Striping in multi-disk video servers[*]

Shahram Ghandeharizadeh and Seon Ho Kim

Department of Computer Science
University of Southern California
Los Angeles, California 90089

ABSTRACT

A challenging task when designing a video server is to support the performance criteria of its target application, i.e., both its desired number of simultaneous displays and the waiting tolerance of a display. With a multi-disk video server, the placement of data has a significant impact on the overall performance of a system. This study quantifies the tradeoffs associated with alternative organization of data across multiple disks. We describe a planner that configures a system to: 1) support the performance criteria of an application, and 2) minimize the cost of a system. The experimental results demonstrate the superiority and flexibility of using the planner to configure a system.

Keywords: striping, data placement, video server, latency, throughput

1 INTRODUCTION

During the past decade, the information technology has evolved to where it is economically viable to store and retrieve video objects. While the entertainment industry envisions the use of video servers as a superior alternative to current video stores, these servers are expected to play a major role in educational applications, library information systems, etc. A server must retrieve a video object at a pre-specified rate in order to ensure its continuous display. Otherwise, if special precautions are not taken then the display of an object may suffer from frequent disruptions and delays, termed hiccups. To support a hiccup-free display, it may not be practical to stage an object in memory in its entirety before initiating its display. This is because video objects are large in size. For example, a 30 minute uncompressed video clip based on NTSC is 10 gigabytes in size. With a lossy compression technique that reduces the bandwidth requirement of this object to 1.5 Mb/s, this object is 337 megabytes in size. Assuming the bandwidth required to display an object (R_C) is lower than the bandwidth of a disk[a]drive (R_D), a superior approach to display an object X is as follows. First, the system partitions the object into equi-sized blocks. Once an object X is referenced, the system stages its first block in memory and initiates its display. Prior to completion of the display of the first block of X, the system stages the next block of X in memory to support a hiccup-free display. Since $R_D > R_C$, the bandwidth of the disk can be multiplexed to support several simultaneous displays. Variations of this simple technique have been described in.[12,2,13,17,14,1]

[*]This research was supported in part by the National Science Foundation under grants IRI-9203389, IRI-9258362 (NYI award), and CDA-9216321, and a Hewlett-Packard unrestricted cash/equipment gift.

[a]This assumption is relaxed in Section 3.4.

Term	Definition
R_D	Disk bandwidth in megabits per second, Mb/s
$\#cyl$	Number of cylinders in a disk drive
T_{W_Seek}	Maximum seek time of a disk drive, in seconds (including the maximum rotational latency time)
$\mathcal{B}$	Size of a block, in megabits (Mbits)
D	Number of disks in the system
d	Number of disks per cluster
k	Number of disk clusters in a system, $k = \lfloor \frac{D}{d} \rfloor$
R_C	Bandwidth required to support a display in Mb/s
n	Number of blocks of an object X, $\lceil \frac{size(X)}{\mathcal{B}} \rceil$
Mem	Available memory in megabits (Mbits)
T_p	Time period, in seconds
$\mathcal{N}$	Maximum number of simultaneous displays (throughput) supported by one disk
$\mathcal{N}_{desired}$	Desired throughput for a given application
ℓ	Maximum latency time for a system, in seconds
$\ell_{desired}$	Maximum latency time tolerated by an application, in seconds

Table 1: List of terms used repeatedly in this paper and their respective definitions

In addition to ensuring a continuous display, a system designer must consider the performance requirements of an application. In an ideal environment, a system costs zero dollars, serves an infinite number of displays with each display observing a zero latency time. (Latency time is defined as the amount of time elapsed from the arrival time of a request to the onset of the display of its referenced object.) However, such a system is infeasible, motivating one to design a system based on the number of simultaneous displays desired by the target application and the waiting tolerance of a display. Different applications may require different performance criteria. For example, a video-on-demand system designed for a city may expect to have thousands of simultaneous displays as its maximum load. For such a system, it might be reasonable to assume that a client at home can tolerate a latency in the order of minutes previewing commercial clips that reduce cost. Alternatively, a system targeted for a hotel may expect to support hundreds of simultaneous displays as its maximum load with each display tolerating a latency in the order of seconds.

The bandwidth of a single disk drive might be insufficient to support the number of simultaneous displays desired by an application. One may assume a multi-disk platform consisting of D disks to resolve this limitation. With such a platform, the workload of a display should be evenly distributed across the D disks in order to avoid formation of bottlenecks. Striping[1] is a technique to accomplish this objective. This technique partitions the D disks into k clusters of disks with each cluster consisting of d disks: $k = \lfloor \frac{D}{d} \rfloor$. Next, it assigns the blocks of object X to the clusters in a round-robin manner. The first block of X is assigned to an arbitrarily chosen disk cluster. Each block of an object is declustered[8] across the d disks that constitute a cluster. For example, in Figure 1, a system consisting of six disks is partitioned into three clusters, each consisting of two disk drives. The assignment of the blocks of X starts with cluster 0. This block is declustered into two fragments: $X_{0.0}$ and $X_{0.1}$. When a request references object X, the system employs the idle slot on the cluster that contains X_0 (say C_i) to display its first block. Before the display of X_0 completes, the system employs cluster $C_{(i+1)\ mod\ k}$ to display X_1. This process is repeated until all blocks of an object have been retrieved and displayed.

This study is a contribution because it demonstrates that the size of a striping unit (i.e., number of disks activated to read a block and the block size) should be chosen based on the performance criteria of a target application. The discussion of[1] does not address the performance requirements of an application and chooses d based on the bandwidth required to display an object, e.g., if the bandwidth required to display an object is less than one disk then d=1. Given a system with D disks, the value chosen for d has a significant impact on: 1) the maximum number of simultaneous displays supported by the system, 2) the maximum latency observed by a

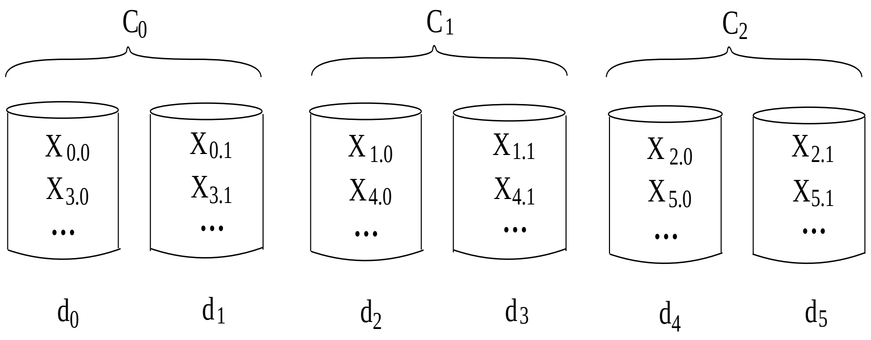


Figure 1: Simple Striping

display, and 3) the amount of memory required by the system. In addition to quantifying the tradeoff associated with these factors, we describe a configuration planner that consumes the performance criteria of an application (its desired number of simultaneous displays and latency) to compute system parameters: D, d, required memory, etc. The planner computes these parameters with the objective to minimize system cost. Our experimental results demonstrate the flexibility of using this planner.

