A Comparison of Alternative Continuous Display Techniques with Heterogeneous Multi-Zone Disks *

Shahram Ghandeharizadeh and Seon Ho Kim

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

Abstract

A number of recent technological trends have made data intensive applications such as continuous media (audio and video) servers a reality. These servers are expected to play an important role in applications such as video-on-demand, digital library, news-on-demand, distance learning, etc. Continuous media applications are data intensive and might require storage subsystems that consist of hundreds of (multi-zone) disk drives. With the current technological trends, a homogeneous disk subsystem might evolve to consist of a heterogeneous collection of disk drives. Given such a storage subsystem, the system must continue to support a hiccup-free display of audio and video clips. This study describes extensions of four continuous display techniques for multi-zone disk drives to a heterogeneous platform. These techniques include IBM's Logical Track [21], HP's Track Pairing [4], and USC's FIXB [9] and dead-line driven techniques [10]. We quantify the performance tradeoff associated with these techniques using analytical models and simulation studies. The obtained results demonstrate tradeoffs between the cost per simultaneous stream supported by a technique, the wasted disk space, and the incurred startup latency.

1 Introduction

Continuous media objects, audio and video clips, are large in size and must be retrieved at a pre-specified rate in order to ensure their hiccup-free display [24, 8]. Even with the introduction of 50 gigabyte disk drives, a video library consisting of 1000 MPEG-2 titles (with an average display time of 90 minutes) requires sixty such disks for data storage[1]. Over time such a storage system will evolve to consist of a heterogeneous collection of disk drives. This is because the system administrator is forced to buy new disk drives over time and the original disk models will either be scarce or simply unavailable at the time of purchase. Consider each observation in turn. There are several reasons

*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]Assuming an average bandwidth requirement of 4 Mbps for each clip, the system designer might utilize additional disk drives to satisfy the bandwidth requirement of this library, i.e., number of simultaneous users accessing the library.

why a system administrator might be forced to buy new disk drives over time. First, the application might require either a larger storage capacity due to introduction of new titles or a higher bandwidth due to a larger number of users accessing the library. Second, existing disks might fail and need to be replaced[2]. The system administrator may not be able to purchase the original disk models due to the technological trends in the area of magnetic disks: Approximately every 12 to 18 months, the cost per megabyte of disk storage drops by 50%, its storage space doubles in size, and its average transfer rate increases by 40% [20, 14]. Older disk models are discontinued because they cannot compete in the market place. For example, a single manufacturer introduced three disk models in the span of six years, a new model every two years, see Table 1. The oldest model (introduced in 1994) costs more than the other two while providing both a lower storage capacity and bandwidth.

With a heterogeneous disk subsystems, a continuous media server must continue to deliver the data to a client at the bandwidth pre-specified by the clip. For example, if a user references a movie that requires 4 megabits per second (Mbps) for its continuous display, then, once the system initiates its display, it must be rendered at 4 Mbps for the duration of its presentation[3]. Otherwise, a display may suffer from frequent disruptions and delays, termed hiccups. Numerous studies [16, 19, 2, 23, 3, 11, 12] have described techniques in support of a hiccup-free display assuming a homogeneous collection of single-zone disk drives. Single-zone disk drives provide a constant transfer rate. A multi-zone disk drive consists of a number of regions (termed zones) that provide a different storage capacity and transfer rate. For example, the Seagate Barracuda 18 provides 18 gigabyte of storage and consists of 9 zones (see Table 1). To the best of our knowledge, there are only four techniques in support of hiccup-free display with multi-zone disk drives: IBM's Logical Track [21], Hewlett Packard's Track Pairing [4] and USC's FIXB [9] and dead-line driven [10] techniques. Studies that investigate stochastic analytical models in support of admission control with multi-zone disks, e.g., [18], are orthogonal because they investigate only admission control (while the above four techniques describe how the disk bandwidth should be scheduled, the block size for each object, and admission control). Moreover, we are not aware of a single study that investigates hiccup-free display using a heterogeneous collection of multi-zone disk drives.

