K-1}}{R_{D_{K-1}}} + T_{Seek_{K-1}} \tag{6}$$

Solving Equations 5 and 6 yields the individual fragment sizes $\mathcal{B}_i$ for each physical disk type i:

$$\mathcal{B}_i = \frac{R_{D_i} \times \left[T_{Seek_i} \times \mathcal{N} \times R_C - \sum_A + \sum_B \right]}{\sum_{z=0}^{K-1} R_{D_z} - R_C \times \mathcal{N}} \tag{7}$$

[3] Fragment sizes are approximately proportional to the disk transfer rates, i.e., the smallest fragment is assigned to the disk with the slowest transfer rate. However, different seek overheads may result in significant variations (see Figure 5).

Disk Model	Fragment Size [KByte]	Fragment display Time [sec]	Number of Fragments	% Space for Traditional Data
Seagate ST32171W	$\mathcal{B}_2$ = 543.4	2.830	3,883	0%
Seagate ST31200W	$\mathcal{B}_1$ = 230.4	1.200	4,471	13.2%
Quantum PD425S	$\mathcal{B}_0$ = 101.4	0.528	4,101	5.3%

Table 3: $\mathcal{N}_{Tot} = 120$ displays of MPEG-1 ($R_C = 1.5$ Mb/s) with grouping, logical block size is 875.2 KByte

with

$$\sum\nolimits_A \overset{\text{def}}{=} \sum\nolimits_{x=0}^{i-1} R_{D_x} \times T_{Seek_i} + \sum\nolimits_{x=i+1}^{K-1} R_{D_x} \times T_{Seek_i}$$

$$\sum\nolimits_B \overset{\text{def}}{=} \sum\nolimits_{y=0}^{i-1} R_{D_y} \times T_{Seek_y} + \sum\nolimits_{y=i+1}^{K-1} R_{D_y} \times T_{Seek_y}$$

To support $\mathcal{N}$ displays with one logical disk, the system requires $2 \times \mathcal{N}$ memory frames. This is because each display requires two buffers: One buffer contains the block whose data is being displayed (it was staged in memory during the previous time period), while a second is employed by the read command that is retrieving the subsequent logical block into memory[4]. The system toggles between these two frames as a function of time until a display completes. As one increases the number of logical disks (m), the total number of displays supported by the system ($\mathcal{N}_{Tot} = m \times \mathcal{N}$) increases, resulting in a higher amount of required memory by the system:

$$M = 2 \times \mathcal{N}_{Tot} \times \mathcal{B}^l \tag{8}$$

In the presence of $\mathcal{N}_{Tot} - 1$ active displays, we can compute the maximum latency that a new request might observe. In the worst case scenario, a new request might arrive referencing a clip X whose first fragment resides on logical disk 0 (ld_0) and miss the opportunity to utilize the time period with sufficient bandwidth to service this request. Due to a round-robin placement of data and activation of logical disks, this request must wait for m time periods before it can retrieve X_0. Thus, the maximum latency L_{max} is:

$$L_{max} = m \times T_p = m \times \frac{\mathcal{B}^l}{R_C} \tag{9}$$

These equations quantify the resource requirement of a system with this technique. They can be employed to conduct an analytical comparison with the other techniques.

2.2 Staggered Grouping

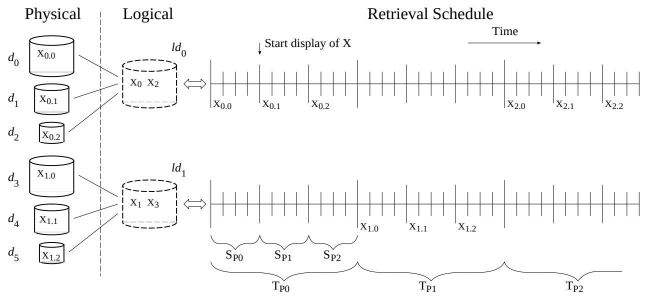


Figure 6: Staggered Grouping

Staggered grouping is an extension of the grouping technique of Section 2.1. It minimizes the amount of memory required to support $\mathcal{N}$ simultaneous displays based on the following observation:

[4]With off-the-shelf disk drives and device drivers, a disk read command requires the address of a memory frame to store the referenced block. The amount of required memory can be reduced by employing specialized device drives that allocate and free memory on-demand [20].

```
Procedure PeakMem ($S_p, R_C, \mathcal{B}_0, \ldots, \mathcal{B}_{K-1}, K$);
    $Consumed = S_p \times R_C$;
    $Memory = Peak = \mathcal{B}_0 + Consumed$;
    For $i$ = 1 to $K - 1$ do
        $Memory = Memory + \mathcal{B}_i - Consumed$;
        If ($Memory > Peak$) then $Peak = Memory$;
    If (($\mathcal{B}_0 + \mathcal{B}_1$) $> Peak$) then $Peak = \mathcal{B}_0 + \mathcal{B}_1$;
    Return($Peak$);
end PeakMem;
```

Figure 7: Dynamic computation for peak memory requirement

Not all fragments of a logical block are needed at the same time. Assuming that each fragment is a contiguous chunk of a logical block, fragment $X_{0.0}$ is required first, followed by $X_{0.1}$ and $X_{0.2}$ during a single time period. Staggered grouping orders the physical disks that constitute a logical disk based on their transfer rate and assumes that the first portion of a block is assigned to the fastest disk, second portion to the second fastest disk, etc. (The explanation for this constrained placement is detailed in the next paragraph.) Next, it staggers the retrieval of each fragment as a function of time. Once a physical disk is activated, it retrieves $\mathcal{N}$ fragments of blocks referenced by the $\mathcal{N}$ active displays that require these fragments, see Figure 6. The duration of this time is termed a sub-period (S_{p_i}) and is identical in length for all physical disks. Due to the staggered retrieval of the fragments, the display of an object X can start at the end of the first sub-period (as opposed to the end of a time period as is the case for disk grouping).

The ordered assignment and activation of the fastest disk first is to guarantee a hiccup-free display. In particular, for slower disks, the duration of a sub-period might exceed the display time of the fragment retrieved from those disks. This is because more data is assigned to the fastest disks of a logical disk to compensate for the slower participating disks. As an example, with the parameters of Table 3, each sub-period is 1.519 seconds long with each disk retrieving 20 fragments (a time period is 4.558 seconds long). The third column of this table shows the display time of each fragment. More data is retrieved from the Seagate ST32171W (with display time of 2.830 seconds) to compensate for the bandwidth of the slower Quantum disk (display time of 0.528 seconds). If the first portion of block X ($X_{0.0}$) is assigned to the Quantum then the client referencing X might suffer from a hiccup after 0.528 seconds of display because the second fragment is not available until 1.519 seconds (one sub-period) later. By assigning the first portion to the fastest disk (Seagate ST32171W), the system prevents the possibility of hiccups. It is important to note that an amount of data equal to a logical block is consumed during a time period.