The rest of the paper is organized as follows. Section 2 describes a technique to support continuous display of multiple objects using a single disk drive. It outlines analytical models to establish two fundamental relationships, namely, those of: 1) throughput vs. latency time, and 2) disk bandwidth vs. memory requirement of an application. These models extend naturally to a multi-disk platform, providing the foundation for Section 3 to describe the tradeoffs associated with choosing a cluster size. Section 3 also details the configuration planner. Section 4 demonstrates the flexibility of the planner and how it trades memory for disk bandwidth to configures the cheapest possible system while satisfying the performance requirements of an application. Section 5 offers brief conclusions and our future research directions.

2 CONTINUOUS DISPLAY USING A SINGLE DISK

In this paper, we make the following simplifying assumptions;

1. The system is configured with a fixed amount of memory and a single disk drive. The disk drive has a fixed transfer rate (R_D) and provides a large storage capacity (more than one gigabyte). An example disk drive from the commercial arena is Seagate Barracuda 2-2HP that provides a 2 Gigabyte storage capacity and a minimum transfer rate of 68.6 Megabits per second (Mb/s).[15]
2. The objects that constitute the video server belong to a single media type and require a fixed bandwidth for their display (R_C).
3. $R_D > R_C$. Relaxation of this assumption is straightforward and described in Section 3.4.
4. A multi-user environment requiring simultaneous display of objects to different users.

To support continuous display of an object X, it is partitioned into n equi-sized blocks: X_0, X_1, ..., X_{n-1}, where n is a function of the block size ($\mathcal{B}$) and the size of X (see Table 1). A *time period* (T_p) is defined as the time required to display a block:

$$T_p = \frac{\mathcal{B}}{R_C} \tag{1}$$

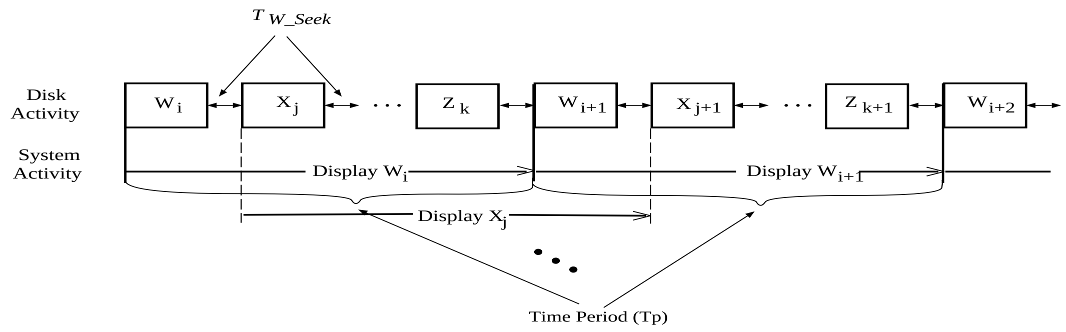


Figure 2: Time Period

When an object X is referenced, the system stages X_0 in memory and initiates its display. Prior to completion of a time period, it initiates the retrieval of X_1 into memory in order to ensure a continuous display. This process is repeated until all blocks of an object have been displayed.

To support simultaneous display of several objects, a time period is partitioned into fixed-size slots, with each slot corresponding to the retrieval time of a block from the disk drive. The number of slots in a time period defines the number of simultaneous displays that can be supported by the system. For example, a block size of 750 Kbytes corresponding to a MPEG-1 compressed movie ($R_C = 1.5$ Mb/s) has a 4 second display time ($T_p = 4$). Assuming a typical magnetic disk with a transfer rate of 24 Mb/s ($R_D = 24$ Mb/s) and maximum seek time of 35 milliseconds, 14 such blocks can be retrieved in 4 seconds. Hence, a single disk supports 14 simultaneous displays. Figure 2 demonstrates the concept of a time period and a time slot. Each box represents a time slot. Assuming that each block is stored contiguously on the surface of the disk, the disk incurs a seek every time it switches from one block of an object to another. We denote this as T_{W_Seek} and assume that it includes the maximum rotational latency time of the disk drive. We will not discuss rotational latency further because it is a constant added to every seek time.

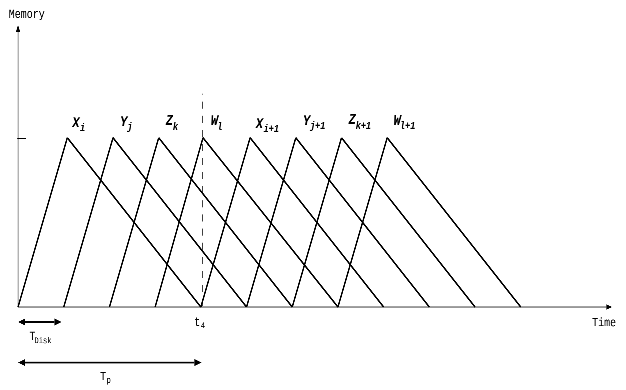
To display $\mathcal{N}$ simultaneous blocks per time period, the system should provide sufficient memory for staging the blocks. Assume the system maintains a free pool of memory frames with the size of each frame corresponding to that of a block. If the $\mathcal{N}$ active displays compete for the available frames then the memory requirement of the system is $(\mathcal{N}+1)\mathcal{B}$.[18] An optimistic technique as described in[11] estimates the amount of required memory as $\frac{\mathcal{N}\mathcal{B}}{2}$ to support $\mathcal{N}$ simultaneous displays. To observe this, Figure 3a shows the memory requirements of each display as a function of time for a system that supports four simultaneous displays. A time period is partitioned into 4 slots. The duration of each slot is denoted T_{Disk}. During each T_{Disk} for a given object (e.g., X), the disk is producing data while the display is consuming it. Thus, the amount of data staged in memory during this period is lower than $\mathcal{B}$. (It is $T_{Disk} \times R_D - T_{Disk} \times R_C$, however, we assume $\mathcal{B}$ in order to simplify discussion.) Consider the memory requirement of each display for one instant in time, say t_4: X requires no memory, Y requires $\frac{\mathcal{B}}{3}$ memory, Z requires $\frac{2\times\mathcal{B}}{3}$ memory, and W requires at most $\mathcal{B}$ memory[b]. Hence the total memory requirement for these four displays is $2\mathcal{B}$ (i.e., $\frac{\mathcal{N}\mathcal{B}}{2}$); we refer the interested reader to[11] for the complete proof. We arbitrarily chose this optimistic approach to compute the memory requirements of a server. Hence, if Mem denotes the amount of configured memory for a system, then the following constraint must be satisfied:

$$\frac{\mathcal{N}\times\mathcal{B}}{2} \leq Mem \tag{2}$$

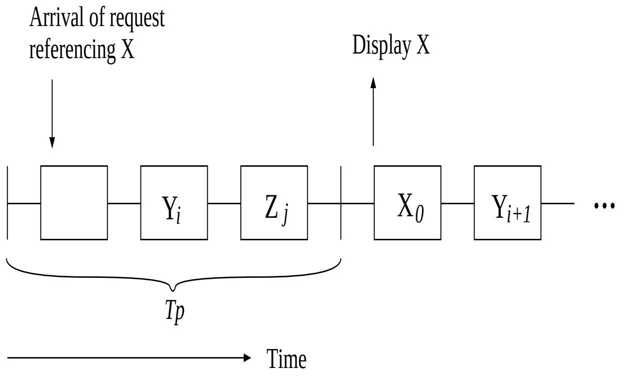
To compute the size of a block, from Figure 2, it is trivial that:

$$\mathcal{B} = (\frac{T_p}{\mathcal{N}} - T_{W\_Seek}) \times R_D \tag{3}$$

[b]One may choose an alternative technique that results in a different amount of memory with no impact on the underlying methodology.



a. Memory requirement for four streams



b. Maximum latency for a request referencing X

Figure 3:

By substituting $\mathcal{B}$ from Equation 3 into Equation 1 we obtain:

$$T_p = \mathcal{N} \times T_{W\_Seek} \times \frac{R_D}{R_D - (\mathcal{N} \times R_C)} \tag{4}$$

The duration of a time period (T_p) defines the maximum latency incurred when the number of active displays is fewer than $\mathcal{N}$. To illustrate the maximum latency, consider the following example. Assume a system that supports three simultaneous displays ($\mathcal{N} = 3$). Two displays are active (Y and Z) and a new request referencing object X arrives. This request arrives a little too late to consume the idle slot[c]. Thus, the display of X is delayed by one time period before it can be activated. Note that this maximum latency is applicable when the number of active displays is less than the total number of displays supported by the system ($\mathcal{N}$). Otherwise, the maximum latency should be computed based on appropriate queueing models.

Observe from Figure 2 that the disk incurs a T_{W_Seek} between the retrieval of each block. The disk performs wasteful work when it seeks (and useful work when it transfers data). T_{W_Seek} reduces the bandwidth of the disk drive. From the perspective of a video server, the effective bandwidth of the disk drive is a function of $\mathcal{B}$ and T_{W_Seek}. It is defined as:

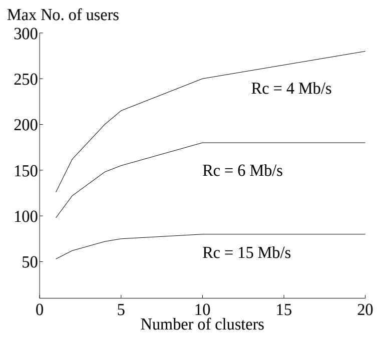
$$B_{Disk} = R_D \times \frac{\mathcal{B}}{\mathcal{B} + (T_{W\_Seek} \times R_D)} \tag{5}$$

The percentage of wasted disk bandwidth is quantified as:

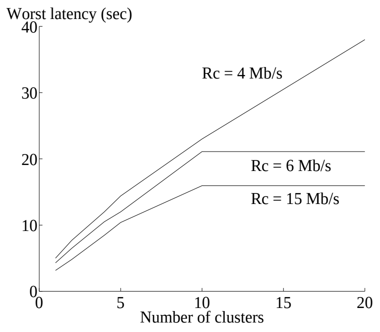
$$\frac{R_D - B_{Disk}}{R_D} \times 100 \tag{6}$$

Equations 2 to 6 establish the relationship between: 1) maximum throughput and latency time of a system, and 2) the available memory and disk bandwidth of a system. To illustrate, assume a system with a fixed amount of memory and a single disk drive. Given a desired throughput, one may compute the worst latency time using Equation 4. The theoretical upper bound on the throughput is determined by the transfer rate of the disk drive (R_D) and is defined as $\lfloor \frac{R_D}{R_C} \rfloor$ (the lower bound on this value is 0). Using Equation 3, the size of a block can be determined. The system can be configured with such a block size to support the desired throughput only if it is configured with sufficient amount of memory, i.e., the constraint imposed by Equation 2 is satisfied. Otherwise, the desired throughput should be reduced. This minimizes the amount of required memory, however, it results in a smaller block size that wastes a higher percentage of the disk bandwidth (Equations 5 and 6).

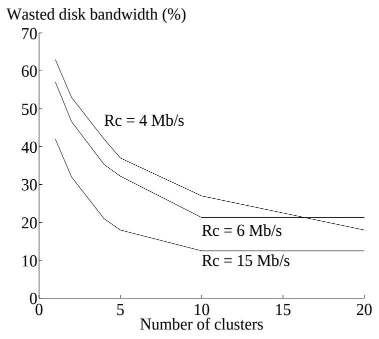
[c]Its display cannot start because it would interfere with the display of object Y, see Figure 3b.



a. Throughput



b. Maximum latency Time



c. Wasted disk bandwidth

Figure 4:

Disk Capacity	2.08 Gigabytes
Number of Cylinders $\#cyl$	$2,710$
Minimum Transfer Rate R_D	68.6 Mb/s
Maximum Seek Time	17 milliseconds
Maximum Rotational Latency Time	8.33 milliseconds

Table 2: *Seagate Barracuda 2-2HP* disk parameters

The next section extends these equations to a multi-disk hardware platform.

3 STRIPING

Simple striping[1] is a technique that distributes the load of a display uniformly across the available disk drives. Section 1 provides an overview of this technique. Given a system with a fixed amount of resources (disk and memory), the number of disks that constitute a cluster have a significant impact on the overall performance of a system. To illustrate, assume a system that consists of 20 disks (see Table 2 for disk specs) with 320 megabyte of memory. Figure 4 presents the maximum throughput and latency of the system as a function of the number of clusters for three different databases consisting of media types that require a bandwidth of 4, 6, and 15 Mb/s for their display. The throughput of the system increases as a function of the number of clusters because the percentage of wasted disk bandwidth decreases (see Figure 4). This in turn results in a higher latency. This behavior is best explained by considering the extreme choice of values for d, i.e., $d = 1$ and $d = D$. For each, we describe the scalability characteristics of the system as a function of additional resources (disk drive and memory) in a system. We assume a system consisting of one disk and 16 megabyte of memory as our base and focus on a media type with a bandwidth requirement of 4 Mb/s because the trends are identical for the other media types. The amount of memory (i.e., 16 megabyte) was chosen such that 20% of the disk bandwidth is wasted when the system is configured to maximize the number of displays (with R_C = 4 Mb/s). Next, we quantify the maximum throughput and latency time of the system with each technique when both the number of disk drives and the amount of memory increases by a factor of i, $1 \leq i \leq 20$. For example, when $i = 4$, the system consists of four disks and 64 megabyte of memory.