This study extends the four techniques to a heterogeneous disk subsystem. While these extensions are novel and a contribution in their own right, we believe that the primary contribution of this study is the performance comparison of these techniques and quantification of their tradeoff. This is because three of the described techniques assume certain characteristics about the target platform.

[2]Disks are so cheap and common place that it is more economical to replace the failed ones instead of fixing them.

[3]This study assumes constant bit rate media types. Extensions of this work in support of variable bit rate can be accomplished by extending our proposed techniques with those surveyed in [1].

	Seagate Barracuda 4LP Introduced in 1994, 2 GBytes capacity, with a $1,200 price tag				Seagate Cheetah 4LP Introduced in 1996, 4 GBytes capacity, with a $1,100 price tag				Seagate Barracuda 18 Introduced in 1998, 18 GBytes capacity, with a $900 price tag			
Zone id	Size (MB)	Track Size (MB)	No. of Tracks	Rate (MB/s)	Size (MB)	Track Size (MB)	No. of Tracks	Rate (MB/s)	Size (MB)	Track Size (MB)	No. of Tracks	Rate (MB/s)
0	506.7	0.0908	5579	10.90	1017.8	0.0876	11617	14.65	5762	0.1268	45429	15.22
1	518.3	0.0903	5737	10.84	801.6	0.0840	9540	14.05	1743	0.1214	14355	14.57
2	164.1	0.0864	1898	10.37	745.9	0.0791	9429	13.23	1658	0.1157	14334	13.88
3	134.5	0.0830	1620	9.96	552.6	0.0745	7410	12.47	1598	0.1108	14418	13.30
4	116.4	0.0796	1461	9.55	490.5	0.0697	7040	11.65	1489	0.1042	14294	12.50
5	121.1	0.0767	1579	9.20	411.4	0.0651	6317	10.89	1421	0.0990	14353	11.88
6	119.8	0.0723	1657	8.67	319.6	0.0589	5431	9.84	1300	0.0923	14092	11.07
7	103.2	0.0688	1498	8.26					1268	0.0867	14630	10.40
8	101.3	0.0659	1536	7.91					1126	0.0807	13958	9.68
9	92.0	0.0615	1495	7.38								
10	84.6	0.0581	1455	6.97								

Table 1: Three different Seagate disk models and their zone characteristics

Our performance results enable a system designer to evaluate the appropriateness of a technique in order to decide whether it is worthwhile to refine its design by eliminating its assumptions.

The rest of this paper is organized as follows. Section 2 introduces five hiccup-free display techniques for a heterogeneous disk subsystem: two for IBM's Logical Track (termed OLT1 and OLT2), and one for each of the other techniques. Section 3 quantifies the performance tradeoff associated with these techniques. Our results demonstrate tradeoffs between cost per simultaneous stream supported by a technique, its startup latency, throughput, and the amount of disk space that it wastes. For example, while USC's FIXB results in the best cost/performance ratio, the potential maximum latency incurred by each user is significantly larger than the other techniques. The choice of a technique is application dependent: One must analyze the requirements of an application and choose a technique accordingly. For example, with nonlinear editing systems, the deadline driven (DD) technique is more desirable than the others because it minimizes the latency incurred by each users [15]. Our conclusions and future research directions are contained in Section 4.

2 Five Techniques

In order to describe the alternative techniques, we assume a configuration consisting of K disk models: D_0, D_1, ..., D_{K-1}. There are q_i disks belonging to disk model i: d^i_0, d^i_1, ..., $d^i_{q_i-1}$. A disk drive of model D_i consists of m_i zones. To illustrate, Figure 1 shows a configuration consisting of two disk models D_0 and D_1 (K=2). There are two disks of each model (q_0=q_1=2), numbered d^0_0, d^0_1, d^1_0, d^1_1. Disks of model 0 consist of 2 zones (m_0=2) while those of model 1 consist of 3 zones (m_1=3). Zone j of a disk (say d^0_0) is denoted as $Z_j(d^0_0)$. Figure 1 shows a total of 10 zones for the 4 disk drives and their unique indexes. The k^{th} physical track of a specific zone is indexed as $PT_k(Z_j(d^i_0))$.