The amount of memory required by the system is a linear function of the peak memory ($\hat{P}$) required by a display during a time period, $\mathcal{N}$ and m, i.e., $M = \mathcal{N} \times m \times \hat{P}$. For a display, the peak memory can be determined using the simple dynamic computation of Figure 7. This figure computes the maximum of two possible peaks: one that is reached while a display is in progress and a second that corresponds to when a display is first started. Consider each in turn. For an in-progress display, during each time period that consists of K sub-periods, the display is one sub-period behind the retrieval. Thus, at the end of each time period, one sub-period worth of data remains in memory (the variable termed $Consumed$ in Figure 7). At the beginning of a time period, this memory requirement increases by fragment size $\mathcal{B}_0$ because this much memory must be allocated to activate the disk. Next, from this value, it subtracts the amount of data displayed during a sub-period and adds the size of the next fragment, i.e., the amount of memory required to read this fragment. This is repeated for all sub-periods that constitute a time period. In the example of Figure 6 (and with the fragment sizes of Table 3), the peak amount of required memory is 835.2 KBytes which is reached at the beginning of each time period. This compares favorably with the two logical blocks (1.75 MBytes) per display required by disk grouping.

When a retrieval is first initiated, its memory requirement is the sum of the first and second fragment of a block. This is because this technique starts the display of the first fragment at the end of the first sub-period (and the block for the next fragment must be allocated in order to issue the disk read for $\mathcal{B}_1$). The maximum of this value and the peak for a display in progress specifies the memory requirement of a display.

2.3 Disk Merging

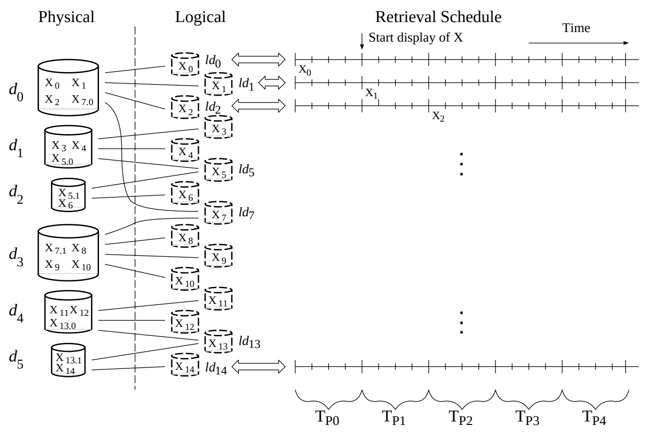


Figure 8: Disk Merging

This technique separates the concept of logical disk from the physical disks all together. Moreover, it forms logical disks from fractions of the available bandwidth and storage capacity of the physical disks, see Figure 8. These two concepts are powerful abstractions that enable this technique to utilize an arbitrary mix of physical disk models. The design of this technique is as follows. First, it chooses how many logical disks should be mapped to each of the *slowest physical disks* and denotes this factor with p_0. For example, in Figure 8, the two slowest disks d_2 and d_5 each represent 1.5 logical disks, i.e., $p_0 = 1.5$. There are tradeoffs associated with choosing p_0 which will be detailed later in this section.

Next, the time period T_p and the block size necessary to support p_0 logical disks on a physical disk can be established by extending Equation 4 and solve for the block size $\mathcal{B}^l$ as follows:

$$T_p \geq \mathcal{N} \times \left(\frac{\mathcal{B}^l \times p_0}{R_{D_0}} + \lceil p_0 \rceil \times T_{Seek_0}\left(\frac{\#cyl_0}{\lceil p_0 \rceil \times \mathcal{N}}\right) \right) \quad (10)$$

$$\mathcal{B}^l = \frac{\mathcal{N} \times R_C \times R_{D_0} \times \lceil p_0 \rceil \times T_{Seek_0}}{R_{D_0} - p_0 \times \mathcal{N} \times R_C} \quad (11)$$

All the logical disks in the system must be identical. Therefore, T_p and $\mathcal{B}^l$ obtained from Equations 10 and 11 determine how many logical disks map to the other, faster disk types i in the system. Each factor p_i, $i \geq 1$, must satisfy the following constraint:

$$\frac{\mathcal{B}^l}{\mathcal{N} \times R_C} = \frac{\mathcal{B}^l \times p_i}{R_{D_i}} + \lceil p_i \rceil \times T_{Seek_i}\left(\frac{\#cyl_i}{\lceil p_i \rceil \times \mathcal{N}}\right) \quad (12)$$

Because of the ceiling-function there is no closed formula solution for p_i from the above equation. However, numerical solutions can easily be found. An initial estimate for p_i may be obtained from the bandwidth ratio of the two different disk types involved. When iteratively refined, this value converges rapidly towards the correct ratio. In the example configuration of Figure 8, the physical disks d_1 and d_4 realize 2.5 logical disks each ($p_1 = 2.5$) and the disks d_0 and d_3 support 3.5 logical disks ($p_2 = 3.5$). Note that fractions of logical disks (e.g., 0.5 of d_0 and 0.5 of d_3) are combined to form additional logical disks (e.g., ld_7, which contains block X_7 in Figure 8). The total number of logical disks (K^l_{Tot}) is defined as[5]:

$$K^l_{Tot} = \left\lfloor \sum_{i=0}^{\#types-1} (p_i \times \#disks_i) \right\rfloor \quad (13)$$

Once a homogeneous collection of logical disks is formed, the block size $\mathcal{B}^l$ of a logical disk determines the total amount of required memory (M), the maximum startup latency (L_{max}), and the total number of simultaneous displays ($\mathcal{N}_{Tot}$). The equations for these are an extension of those described in Section 2.1 and are as follows.

$$\mathcal{N}_{Tot} = \mathcal{N} \times K^l_{Tot} \quad (14)$$

$$M = 2 \times \mathcal{N}_{Tot} \times \mathcal{B}^l \quad (15)$$

$$L_{max} = K^l_{Tot} \times T_p = K^l_{Tot} \times \frac{\mathcal{B}^l}{R_C} \quad (16)$$