3.1 $d = 1$

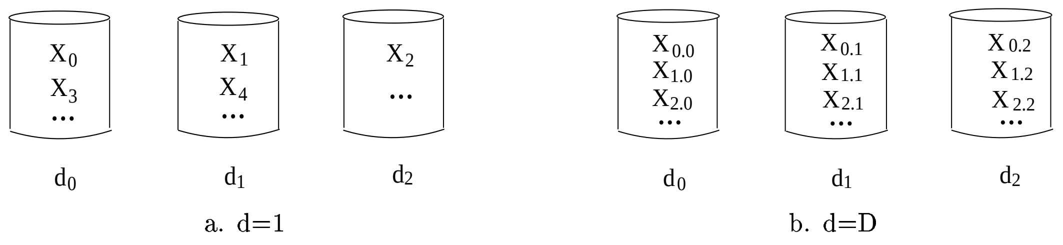


Figure 5: Two extreme stripe sizes (D=3)

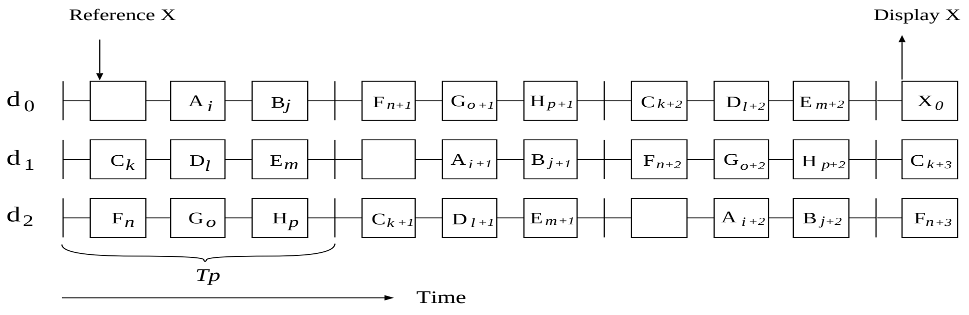


Figure 6: Maximum latency time with striping

When $d = 1$, the blocks of an object X are assigned to the D disk drives in a round-robin manner. The assignment of X_0 starts with an arbitrary disk. Assuming a system with three disk drives, Figure 5a demonstrates the assignment of blocks of X with this choice of value. The definition of a time period and its derivation are identical to Equation 4. The total number of simultaneous displays supported by the system is: $\mathcal{N} \times D$. Hence, throughput of the system (i.e., maximum number of displays) scales linearly as a function of additional resources in the system. However, its maximum latency also scales linearly (see Figure 7). To demonstrate this, assume that each disk in Figure 5a can support three simultaneous displays. Assume that eight displays are active and that the assignment of object X starts with d_0 (X_0 resides on d_0). If the request referencing object X arrives too late to utilize the idle slot of d_0, it must wait three (i.e., D) time periods before the idle slot on d_0 become available again (see Figure 6). Hence, the maximum latency time is $T_p \times D$.

The effective disk bandwidth of the system is a function of $\mathcal{B}$, T_{W_Seek}, and D. It is defined as:

$$B_{Disk} = D \times R_D \times \frac{\mathcal{B}}{\mathcal{B} + (T_{W\_Seek} \times R_D)} \tag{7}$$

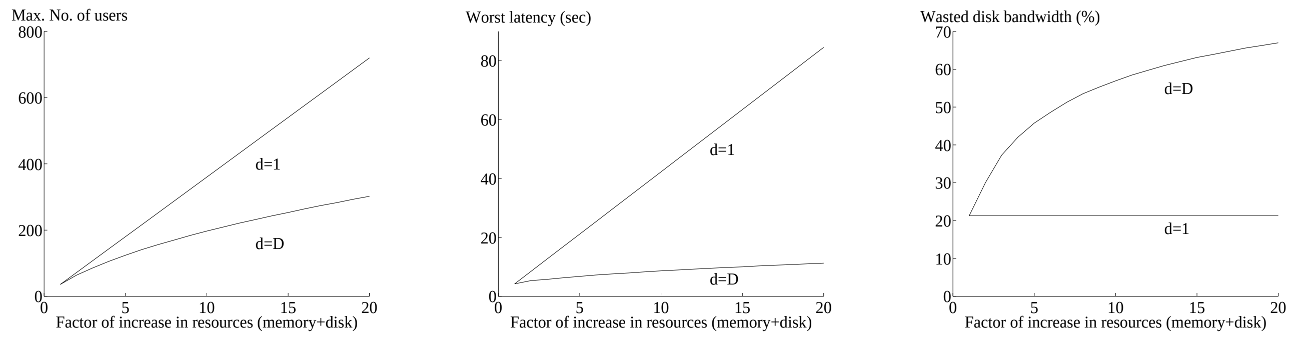
The percentage of wasted disk bandwidth is quantified as:

$$\frac{(D \times R_D) - B_{Disk}}{D \times R_D} \times 100 \tag{8}$$

(This expression is identical to that of Equation 6 because D can be factored out.) Thus, the wasted disk bandwidth is a constant as a function of additional resources, see Figure 7. This explains why the throughput of the system scales linearly.

3.2 $d = D$

When $d = D$, each display utilizes the total bandwidth of D disks. Logically, D disk drives are conceptualized as a single disk. Each block of an object is dispersed across the D disk drives. For example, in Figure 5b, block



a. Throughput b. Maximum latency Time c. Wasted disk bandwidth

Figure 7: $R_C = 4$ Mb/s

X_0 is declustered across the 3 disks. Each fragment of this block is denoted as $X_{0,i}$, $0 \leq i < D$. This approach enables the system to display an object using the design of Section 2. The only difference is that D disks are activated during each time slot (instead of one disk). The derivation of a time period, the maximum latency and required amount of memory are identical to those of Section 2. The effective bandwidth of the disks in the system is defined as:

$$B_{Disk} = D \times R_D \times \frac{\mathcal{B}}{\mathcal{B} + (T_{W\_Seek} \times D \times R_D)} \tag{9}$$

The amount of wasted disk bandwidth is computed using Equation 8. (This equation is not equivalent to Equation 6 because $D \times R_D$ appears in the denominator of Equation 9.)

As compared to $d = 1$, the system wastes the bandwidth of the disk drives as a function of additional resources, see Figure 7. This is because the amount of data transferred per disk drive decreases as a function of additional disks. The amount of memory needed to render the percentage of wasted disk bandwidth a constant far exceeds that of a linear growth. This causes a sub-linear increase in throughput, see Figure 7. By supporting fewer users, the duration of a time period remains relatively unchanged, enabling the system to provide a lower latency time. This demonstrates the tradeoff between throughput and latency time of a system. It is important to note that if enough memory was provided to enable the number of users to scale linearly as a function of additional disk drives, its maximum latency would have been identical to that of $d = 1$.