Term	Definition
K	number of disk models
D_i	disk model i, $0 \leq i < K$
q_i	number of disks for disk model D_i
d^i_j	jth disk drive of disk model D_i, $0 \leq j < q_i$
m_i	number of zones for each disk of disk model D_i
$Z_i(d^l_k)$	zone i of disk d^l_k, $0 \leq i < m_l$
$\#TR_i$	number of tracks in disk model i
$NT(Z_i())$	number of tracks in zone i
$PT_i(Z_j())$	track i of zone j, $0 \leq i < NT(Z_j())$
LT_i	logical track i
$AvgR_i$	average transfer rate of disk model i
B_i	block size for disk model i
T_{W_Seek}	Worst seek time of a zone (including the maximum rotational latency time)
T_{cseek}	Seek time required to make a complete span
$R(Z_i)$	Transfer rate of Z_i
$S(Z_i)$	Storage capacity of Z_i
T_{scan}	Time to perform one sweep of m zones
$T_{MUX}(Z_i)$	Time to read $\mathcal{N}$ blocks from zone Z_i
R_C	Display bandwidth requirement (Consumption rate)
$\mathcal{N}$	Maximum number of simultaneous displays (throughput)
ℓ	Maximum latency time

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

We use the set notation, $\{\ :\ \}$, to refer to a collection of tracks from different zones of several disk drives. This notation specifies a variable before the colon and, the properties that each instance of the variable must satisfy after the colon. For example, to refer to the first track from all zones of the disk drives that belong to disk model 0, we write: $\{PT_0(Z_j(d^0_i))\ :\ \forall i, j\ where\ 0 \leq j < m_0\ and\ 0 \leq i < q_0\}$. With the configuration of Figure 1, this would expand to:
$\{PT_0(Z_0(d^0_0)), PT_0(Z_0(d^0_1)), PT_0(Z_1(d^0_0)), PT_0(Z_1(d^0_1))\}$.

2.1 IBM's Logical Track [21]

This section starts with a description of this technique for a single multi-zone disk drive. Subsequently, we introduce two variations of this technique, OLT1 and OLT2, for a heterogeneous collection of disk drives. While OLT1 constructs several logical disks from the physical disks, OLT2 provides the abstraction of only one logical disk. We describe each in turn.

With a single multi-zone disk drive, this technique constructs a logical track from each distinct zone provided by the disk drive. Conceptually, this approach provides equi-sized logical tracks with a single data transfer rate such that one can apply traditional continuous display techniques [2, 23, 3, 11, 17, 19]. With K different disk models, a naive approach would construct a logical track LT_k