The overall number of logical disks constructed heavily depends on the initial choice of p_0. Figure 9 shows possible configurations for a system based on ten disks of type PD425S, ST31200W, and ST32171W each. In Figure 9a, system throughputs are marked in the configuration space for various sample values of p_0. A lower value of p_0 results in fewer, higher performing logical disks. For example, if $p_0 = 1.0$ then each logical disk can support up to 8 displays simultaneously ($\mathcal{N} \leq 8$). Figure 9a shows the total throughput for each $\mathcal{N} = 1, 2, \ldots, 8$. As p_0 is increased, more logical disks are mapped to a physical disk and the number of concurrent streams supported by each logical disk decreases. A value of $p_0 = 8.0$ results in logical disks that can support just one stream ($\mathcal{N} = 1$). However, the overall number of logical disks also roughly increases eight-fold. The maximum value of the product $p_0 \times \mathcal{N}$ is approximately constant[6] and limited by the upper bound of $\frac{R_{D_0}}{R_C}$. In Figure 9 this limit is $\frac{R_{D_0}=13.04Mb/s}{R_C=1.5Mb/s} = 8.69$. Pairs of $\langle p_0, \mathcal{N} \rangle$ that approach this maximum will yield the highest possible throughput (e.g., $\langle 1.7, 5 \rangle$ or $\langle 8.5, 1 \rangle$). However, the amount of memory required to support such a high number of streams increases exponentially (see Figure 9b). The most economical configurations (i.e., between 500 and 650 streams, see Figure 9c) can be achieved with a number of different, but functionally equivalent, $\langle p_0, \mathcal{N} \rangle$ combinations. Because the worst case startup latency is a linear function of the number of logical disks (see Equation 16), configurations with small values of p_0 are preferable. We are currently designing a configuration planner that will find a minimum cost system based on the performance objectives for a given application.

Note that the storage space associated with each logical disk may vary depending onto which physical disk it is mapped. Table 4 shows a sample configuration. Because the round-robin block assignment results in a uniform space allocation of data, the system will be limited by the size of the smallest logical disk. The extra space on the larger logical disks can be used to store traditional data, such as text and records, that can be accessed during off-peak hours. Alternatively, the number of logical disks can manually be decreased on some of the physical disks to equalize the disk space. The latter approach will waste some fraction of the disk bandwidth.

[5] $\#types$ denotes the number of different disk types employed.

[6] It is interesting to note that, because of this property, the perceived improvement of the average seek distance d from $d = \frac{\#cyl}{\mathcal{N}}$ (Equation 4) to $d = \frac{\#cyl}{\lceil p_i \rceil \times \mathcal{N}}$ (Equation 12) does not lower the seek overhead.

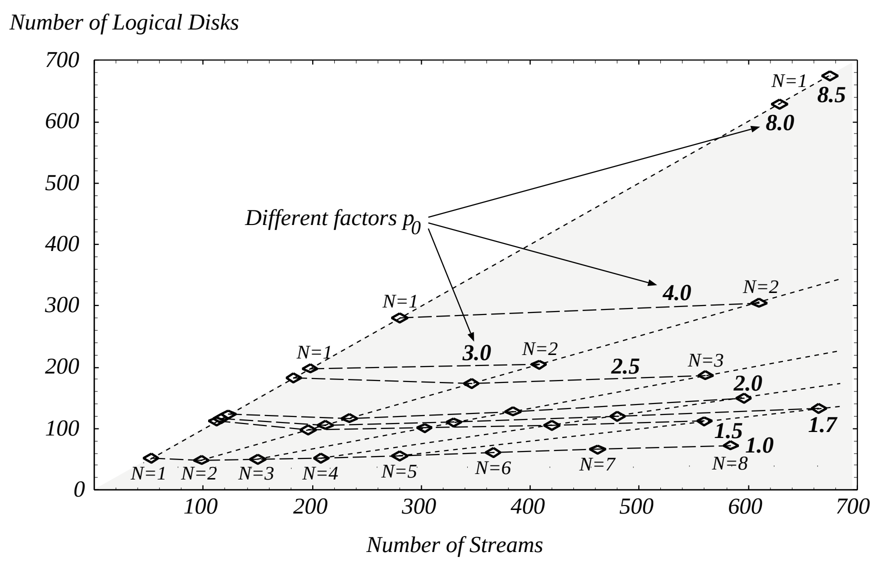


Figure 9a: Configuration space

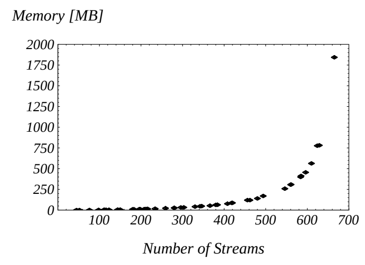


Figure 9b: Memory

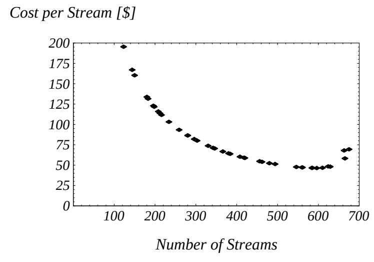


Figure 9c: Cost

Figure 9: Sample system configurations with different choices for p_0 (i.e., the number of logical disks that are mapped to the slowest physical disk type), with 10 disks each of PD425S, ST31200W, and ST32171W

Disk Model	p_i	Logical block size [MBytes]	Logical disk size [MBytes]	% Space for traditional data
Seagate ST32171W	$p_2 = 8.05$		256.0	7.2%
Seagate ST31200W	$p_1 = 3.58$	1.386	281.0	17.7%
Quantum PD425S	$p_0 = 1.7$		238.8	0%

Table 4: $\mathcal{N}_{Tot} = 130$ displays of MPEG-1 ($R_C = 1.5$ Mb/s) with disk merging ($p_0 = 1.7$, $K^l_{Tot} = 26$)

2.4 Multi-Zone Disk Drives

Modern magnetic disk drives feature variable transfer rates due to a technique called zone-bit recording (ZBR) which increases the amount of data being stored on a track as a function of its distance from the disk spindle. Several techniques have been proposed to harness the average transfer rate of the zones [19, 4, 11, 9, 27]. These techniques are orthogonal to disk grouping, staggered grouping, and disk merging. Due to lack of space we will describe only one approach and why its design is orthogonal. Assuming that the number of tracks in every zone is a multiple of some fixed number, [19] constructs Logical Tracks (LT) from the same numbered physical track of the different zones. The order of tracks in a LT is by zone number. When displaying a clip, the system reads a LT on its behalf per time period. This forces the disk to retrieve data from the constituting physical tracks in immediate succession by zone order. An application observes a constant disk transfer rate for each LT retrieval. LTs are employed in disk grouping and staggered grouping as follows. With these techniques, each logical block is declustered across the participating physical disks into fragments. These fragments are then assigned to one or more LTs on each of the physical disks according to the size of the fragments. Similarly, for disk merging the logical blocks are assigned to LTs.