3.3 Configuration planner

Given a desired throughput and latency ($\mathcal{N}_{desired}$,$\ell_{desired}$), this section describes a configuration planner that determines a value for the configuration parameters of a systems: D, k, d, $\mathcal{B}$, Mem. The value of these parameters is chosen such that the total cost of the system is minimized. Its cost parameters include: \$/megabyte of memory, and[d] \$/disk. We assume that these costs scale linearly as a function of Mem and D. This is a simplistic assumption used to demonstrate the underlying methodology. Specialization of this methodology with complex cost functions is architecture dependent. It is a straightforward extension of our methodology and its pseudo-code.

The first part of the planner (see Figure 10) decides the number of disk drives needed in the environment, i.e., D. It iterates over possible values for D. For each value, its computes: 1) the theoretical upper bound of throughput ($\mathcal{N}_{max}$), and 2) the latency (termed ℓ_{min}) when $d = D$ for the desired throughput (using $\mathcal{N}_{desired}$ as input to Equation 4). The chosen value of D can provide neither a throughput higher than $\mathcal{N}_{max}$ nor a latency

[d]Note that, as compared to memory, we do not consider \$/megabyte of disk storage because the bottleneck resource is disk bandwidth. The disk bandwidth increases by introducing additional disk drives. Therefore, the cost per disk drive must be considered.

```
Max-throughput($D$, $R_C$)
    $\mathcal{N}_{Theory} = \lfloor \frac{D \times R_D}{R_C} \rfloor$
    $T_p = -\infty$
    **while** ($T_p < 0$ and $\mathcal{N}_{Theory} > 0$) **do**
        Compute $T_p$ using Equation 4 with $\mathcal{N}_{theory}$
        **if** ($T_p > 0$)
            **return** ($\mathcal{N}_{theory}$)
        $\mathcal{N}_{theory} = \mathcal{N}_{theory} - 1$
    **end** (* **while** *)
    **return** ($\mathcal{N}_{theory}$)
**end** (* **procedure** *)
```

Figure 8: Compute the maximum throughput of a system with D disks

```
configure-cluster($d$, $k$, $L$, $R_C$)
    $\mathcal{N}_{cluster} = Max - throughput(d, R_C)$
    $\mathcal{N}_{system} = \mathcal{N}_{cluster} \times k$
    **while** ($\mathcal{N}_{system} \geq \mathcal{N}_{desired}$) **do**
        Compute $T_p$ using Equation 4 with $\mathcal{N}_{cluster}$ and total transfer rate of $d$ disks
        $\ell_{config} = T_p \times k$
        Compute $\mathcal{B}$ using Equation 3
        **if** ($\ell_{config} \leq \ell_{desired}$) **then**
            Insert $< k, d, \mathcal{N}, \ell, \mathcal{B} >$ in list $L$
        $\mathcal{N}_{cluster} = \mathcal{N}_{cluster} - 1$
        $\mathcal{N}_{system} = \mathcal{N}_{cluster} \times k$
    **end** (* **while** *)
**end** (* **procedure** *)
```

Figure 9: Compute the feasibility of a cluster

```
Compute-config-params($R_C$, $\mathcal{N}_{desired}$, $\ell_{desired}$)
    Initialize list $L$ to be empty
    $D = 1$ ; $done = FALSE$
    $min\_cost = +\infty$
    **while** not $done$
        $\mathcal{N}_{max}$ = Max-throughput($D$,$R_C$)
        $\ell_{min}$ = invoke Equation 4 with $\mathcal{N}_{desired}$ and total bandwidth of $D$ disks
        **if** ($\ell_{min} \leq \ell_{desired}$ and $\mathcal{N}_{max} \geq \mathcal{N}_{desired}$) **then**
            **for** $k = 1$ to $D$ **do** configure-cluster($d = \frac{D}{k}$,$k$,$L$,$R_C$)
        $D = D + 1$
        compute the least expensive configuration found, store its cost in $min\_cost$
        **if** $min\_cost < D \times \frac{\$}{disk}$ **then** $done = TRUE$
    **end** (* **while** *)
    Compute the cost of each entry in list $L$
    Chose the least expensive set of configuration parameters
**end** (* **procedure** *)
```

Figure 10: Compute the configuration parameters of a system

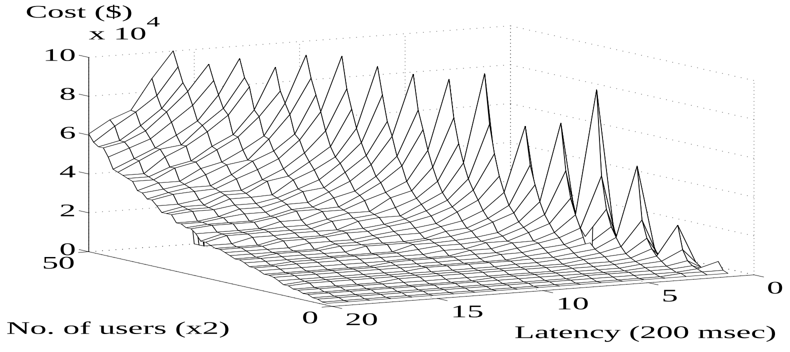


Figure 11: Optimization Search Space

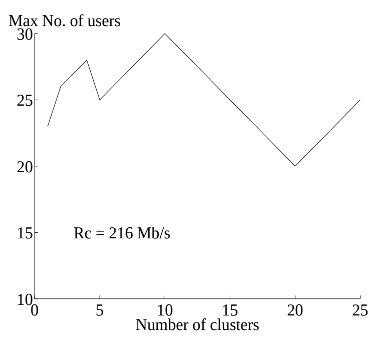
lower than ℓ_{min} for the desired number of users. If the chosen D value has the potential to realize the performance objective of the target application ($\mathcal{N}_{max} \geq \mathcal{N}_{desired}$ and $\ell_{min} \leq \ell_{desired}$) then the planner proceeds to search the optimization space (see Figure 11). It searches this space exhaustively analyzing all possible values for the number of clusters. This is because the optimization search space is irregular (see Figure 11). However, for a given $\mathcal{N}_{desired}$ and $\ell_{desired}$, the search space is small (few thousand states at most) and can be traversed in less than a second using the current CPU technology[e] (HP 9000/735 workstations in our case). For each cluster size, it computes the possible throughput and latency times. It maintains a set of values as a possible configuration only if it satisfies the performance objective of an application. The result is a list of possible configuration parameters for the system. In a final step, the planner computes the cost of each configuration and selects the least expensive one.

The termination condition of pseudo-code provided in Figure 10 is slightly complex. It maintains the least expensive configuration found thus far. Assume this configuration costs F dollars. As soon as the number of disks (D) exceeds $\frac{F}{\$/disk}$, the planner can terminate because larger D values cannot yield a configuration cheaper than the one found thus far.