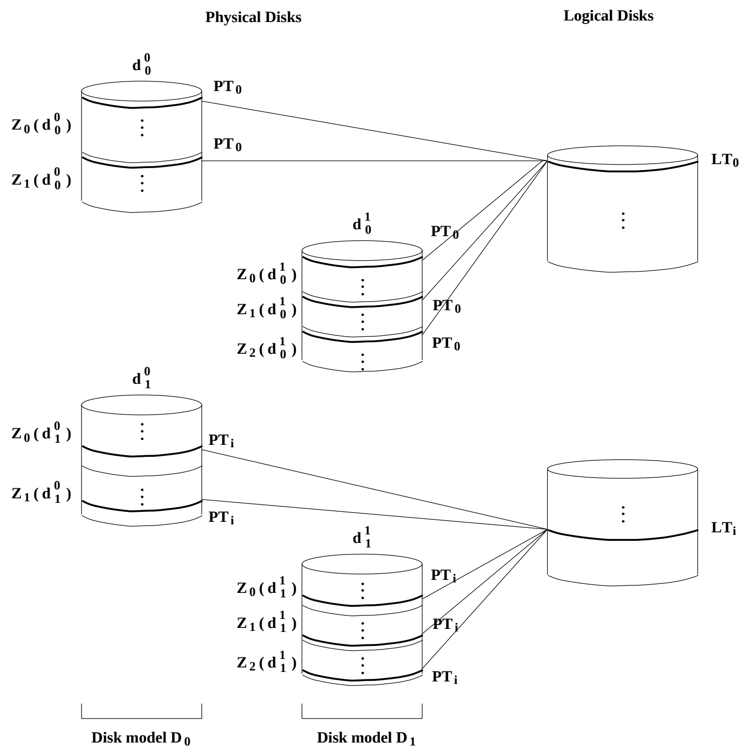


Figure 1: OLT1

by utilizing one track from each zone: $LT_k = \{ PT_k(Z_j(d_p^i)) : \forall i,j,p \; where \; 0 \leq j < m_i \; and \; 0 \leq i < K \; and \; 0 \leq p < q_i \}$. With this technique, the value of k is bounded by the zone with the fewest physical tracks, i.e., $0 \leq k < Min[NT(Z_j(d_{q_i}^i))]$, where $NT(Z_j(d_{q_i}^i))$ is the number of physical tracks in zone j of disk model D_i. Large logical tracks result in a significant amount of memory per simultaneous display, rendering a continuous media server economically unavailable. In the next section, we describe two optimized versions of this technique that render its memory requirements reasonable.

2.2 Optimized Logical Track 1 (OLT1)

Assuming that a configuration consists of the same number of disks for each model[4], OLT1 constructs logical disks by grouping one disk from each disk model (q logical disks). For each logical disk, it constructs a logical track consisting of one track from each physical zone of a disk drive. To illustrate, in Figure 1, we pair one disk from each model to form a logical disk drive. The two disks that constitute the first logical disk in Figure 1, i.e., disks d_0^0 and d_0^1, consist of a different number of

[4]This technique is applicable as long as the number of disks for each model is a multiple of the model with the fewest disk drives: if $min(q_i)$, $0 \leq i < K$, denotes the model with fewest disks, then q_j is a multiple of $min(q_i)$.

zones (d_0^0 has 2 zones while d_0^1 has 3 zones). Thus, a logical track consists of 5 physical tracks, one from each zone.

Logical disks appear as a homogeneous collection of disk drives with the same bandwidth. There are a number of well known techniques that can guarantee hiccup-free display given such an abstraction, see [2, 23, 3, 11, 17, 19, 22]. Briefly, given a video clip X, these techniques partition X into equi-sized blocks that are striped across the available logical disks [3, 22, 23]: one block per logical disk in a round-robin manner. A block consists of either one or several logical tracks.

Let T_i denote the time to retrieve m_i tracks from a single disk of model D_i consisting of m_i zones: $T_i = m_i \times (a\ revolution\ time\ +\ seek\ time)$. Then, the transfer rate of a logical track (R_{LT}) is: $R_{LT} = \frac{size\ of\ a\ logical\ track}{Max[T_i]}\ \forall i,\ 0 \leq i < K$.

In Figure 1, to retrieve a LT from the first logical disk, d_0^0 incurs 2 revolution times and 2 seeks to retrieve two physical tracks, while disk d_0^1 incurs 3 revolutions and 3 seeks to retrieve three physical tracks. Assuming a revolution time of 8.33 milliseconds (7200 rpm) and the average seek time of 10 milliseconds for both disk models, d_0^0 requires 36.66 milliseconds ($T_0 = 36.66$) while d_0^1 requires 54.99 ($T_1 = 54.99$) milliseconds to retrieve a LT. Thus, the transfer rate of the LT is determined by disk model D_1. Assuming that a LT is 1 megabyte in size, its transfer rate is $\frac{size\ of\ a\ logical\ track}{Max[T_0, T_1]} = \frac{1\ megabyte}{54.99\ milliseconds} = 18.19$ megabytes per second.