2.5 An Analytical Comparison

We compared the three non-partitioning techniques with each other using three different disk subsystem configurations consisting of a different number of disks:

1. 15 Quantum PD425S, 10 Seagate ST31200W, and 5 Seagate ST32171W disk drives, termed 15-10-5 configuration
2. 10 disks of each type, termed 10-10-10 configuration
3. 5 PD425S, 10 ST31200W, and 15 ST32171W disks, termed 5-10-15 configuration

We used the analytical models of Sections 2.1, 2.2, and 2.3 to compute the memory requirement (M), the average-minimum startup latency ($L_{avg-min} = \frac{T_p}{2}$)[7], the maximum startup latency (L_{max}), and the maximum number of streams supported ($\mathcal{N}_{Tot}$) with each technique. The transfer rate R_D used for a each disk drive type is listed in Table 1 and the consumption rate R_C per stream was set to 1.5 Mb/s (e.g., MPEG-1).

For a service provider of a video-on-demand system, the cost per stream of a configuration might be an important consideration. For each technique, we compute the cost per stream assuming $8 per MByte of memory and a fixed cost of $800 per disk drive. The obtained results and the final observations remain unchanged if a lower disk price is assumed.

Figure 10 shows the cost per stream and the range of startup latency incurred with each, the 15-10-5, the 10-10-10, and the 5-10-15 configuration for disk grouping, staggered grouping, and disk merging. Note that the maximum number of streams that each technique can support with a given platform is approximately the same because the limiting factor is the aggregate bandwidth of the disk subsystem. However, in general, both disk grouping and staggered grouping can support a slightly higher number of displays (at a very high cost per stream).

In Figures 10a, 10c, and 10e, the cost per stream decreases as a function of $\mathcal{N}$ at first because: (1) the fixed disk cost is divided across a larger number of streams, and (2) the block sizes are small, reducing the total memory requirement of the system by wasting a significant percentage of the disk bandwidth. To illustrate, Fig-

[7] On a very lightly loaded system a request can be started at the beginning of the next time period. Depending on the request's arrival time within the current time period, its wait time is $0 < L_{min} \leq T_p$, or on average $\frac{T_p}{2}$.

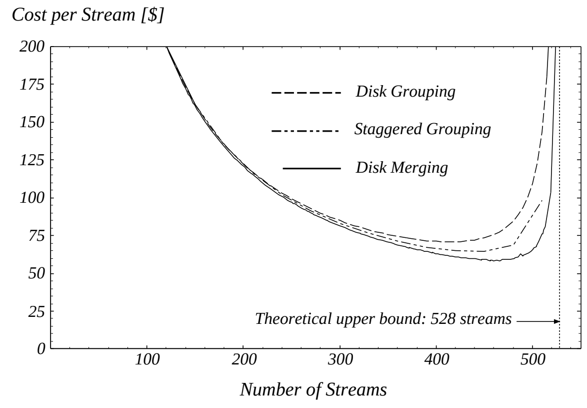


Figure 10a: Cost per stream (15-10-5)

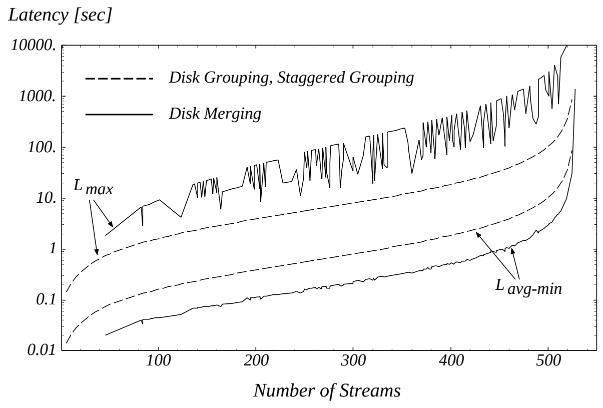


Figure 10b: Startup latency range (15-10-5)

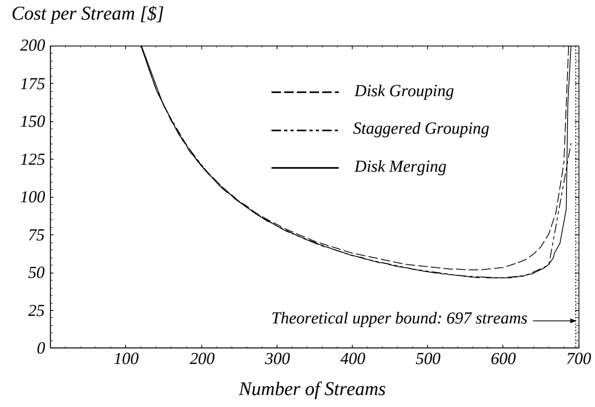


Figure 10c: Cost per stream (10-10-10)

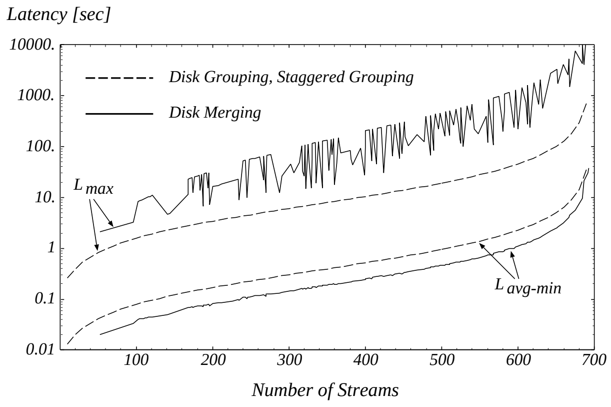


Figure 10d: Startup latency range (10-10-10)