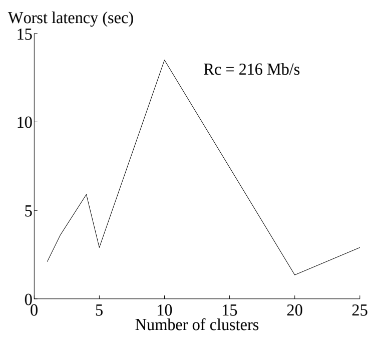
3.4 High bandwidth objects

There exist applications that cannot tolerate loss of data (e.g., medical data, scientific video signals such as those collected by NASA[4]). In these cases, the bandwidth required to display an object (R_C) may exceed the bandwidth of a single disk (R_D). This forces the system to utilize the aggregate bandwidth of at least $M = \lceil \frac{R_C}{R_D} \rceil$ disk drives to support a continuous display.[8,7,1] For example, with the disk specs of Table 2, if the bandwidth required to display an object is 216 Mb/s[10,3] then each cluster must consist of at least four disks ($d \geq 4$). Given a system with 100 disks and 1.6 gigabyte of memory, Figure 12 shows the maximum throughput and latency of the system as a function of the number of clusters (1, 2, 4, 5, 10, 20, and 25). As compared to the low bandwidth objects, the performance of the system is irregular as a function of the number of clusters because of the irregular nature of wasted disk bandwidth (see Figure 12). This is best clarified using an example. With 25 clusters, each cluster consist of four disks with aggregate bandwidth of 274.4 Mb/s and supports one display. This configuration supports a maximum of 25 displays. With 10 clusters, each cluster consists of ten disks with aggregate bandwidth of 686 Mb/s and supports three display. This configuration minimizes the percentage of wasted disk bandwidth attributed to fragmentation and supports 30 simultaneous displays.

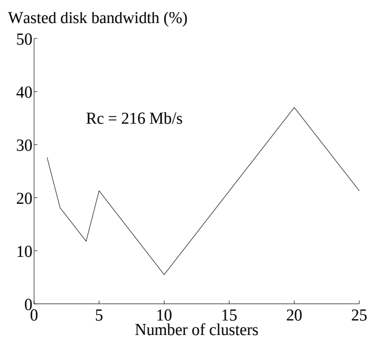
[e]The small size of search space discouraged us from developing heuristics to further shrink the search space. Moreover, this search is done only once at system configuration time to reduce cost. It might not be beneficial to risk the possibility of a high cost with a heuristic that fails to minimize cost.



a. Throughput



b. Maximum latency Time



c. Wasted disk bandwidth

Figure 12: R_C=216 Mb/s

The configuration planner supports high bandwidth objects because it ensures that $d \geq M$ in order to support a continuous display.

3.5 Discussion

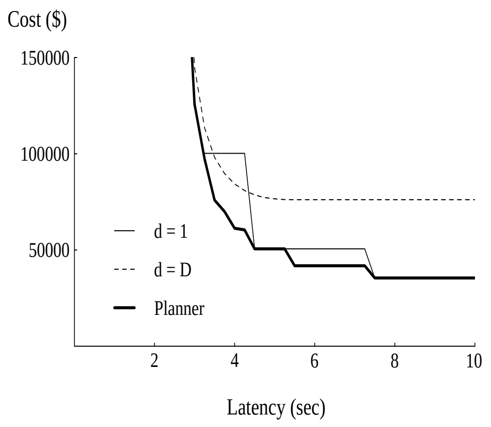
To simplify discussion, we have employed the simple technique of Section 2 to support hiccup-free displays. However, this display technique can be optimized using disk scheduling techniques (e.g., GSS[18]), data placement techniques (e.g., REBECA[6]), and a combination of these two techniques. These techniques maximize the throughput of a cluster by reducing the impact of seek times. In essence they perform local optimizations that are orthogonal to the global optimizations performed by the planner proposed in this study. Thus, the planner can be extended to incorporate these techniques.

4 A COMPARISON

We conducted the following three sets of experiments to demonstrate the flexibility and usefulness of the configuration planner. Each disk drive assumed in this study provides a transfer rate of 68.6 megabits per second and costs \$1000. We assumed that the cost per megabyte of memory is \$35. We consider two different media types: those that require either a 15 or a 216 Mb/s bandwidth requirement. We start by focusing on the media type that requires a 15 Mb/s bandwidth requirement.

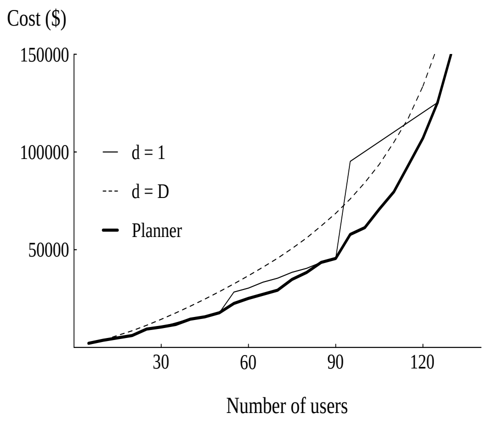
The first experiment fixed the desired number of users at 100 and varied the desired latency. It invoked the configuration planner for each $\ell_{desired}$ and $\mathcal{N}_{desired} = 100$. We also computed the cheapest system configured with either $d = 1$ or $d = D$. The obtained results are presented in Table 3. The last column of Table 3 presents the percentage savings[f] provided by the planner relative to both $d = 1$ and $d = D$. In general, the proposed planner provides a lower cost system as compared to both $d = 1$ and $d = D$. A slight increase in latency (from 3.0 to 3.5 seconds) can result in a significant drop in the required amount of resources (and cost). However, increasing the desired latency past a certain threshold provides no added savings. For example, increasing the latency beyond 7.5 seconds has no impact because a minimum of 34 disks with 42 megabyte of memory is required to support 100 displays. Note that planner provides significant savings as compared to $d = D$. For a few chosen values of

[f] For example, its computation for $d = D$ is: $100 \times \frac{cost(d=D) - cost(planner)}{cost(d=D)}$; note that this number may not exceed 100%.