This example demonstrates that OLT1 wastes disk bandwidth by requiring one disk to wait for another to complete its physical track retrievals. In our example, this technique wastes 33.3% of D_0's bandwidth. In addition, this technique wastes disk space because the zone with the fewest physical tracks determines the total number of logical tracks. In particular, this technique eliminates the physical tracks of those zones that have more than NT_{min} tracks, $NT_{min} = Min[NT(Z_j(d_{q_i}^i))]$, i.e., it eliminates $PT_k(Z_j(d_{q_i}^i))$ with $NT_{min} \leq k < NT(Z_j(d_{q_i}^i))$, for all i and j, $0 \leq i < K$ and $0 \leq j < m_i$.

2.3 Optimized Logical Track 2 (OLT2)

OLT2 extends OLT1 with the following additional assumption: each disk model has the same number of zones, i.e., m_i is identical for all disk models, $0 \leq i < K$. Using this assumption, it constructs logical tracks by pairing physical tracks from zones that belong to different disk drives. This is advantageous for two reasons. First, it eliminates the seeks required per disk drive to retrieve the physical tracks. Second, assuming an identical revolution rate of all heterogeneous disks, it prevents

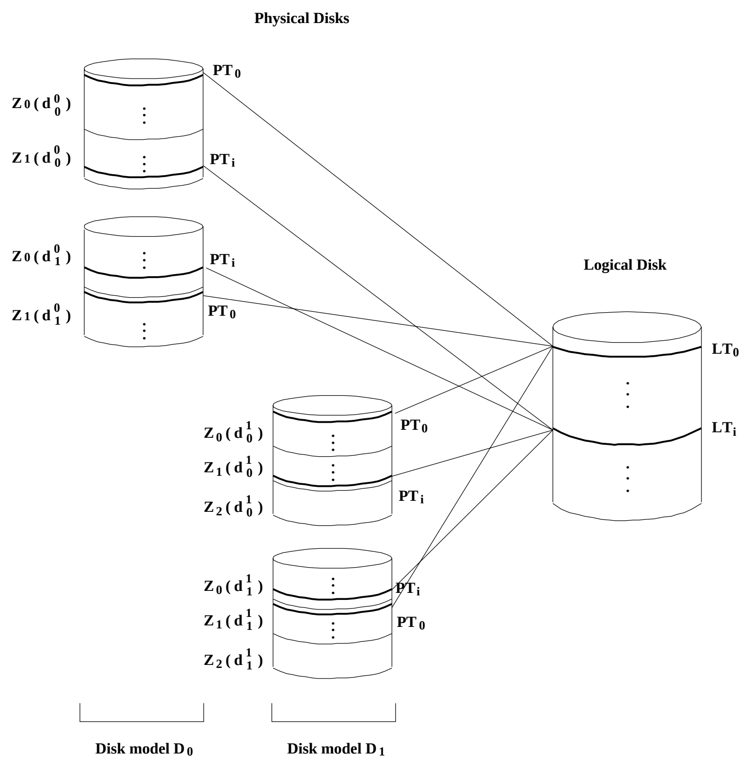


Figure 2: OLT2

one disk drive to wait for another.

The details of OLT2 are as follows. First, it reduces the number of zones of each disk to that of the disk with fewest zones: $m_{min} = Min[m_i]$ for all i, $0 \leq i < K$. Hence, we are considering only zones, $Z_j(d_k^i)$ for all i, j, and k ($0 \leq i < K$, $0 \leq j < m_{min}$, and $0 \leq k < q$). For example, in Figure 2, the slowest zone of disks of d_0^1 and d_1^1 (Z_2) are eliminated such that all disks utilize only two zones. This technique requires m_{min} disks of each disk model (totally $m_{min} \times K$ disks). Next, it constructs LTs such that no two physical tracks (from two different zones) in a LT belong to one physical disk drive. A logical track LT_k consists of a set of physical tracks:

$$LT_k = \{PT_{k \ mod \ NT_{min}}(Z_{(\lfloor \frac{k}{NT_{min}} \rfloor + j) \ mod \ m_{min}}(d^i_{j \ mod \ m_{min}})) \ : \forall i, j \ where \ 0 \leq i < K \ and \ 0 \leq j < m_{min}\}$$

The total number of LTs is $m_{min} \times NT_{min}$, thus $0 \leq k < m_{min} \times NT_{min}$.