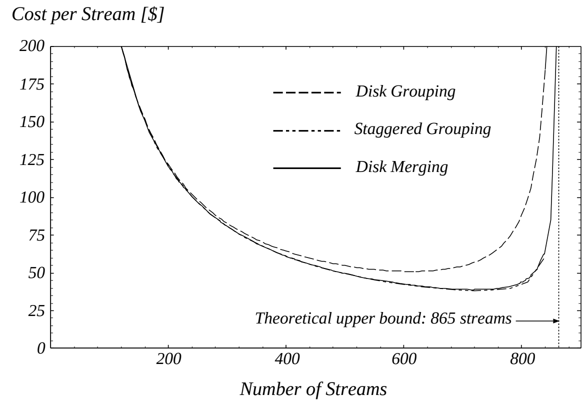


Figure 10e: Cost per stream (5-10-15)

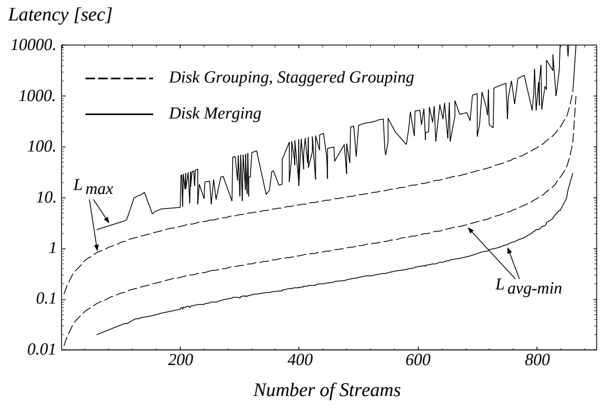


Figure 10f: Startup latency range (5-10-15)

Figure 10: Cost per stream, average-minimum and maximum startup latency for three different disk subsystem configurations (MPEG-1 streams with $R_C = 1.5$ Mb/s)

ure 11 shows the memory and disk cost per stream as a function of $\mathcal{N}$. As the number of streams is increased, the block size becomes larger, reducing the impact of disk seek time and rotational latency delays. Beyond a certain value of $\mathcal{N}$ (e.g., 560 in Figure 11), the disk bandwidth becomes a scarce resource. In order to support more displays, the system must assume very large blocks to further reduce the overhead associated with disk seeks and rotational delays. However, increased block sizes directly translate into a higher memory requirement. This renders the memory cost significant, increasing the cost per stream.

The disk merging technique results in a higher variance for the startup latency of a stream. In Figures 10b, 10d, and 10f the time period T_p is between two to three times shorter for disk merging than for the two other techniques. This is reflected in the reduced average minimum startup latency $L_{avg-min}$ (note the logarithmic scale). At the same time, the maximum latency is also longer by a factor of 2 to 30. The variability of the L_{max} curve stems from the fact that configurations providing almost identical throughput may be based on a different number of logical disks. An increased number of logical disks directly translates into a longer worst case latency. A configuration planner could easily select the configura-

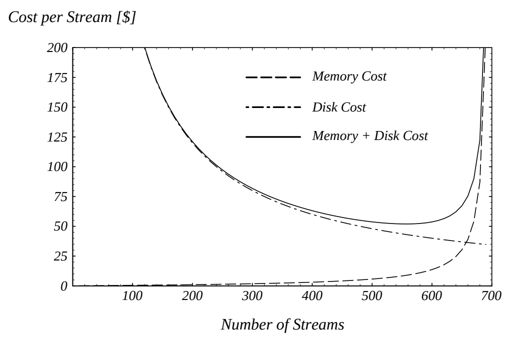


Figure 11: Example cost breakdown into memory and disk components (10-10-10)

tions that result in a lower L_{max}. In general, simulations suggest that a reduced time period T_p is desirable because the expected startup latency is typically a small multiple of T_p for most of a server's operating range (e.g., $\leq$ 90% of full capacity, see [16]). The startup latency only approaches L_{max} when a server reaches 100% utilization, i.e., when the system is close to being overloaded.

Overall, disk merging is the most promising technique among the three non-partitioning strategies.

3 An Experimental Comparison

We compared disk merging with a simple partitioning strategy to quantify their tradeoffs. We utilized a simulation model because it is difficult to develop analytical models for the partitioning scheme. The simulator was configured with a total of 30 disks in a 10-10-10 configuration (Quantum PD425S, Seagate ST31200W, Seagate ST32171W). To simulate the partitioned system, each group of 10 homogeneous disk drives was treated as an individual server (termed Subserver 0, Subserver 1, and Subserver 2). For each subserver the block size $\mathcal{B}$ and the time period T_p were calculated to yield the lowest cost per stream. Table 5a summarizes the resulting parameters.

Disk merging assumes a logical to physical disk ratio for the slowest disks (Quantum PD425S) of $p_0 = 1.1$. The 10 Seagate ST31200W disks realize 24 logical disks and the 10 Seagate ST32171W support 51 logical drives for a total of 86 logical disks. The maximum number of streams supported by an individual logical disk was $\mathcal{N} = 7$ with a block size of 444 KByte. This resulted in a maximum throughput of $\mathcal{N}_{Tot} = 602$ streams. This configuration was chosen because its maximum throughput was comparable to the partitioned scheme. The slightly higher memory requirement (523 MBytes vs. 465 MBytes) resulted in a cost per stream that was marginally increased (1.3%) over the partitioned system (see Table 5b). Since the cost per stream is almost identical for the two systems, we will focus on the average startup latency to compare the two techniques.

The simulated database consisted of video clips whose display was one hour long and required a constant bit rate of 1.5 Mb/s (e.g., MPEG-1). This resulted in a uniform storage requirement of 675 MByte per clip. For disk merging, the Quantum PD425S disks limits the storage capacity for continuous media, rendering only 31.74 GByte (out of 34 GByte) available that can store 47 video clips. The partitioned system can take advantage of all the storage space in the system and store 50 clips. However, to have a system comparable to disk merging we chose to store the same 47 clips. The additional space for three clips can be used to replicate some of the objects to reduce the sensitivity of the system to access pattern fluctuations.

The frequency of access to different media clips is usually quite skewed for a video-on-demand system, i.e., a few newly released movies are very popular while most of the rest are accessed infrequently. Furthermore, the set of popular movies may change over the course of a day. For example, movies that appeal to children may register their peak access during the afternoon while other movies may be popular with adults in the evening. The long term (i.e., weekly) distribution pattern can be modeled using Zipf's law [32], which defines the access frequency of movie i to be $F(i) = \frac{c}{i^{1-d}}$, where c is a normalization constant and d controls how quickly the access frequency drops off. In our simulations, d is set to equal 0.271. This value has been shown to approximate empirical data for rental movies [6]. In our experiments we focus on the the short term (i.e., daily) changes because we consider them the most critical[8]. Table 6 shows how the shifting workload is modeled with a simple increase ($+5\%$) of the total number of requests directed towards Subservers 0 and 1 and a proportional (-10%) decrease for Subserver 2 at half time of a 24 hour experiment. The resulting Zipf distributions for the 47 movies in the database are detailed in Figure 12. The left- and right-most graphs show the two identical access frequency assignments at the beginning and at the end of the 24 hour experiment. After 12 hours (the middle graph) the accesses are still Zipf distributed; however, the popularity of eight movies has either increased or decreased. Intermediate points were linearly interpolated between these three distributions.