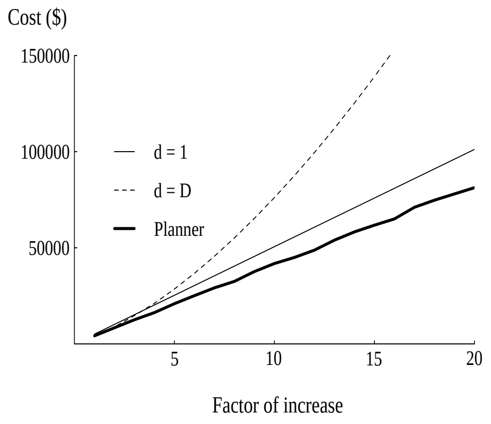
$\ell_{desired}$	Configuration planner					% Savings relative to	
	Mem (MB)	D	k	d	Cost ($)	$d = D$	$d = 1$
2.75	26	220	20	11	220911	6.08	N/A
3.00	23	125	25	5	125796	13.16	N/A
3.25	154	92	4	23	97375	14.20	2.83
3.50	26	75	25	3	75927	22.63	24.23
3.75	141	65	5	13	69940	22.02	30.21
4.00	37	60	20	3	61289	27.30	38.84
4.25	155	55	5	11	60441	25.29	39.69
4.50	17	50	50	1	50583	35.64	0.00
4.75	17	50	50	1	50583	34.51	0.00
5.00	17	50	50	1	50583	33.94	0.00
5.50	52	40	20	2	41809	45.07	17.35
6.00	52	40	20	2	41809	45.07	17.35
6.25	52	40	20	2	41809	45.07	17.35
6.50	52	40	20	2	41809	45.07	17.35
7.00	52	40	20	2	41809	45.07	17.35
7.25	52	40	20	2	41809	45.07	17.35
7.50	42	34	34	1	35457	53.42	0.00
7.75	42	34	34	1	35457	53.42	0.00
8.00	42	34	34	1	35457	53.42	0.00

Table 3: Fixed throughput of 100 simultaneous displays ($R_C = 15$ Mb/s)



$\mathcal{N}_{desired}$	Configuration planner					% Savings relative to	
	Mem (MB)	D	k	d	Cost ($)	$d = D$	$d = 1$
5	3	2	2	1	2099	0.38	0.00
10	19	3	1	3	3658	0.00	0.41
15	25	4	4	1	4880	20.42	0.00
25	11	9	9	1	9396	17.07	0.00
30	13	10	10	1	10439	27.44	0.00
35	50	10	5	2	11742	33.45	6.24
40	70	12	4	3	14454	31.47	1.05
50	21	17	17	1	17736	37.85	0.00
55	72	20	5	4	22507	30.82	20.55
65	34	26	13	2	27182	33.81	18.59
70	36	28	14	2	29272	35.83	17.33
75	52	33	11	3	34809	31.26	9.46
85	14	43	43	1	43503	30.01	0.00
95	52	56	14	4	57822	23.88	39.26
100	37	60	20	3	61289	27.30	38.84
110	75	77	11	7	79620	24.16	27.77
115	37	92	23	4	93297	20.56	19.04
120	60	105	15	7	107095	19.91	10.94
125	8	125	125	1	125263	18.67	0.00
130	196	145	5	29	151875	15.33	N/A
140	106	230	10	23	233696	11.81	N/A

Table 4: Fixed latency of 4.0 seconds ($R_C = 15$ Mb/s)



Factor of increase	Configuration planner					% Savings relative to	
	Mem (MB)	D	k	d	Cost ($)	$d = D$	$d = 1$
1	6	4	2	2	4197	4.48	17.12
2	11	8	4	2	8376	4.99	17.24
3	16	12	6	2	12555	14.81	17.29
4	36	15	5	3	16277	22.82	19.57
5	26	20	10	2	20913	26.72	17.32
6	31	24	12	2	25093	31.65	17.33
7	119	25	5	5	29159	36.06	17.65
8	72	30	10	3	32522	41.01	19.64
9	47	36	18	2	37630	42.38	17.34
10	52	40	20	2	41809	45.07	17.35
20	179	75	25	3	81259	62.34	19.67
30	271	114	38	3	123497	69.74	18.61
40	357	150	50	3	162486	74.98	19.69
50	449	189	63	3	204725	78.16	19.05
60	535	225	75	3	243714	80.82	19.69
70	627	264	88	3	285952	82.64	19.23
80	713	300	100	3	324941	84.28	19.69
90	805	339	113	3	367180	85.49	19.34
100	891	375	125	3	406169	86.62	19.69
200	1780	750	250	3	812307	92.11	19.70

Table 5: Base throughput of 10 displays with a latency time of 0.7 seconds ($R_C = 15$ Mb/s)

$\ell_{desired}$	Configuration planner					% Savings relative to	
	Mem (MB)	D	k	d	Cost ($)	$d = D$	$d = 4$
0.50	43	516	6	86	517496	1.42	N/A
0.75	21	144	18	8	144732	5.67	N/A
1.00	27	108	18	6	108934	9.80	N/A
1.25	34	90	18	5	91199	17.84	N/A
1.50	84	81	9	9	83957	21.98	N/A
1.75	139	78	6	13	82872	22.71	N/A
2.00	139	78	6	13	82872	22.71	N/A
2.25	60	72	18	4	74086	30.91	0.00
2.50	60	72	18	4	74086	30.91	0.00
3.00	60	72	18	4	74086	30.91	0.00
3.50	60	72	18	4	74086	30.91	0.00
4.00	60	72	18	4	74086	30.91	0.00
4.25	60	72	18	4	74086	30.91	0.00
4.50	253	63	9	7	71845	33.00	3.02
4.75	253	63	9	7	71845	33.00	3.02
5.00	253	63	9	7	71845	33.00	3.02

Table 6: Fixed throughput of 18 simultaneous displays ($R_C = 216$ Mb/s)

$\mathcal{N}_{desired}$	Configuration planner					% Savings relative to	
	Mem (MB)	D	k	d	Cost ($)	$d = D$	$d = 4$
2	9	8	2	4	8329	0.81	0.00
4	67	14	2	7	16328	8.85	1.34
6	93	21	3	7	24259	14.86	2.06
8	120	28	4	7	32190	19.33	2.42
10	146	35	5	7	40121	22.96	2.64
12	173	42	6	7	48052	25.98	2.78
14	200	49	7	7	55983	28.60	2.89
16	226	56	8	7	63914	30.92	2.96
18	60	72	18	4	74086	30.91	0.00
22	72	88	22	4	90525	34.56	0.00
26	85	104	26	4	106964	37.65	0.00
30	97	120	30	4	123403	40.33	0.00
34	110	136	34	4	139842	42.68	0.00
36	271	156	12	13	165487	37.14	N/A
38	173	171	19	9	177069	37.43	N/A
40	422	170	10	17	184787	39.05	N/A

Table 7: Fixed latency of 4 seconds ($R_C = 216$ Mb/s)

Factor of increase	Configuration planner					% Savings relative to	
	Mem (MB)	D	k	d	Cost ($)	$d = D$	$d = 4$
1	19	20	5	4	20659	10.52	0.00
2	146	35	5	7	40121	22.96	2.64
3	50	60	15	4	61757	27.68	0.00
4	279	70	10	7	79776	34.88	3.07
5	82	100	25	4	102854	36.92	0.00
6	412	105	15	7	119431	42.25	3.22
7	492	126	18	7	143225	43.51	0.51
8	545	140	20	7	159087	47.53	3.29
9	625	161	23	7	182880	48.57	1.17
10	678	175	25	7	198742	51.61	3.34
12	811	210	30	7	238397	54.90	3.36
14	944	245	35	7	278052	57.64	3.39
16	1077	280	40	7	317708	59.98	3.40
18	1210	315	45	7	357363	62.00	3.41
20	1343	350	50	7	397018	63.78	3.42
30	2008	525	75	7	595295	70.26	3.45
40	2673	700	100	7	793571	74.46	3.47

Table 8: Base throughput of 5 displays with a latency time of 2 seconds ($R_C = 216$ Mb/s)

$\ell_{desired}$ (3.5, 3.75, 4.0, 4.25, 6.5, and 7.0), it outperforms $d = 1$ by a significant margin. With high $\ell_{desired}$ values, planner simulates $d = 1$ to minimize cost. It does construct clusters to minimize the amount of required memory to provide marginal benefit (when compared to $d = 1$) with $\ell_{desired} \geq 7.5$.