OLT2 may use several possible techniques to force all disks to have the same number of zones. For each disk with δ_z zones more than m_{min}, it can either (a) merge two of its physically adjacent zones into one, repeatedly, until its number of logical zones is reduced to m_{min}, (b) eliminate its innermost δ_z zones, or (c) a combination of (a) and (b). With (a), the number of simultaneous displays is reduced because the bandwidth of two merged zones is reduced to the bandwidth of the slower participating zone. With (b), OLT2 wastes disk space while increasing the average transfer rate of the disk drive, i.e., number of simultaneous displays. In [9], we describe a configuration planner that empowers a system administrator to strike a compromise between these two factors for one of the techniques described in this study (HetFIXB). The extensions of this planner in support of OLT2 is a part of our future research direction.

2.4 Heterogeneous Track Pairing (HTP)

We describe this technique in two steps. First, we describe how it works for a single multi-zone disk drive. Next, we extend the discussion to a heterogeneous collection of disk drive. Finally, we discuss the tradeoff associated with this technique.

Assuming a single disk drive (say d_0^0) with $\#TR_0$ tracks, Track Pairing [4] pairs the innermost track ($TR_{\#TR_0-1}(d_0^0)$) with the outermost track ($TR_0(d_0^0)$), working itself towards the center of the disk drive. The result is a logical disk drive that consists of $\frac{\#TR_0}{2}$ logical tracks that have approximately the same storage capacity and transfer rate. This is based on the (realistic) assumption that the storage capacity of tracks increases linearly as one moves from the innermost track to the outermost track. Using this logical disk drive, the system may utilize one of the traditional continuous display techniques in support of hiccup-free display.

Assuming a heterogeneous configuration consisting of K disk models, HTP utilizes Track Pairing to construct track pairs for each disk. If the number of disks for each disk model is identical ($q_0 = q_1 = ... = q_{K-1}$), HTP constructs q_i groups of disk drives consisting of one disk from each of the K disk models. Next, it realize a logical track that consists of K track pairs, one track pair from each disk drive in the group. These logical tracks constitute a logical disk. Obviously, the disk with the fewest number of tracks determines the total number of logical tracks for each logical disk. With such a collection of homogeneous logical disks, one can use one of the popular hiccup-free display techniques. For example, similar to both OLT1 and OLT2, one can stripe a video clip into blocks and assign the blocks to the logical disks in a round-robin manner.

HTP wastes disk space in two ways. First, the number of tracks in a logical disk is determined by the physical disk drive with fewest track pairs. For example, if a configuration consists of two heterogeneous disks, one with 20,000 track pairs and the other with 15,000 track pairs, then the resulting logical disk will consist of 15,000 track pairs. In essence, this technique eliminates 5,000 track pairs from the first disk drive. Second, while it is realistic to assume that the storage capacity of each