	Percentage of Accesses		
	At 0h	After 12h	After 24h
Subserver 0	13.3%	18.3% ($+5\%$)	13.3%
Subserver 1	26.7%	31.7% ($+5\%$)	26.7%
Subserver 2	60.0%	50.0% (-10%)	60.0%

Table 6: Access distribution among subservers for partitioning (10-10-10 configuration)

For the partitioned system, the clips were initially assigned to the individual subservers with a simple greedy algorithm as follows. Each subserver j is initialized with a fraction of the total streaming capacity of the system, denoted SC_j. For example, Subserver 1 (SC_1) provides for 26.7% ($\frac{160}{80+160+360} = 0.267$) of the system capacity. In the first step, the most popular clip is assigned to subserver j with the highest value SC_j. Next, SC_j is decreased by the access frequency of the clip, $F(i)$. This process is repeated for the second most popular clip, which is assigned to the subserver i with the highest SC_i value remaining. This is repeated until all clips are placed. When a subserver runs out of space, it is removed from a list of the candidate servers for the remaining objects. The above algorithm assigns 6 clips to Subserver 0, 14 to Subserver 1, and 27 to Subserver 2. This simple strategy resulted in a slightly non-optimal placement of the clips (e.g., Subserver 1 was assigned 0.263, not 0.267, of the streaming capacity). Therefore, the clip placement was manually adjusted to produce the optimal layout listed in Table 6 with Subservers 0, 1, and 2 servicing 13.3%, 26.7%, and 60.0% of the requests, respectively. The second column of Table 6 shows how the redistribution of the clip popularity affected the workload for each subserver after 12 hours. The fraction of the requests serviced for both, Subserver 0 and 1, temporarily increased by 5% whereas it declined by 10% for Subserver 2.

For each experiment, the server had to service requests that arrived based on a Poisson distribution over a time period of 24 hours. The request arrival rates of 420, 480, and 540 requests per hour represented system loads of approximately 70%, 80%, and 90%. If requests arrived when the server was completely utilized, then they were queued and serviced using a FCFS policy.

Figure 13 shows the results of the simulations with partitioning in the left and disk merging in the right column of graphs. Figures 13a and 13b depict the impact of an increased workload on the average startup latency that a request experienced (note the logarithmic y-axis). For arrival rate of 420 requests per hour, the partitioned system still services the requests with a low latency most of the time, except for the time between 13 and 16 hours. At an increased rate

[8]Load-balancing algorithms should be quite effective in countering the effects of long term changes to the popularity of movies and their overhead may be amortized over an extended time.

Partitioning	Subserver 0 (PD425S)	Subserver 1 (ST31200W)	Subserver 2 (ST32171W)	Total
Block size $\mathcal{B}$	350 KByte	296 KByte	452 KByte	
Time period T_p	1.824 sec	1.542 sec	2.354 sec	
Throughput $\mathcal{N}_{Sub}$	80	160	360	600
No. of stored clips	6	14	27	47
Memory				465 MB
Cost per stream				\$46.2

Table 5a: Partitioning (10-10-10)

Disk Merging	86 Log. Disks ($p_0 = 1.1$)
Block size $\mathcal{B}$	444 KByte
Time period T_p	2.315 sec
Throughput $\mathcal{N}_{Tot}$	602
No. of stored clips	47
Memory	523 MB
Cost per stream	\$46.8

Table 5b: Disk merging (10-10-10)

Table 5: Parameters for a partitioned and a non-partitioned system

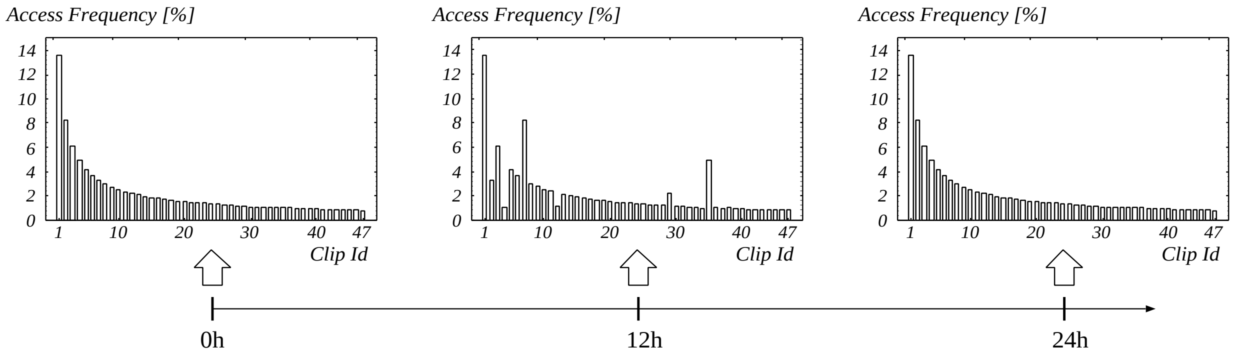


Figure 12: Access frequency distribution to the 47 clips that constitute the experimental database over a time span of 24 hours according to Zipf's law: $F(i) = \frac{c}{i^{1-d}}$ with with $d = 0.271$

of 480 requests per hour, the service degradation starts earlier and is more pronounced. The reason is shown in Figure 13c: Subserver 0 becomes a bottleneck (i.e., the utilization reaches 100%) for an extended period of time due to the shifting workload. Still, with this load the system can recover towards the end of the 24 hour experiment. At an arrival rate of 540 requests per hour the average latency becomes extremely long and—more critical—extended queues are now forming for Subserver 0 and 1, preventing the system from recovering, even after the workload shifts back to its optimum. At the same time, Subserver 2 has idle capacity. The performance of the system that utilizes disk merging is shown in Figures 13b and 13d for the same workloads. Because of the inherent load-balancing property of striping, the system is affected very little by the shifting workload. The latency increases moderately to 10-20 seconds maximum for very high utilization (i.e., 90%).