In the second experiment, we fixed $\ell_{desired}$ at 4.0 seconds and varied the desired number of users. Additional resources are required to support additional users, increasing the system cost (see Table 4). Once again, the results indicate that planner can support a diverse throughput and latency requirement at a lower cost. When $d = 1$, the system could not support more than 130 simultaneous displays with a 4 second latency time. The system can support a maximum of 140 simultaneous displays with this latency.

In the final experiment, we assumed $\mathcal{N}_{desired} = 10$ with a desired latency of 0.7 seconds. Next, we increased both the latency and throughput by a fixed factor to see the impact on cost. The planner exhibits an increase marginally below linear. It provides significant savings relative to both $d = D$ and $d = 1$. With 2000 users and a latency of 140 seconds (last row of Table 5), the proposed planner provides a 92% reduction in cost as compared to $d = D$.

Tables 6, 7, and 8 report the results of these three experiments with high bandwidth objects (R_C=216 Mb/s). This media type requires the participation of at least four disk drives. Note that with $d = 4$, the system cannot support certain performance requirements, e.g., in Table 6, $d = 4$ cannot support 18 displays with a maximum latency lower than 2.25 seconds. However, when the performance objectives are not excessive, $d = 4$ is a competitive configuration, e.g., latencies higher than 2.25 in Table 6 and throughputs lower than 35 in Table 7. In Table 8, we chose a latency and throughput that was not excessive for $d = 4$. Note that the planner trades memory for disk bandwidth to minimize the cost of supporting a desired performance. For example, the amount of memory required to support 25 displays with a ten second latency (the fifth row of Table 8) is lower than that with 20 displays and an eight second (fourth row of the same table) latency. In general, the results demonstrate significant savings with the planner as compared to $d = D$.

5 CONCLUSION AND FUTURE DIRECTIONS

This study quantifies the tradeoffs associated with choosing the stripe size in a multi-disk hardware platform. In addition, it describes a planner that decides a value for configuration parameters of a system to supports the performance criteria of a target application. In an industrial setting, the proposed configuration planner can be extended to consider: 1) choice of different disk drives with various storage capacities, bandwidth, and cost, 2) disk controller cost to manage the disks, 3) system bus architecture to support the desired number of disks, etc. The networking aspects, size of the database and features offered to clients contribute to the dimensions of the optimization space. These factors would complicate the design of a planner. Indeed, the search space of the optimization space might become large enough to motivate the use of heuristics. These extensions are vendor specific specialization of our proposed methodology.

We plan to investigate an implementation of striping using a cluster of workstations. (This cluster simulates a shared-nothing architecture.[16]) Presently, each workstation consists of a fast CPU, 80 megabyte of memory, and eight 1 gigabyte disk drives. Each workstation can be configured with as many as 17 disk drives and upto 400 megabyte of memory. This environment provides many interesting systems issues. Here are a few. How is the configuration planner extended to consider the physical limitations of a workstation and the cost of introducing additional workstations? Assuming an environment consisting of many workstations, if a cluster is to consist of twenty disk drives, should a cluster span multiple workstations? If so, how can the disks be managed effectively to support a continuous display? We intend to investigate these and other issues (e.g., the role of tertiary storage devices[9,5]) as a part of our implementation activity.

6 REFERENCES

[1] S. Berson, S. Ghandeharizadeh, R. Muntz, and X. Ju. Staggered Striping in Multimedia Information Systems. In *Proceedings of ACM SIGMOD*, pages 79–89, 1994.

[2] P. M. Chen and D. Paterson. A new approach to I/O performance evaluation - self-scaling I/O benchmarks, predicted I/O performance. In *Proceedings of the 1993 ACM SIGMETRICS Int'l Conf. on Measurement and Modeling of Computer Systems*, May 1993.

[3] Personal Communication. Dr. Tom Holman, inventor of THX, USC School of Cinema.

[4] J. Dozier. Access to data in NASA's Earth observing system (Keynote Address). In *Proceedings of ACM SIGMOD*, pages 1–1, June 1992.

[5] S. Ghandeharizadeh, A. Dashti, and C. Shahabi. A Pipelining Mechanism to Minimize the Latency Time in Hierarchical Multimedia Storage Managers. *Computer Communications*, March 1995.

[6] S. Ghandeharizadeh, S. H. Kim, and C. Shahabi. On Configuring a Single Disk Continuous Media Server. In *Proceedings of the ACM SIGMETRICS/PERFORMANCE*, May 1995.

[7] S. Ghandeharizadeh and L. Ramos. Continuous Retrieval of Multimedia Data Using Parallelism. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):658–669, August 1993.

[8] S. Ghandeharizadeh, L. Ramos, Z. Asad, and W. Qureshi. Object Placement in Parallel Hypermedia Systems. In *Proceedings of Very Large Databases*, pages 243–254, September 1991.

[9] S. Ghandeharizadeh and C. Shahabi. On Multimedia Repositories, Personal Computers, and Hierarchical Storage Systems. In *Proceedings of ACM Multimedia Conference*, 1994.

[10] Jukka Hamlainen. Video Recording Goes Digital. *IEEE Spectrum*, pages 76–79, April 1995.

[11] R. Ng and J. Yang. Maximizing Buffer and Disk Utilization for News On-Demand. In *Proceedings of Very Large Databases*, 1994.

[12] V.G. Polimenis. The Design of a File System that Supports Multimedia. Technical Report TR-91-020, ICSI, 1991.

[13] P. Rangan and H. Vin. Efficient Storage Techniques for Digital Continuous Media. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):564–573, August 1993.

[14] A. L. N. Reddy and J. C. Wyllie. I/O Issues in a Multimedia System. *IEEE Computer Magazine*, 27(3):69–74, March 1994.

[15] Seagate. Barracuda family. *The Data Technology Company Product Overview*, page 25, March 1994.

[16] M. R. Stonebraker. The case for Shared-Nothing. In *Proceedings of the 1986 Data Engineering Conference*. IEEE, 1986.

[17] 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*, pages 393–400, August 1993.

[18] 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.