When compared with partitioning, the primary advantage of the non-partitioning scheme becomes evident: disk merging is not sensitive to the distribution of accesses or placement of data. In particular, with the partitioning scheme, the system risks the possibility of the slowest server becoming the bottleneck for the entire system, if even a modest workload imbalance occurs.

Some of these problems might be alleviated for a partitioned system by integrating data replication and dynamic load-balancing. However, algorithms that automatically respond to changing workloads have the disadvantage of (a) being detective instead of preventive, i.e., the system must detect undesirable situations and then try to remedy them, and (b) adding to the complexity of the server software. Such an approach will most likely result in additional resource requirements and less than optimal performance at least some of the time (see [10]).

4 Conclusion and Future Research Directions

This paper described three different non-partitioning techniques that guarantee a continuous display with a heterogeneous disk subsystem. These techniques are disk grouping, staggered grouping, and disk merging. Disk merging is more flexible than the other two because it can support an arbitrary mix of physical disks. In our experiments, it resulted in a competitively low cost per stream. When compared with a partitioning scheme, disk merging is sensitive to neither the frequency of access to clips nor their assignment. Thus, it prevents the possibility of a group of disks becoming a bottleneck and determining the overall performance of a server.

We intend to extend disk merging in two ways. First, we are designing a configuration planner that determines system parameters (the logical disk specifications, block size, amount of required memory) to meet the performance objectives of an application while minimizing system cost. Second, we are investigating availability techniques and how the system can continue operation in the presence of physical disk failures. Consider each topic in turn. First, parity based techniques have been investigated to ensure availability to data [3, 22]. With these techniques, two or more logical disks that map onto a single physical disk cannot be part of a single parity group. Otherwise, a single disk failure might render the data unavailable. In essence, the mapping from the logical to physical disks imposes constraints on the formation of parity groups and the id chosen for a logical disk. Second, in the presence of disk failure, we intend to quantify how many displays might observe a hiccup and whether the system might be forced to abort some active display all together.

References

[1] D. Anderson, Y. Osawa, and R. Govindan. Real-time disk storage and retrieval of digital audio and video. UCB/ERL Technical Report M91/646, UC Berkeley, 1991.

[2] S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju. Staggered Striping in Multimedia Information Systems. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1994.

[3] S. Berson, L. Golubchik, and R. R. Muntz. Fault Tolerant Design of Multimedia Servers. In *Proceedings of the ACM*

Partitioning **Disk merging**

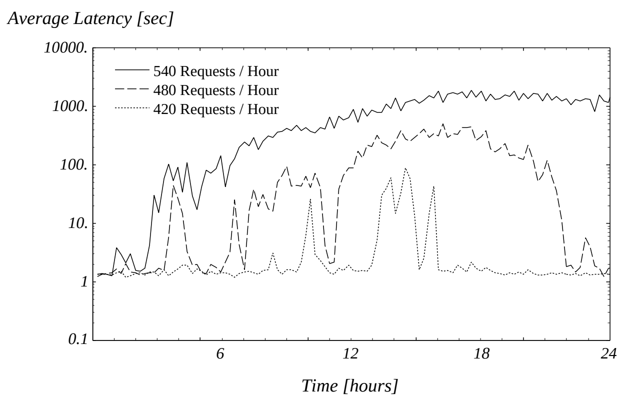


Figure 13a: Average startup latency

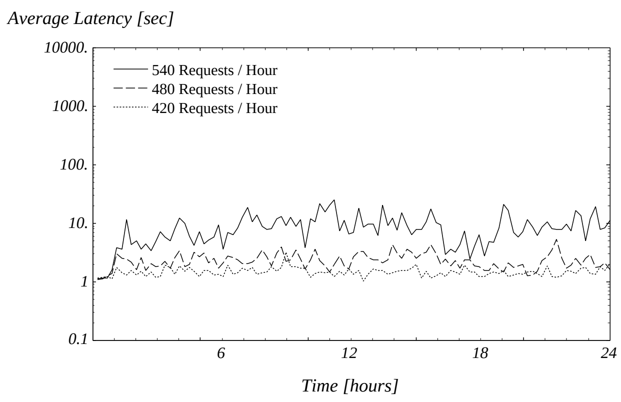
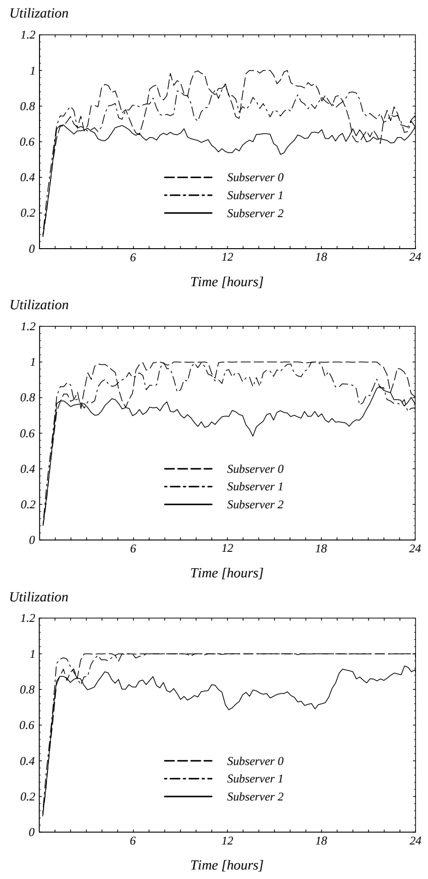


Figure 13b: Average startup latency



420 req/h

480 req/h

540 req/h

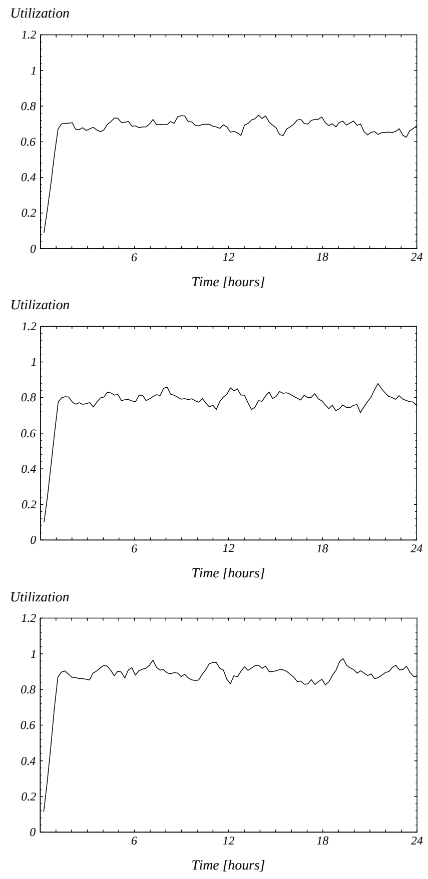


Figure 13c: Subserver utilization

Figure 13d: Server utilization

Figure 13: Average startup latency and system utilization for a partitioning and a disk merging system configuration

SIGMOD International Conference on Management of Data, 1995.

[4] Yitzhak Birk. Track-Pairing: a Novel Data Layout for VOD Servers with Multi-Zone-Recording Disks. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 248–255, 1995.

[5] Ling Tony Chen, Doron Rotem, and Sridhar Seshadri. Declustering Databases on Heterogeneous Disk Systems. In *Proceedings of the 21st International Conference on Very Large Data Bases*, pages 110–121, Zürich, Switzerland, September 1995.

[6] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling Policies for an On-Demand Video Server with Batching. In *Proceedings of the ACM Multimedia*, pages 391–398, 1994.

[7] Asit Dan and Dinkar Sitaram. An Online Video Placement Policy based on Bandwidth to Space Ratio (BSR). In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 376–385, San Jose, May 1995.

[8] R. Flynn and W. Tetzlaff. Disk Striping and Block Replication Algorithms for Video File Servers. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 590–597, Hiroshima, Japan, June, 17-23 1996.

[9] S. Ghandeharizadeh, D. Ierardi, D. H. Kim, and R. Zimmermann. Placement of Data in Multi-Zone Disk Drives. In *Second International Baltic Workshop on DB and IS*, June 1996.

[10] S. Ghandeharizadeh, D.J. Ierardi, and R. Zimmermann. Management in Hierarchical Storage Systems. Technical Report USC-CS-TR94-598, University of Southern California, 1994.

[11] S. Ghandeharizadeh, S. H. Kim, C. Shahabi, and R. Zimmermann. Placement of Continuous Media in Multi-Zone Disks. In Soon M. Chung, editor, *Multimedia Information Storage and Management*, chapter 2. Kluwer Academic Publishers, Boston, August 1996. ISBN: 0-7923-9764-9.

[12] S. Ghandeharizadeh and L. Ramos. Continuous retrieval of multimedia data using parallelism. *IEEE Transactions on Knowledge and Data Engineering*, 5(4), August 1993.

[13] S. Ghandeharizadeh and C. Shahabi. Management of Physical Replicas in Parallel Multimedia Information Systems. In *Proceedings of the Foundations of Data Organization and Algorithms (FODO) Conference*, October 1993.

[14] S. Ghandeharizadeh, J. Stone, and R. Zimmermann. Techniques to Quantify SCSI-2 Disk Subsystem Specifications for Multimedia. Technical Report USC-CS-TR95-610, USC, 1995.

[15] S. Ghandeharizadeh, R. Zimmermann, W. Shi, R. Rejaie, D. Ierardi, and T.W. Li. Mitra: A Scalable Continuous Media Server. *Kluwer Multimedia Tools and Applications*, 5(1):79–108, July 1997.

[16] Shahram Ghandeharizadeh, Seon Ho Kim, Weifeng Shi, and Roger Zimmermann. On Minimizing Startup Latency in Scalable Continuous Media Servers. *Proceedings of Multimedia Computing and Networking 1997 Conference (MMCN'97)*, February 1997.

[17] Ed Grochowski. Disk drive price decline, 1997. IBM Almaden Research Center, San Jose, CA, URL: http://www.storage.ibm.com/storage/technolo/grochows.

[18] Ed Grochowski. Internal (media) data rate trend, 1997. IBM Almaden Research Center, San Jose, CA, URL: http://www.storage.ibm.com/storage/technolo/grochows.

[19] S. R. Heltzer, J. M. Menon, and M. F. Mitoma. Logical Data Tracks Extending Among a Plurality of Zones of Physical Tracks of one or More Disk Devices., April 1993. U.S. Patent No. 5,202,799.

[20] R.T. Ng and J. Yang. Maximizing Buffer and Disk Utilizations for News On-Demand. In *Proceedings of the International Conference on Very Large Databases*, September 1994.

[21] B. Özden, R. Rastogi, and A. Silberschatz. Disk Striping in Video Server Environments. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 580–589, Hiroshima, Japan, June, 17-23 1996.

[22] Banu Özden, Rajeev Rastogi, Prashant Shenoy, and Avi Silberschatz. Fault-tolerant Architectures for Continuous Media Servers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 79–90, June 1996.

[23] David A. Patterson. Terabytes $\gg$ Teraflops (Or why work on processors when I/O is where the action is?), May 13, 1993. Keynote address at the ACM SIGMETRICS Conference in Santa Clara, CA.

[24] P. Rangan and H. Vin. Efficient Storage Techniques for Digital Continuous Media. *IEEE Transactions on Knowledge and Data Engineering*, 5(4), August 1993.

[25] C. Ruemmler and J. Wilkes. An Introduction to Disk Drive Modeling. *IEEE Computer*, pages 17–28, March 1994.

[26] K. Salem and H. Garcia-Molina. Disk striping. In *Proceedings of International Conference on Database Engineering*, February 1986.

[27] R. Tewari, R. P. King, D. Kandlur, and D. M. Dias. Placement of Multimedia Blocks on Zoned Disks. In *Proceedings of IS&T/SPIE Multimedia Computing and Networking*, San Jose, January 1996.

[28] B. Tierney, B. Johnston, H. Herzog, G. Hoo, G. Jin, J. Lee, T. Chen, and D. Rotem. Distributed Parallel Data Storage Systems: A Scalable Approach to High Speed Image Servers. In *Second ACM Conference on Multimedia*, pages 399–405, San Francisco, October 1994.

[29] F.A. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming RAID-A Disk Array Management System for Video Files. In *First ACM Conference on Multimedia*, August 1993.

[30] J.L. Wolf, P.S. Yu, and H. Shachnai. DASD Dancing: A Disk Load Balancing Optimization Scheme for Video-on-Demand Computer Systems. In *Proceedings of the ACM SIGMETRICS*, Ottawa, Canada, May 1995.

[31] P.S. Yu, M-S. Chen, and D.D. Kandlur. Grouped sweeping scheduling for DASD-based multimedia storage management. *Multimedia Systems*, 1(1):99–109, January 1993.

[32] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Reading MA, 1949.