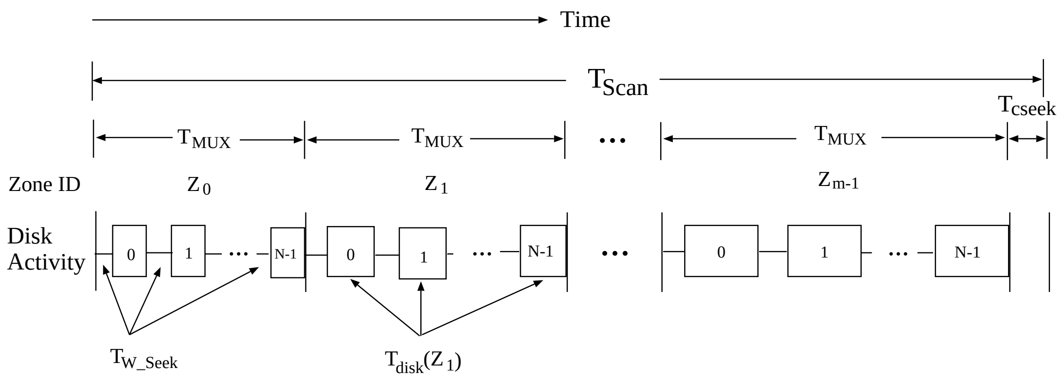


Figure 3: T_{Scan} and its relationship to T_{MUX}

track increases linearly from the innermost track to the outermost one, it is not 100% accurate [4]. Once the logical tracks are realized, the storage capacity of each logical track is determined by the track with the lowest storage capacity.

2.5 Heterogeneous FIXB

In order to describe this technique, we first describe how the system guarantees continuous display with a single multi-zone disk drive. Next, we describe the extensions of this technique to a heterogeneous disk drive.

2.5.1 FIXB with one Multi-zone Disk [9]

With this technique, the blocks of an object X are rendered equi-sized. Let B denote the size of a block. The system assigns the blocks of X to the zones in a round-robin manner starting with an arbitrary zone. FIXB configures the system to support a fixed number of simultaneous displays, $\mathcal{N}$. This is achieved by requiring the system to scan the disk in one direction, say starting with the outermost zone moving inward, visiting one zone at a time and multiplexing the bandwidth of that zone among $\mathcal{N}$ block reads. Once the disk arm reads $\mathcal{N}$ blocks from the innermost zone, it is repositioned to the outermost zone to start another sweep of the zones. The time to perform one such a sweep is denoted as T_{Scan}. The system is configured to produce and display an identical amount of data per T_{Scan} period. The time required to read $\mathcal{N}$ blocks from zone i, denoted $T_{MUX}(Z_i)$, is dependent on the transfer rate of zone i. This is because the time to read a block ($T_{disk}(Z_i)$) during one $T_{MUX}(Z_i)$ is a function of the transfer rate of a zone.

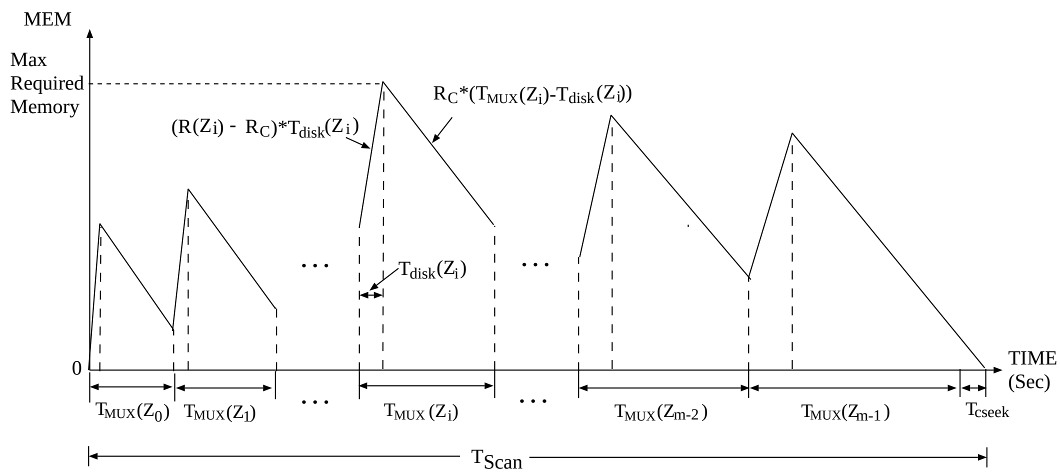


Figure 4: Memory required on behalf of a display

Figure 3 shows T_{Scan} and its relationship with $T_{MUX}(Z_i)$ for m zones. During each T_{MUX} period, $\mathcal{N}$ active displays might reference different objects. This would force the disk to incur a seek when switching from the reading of one block to another, termed T_{W_Seek}. T_{W_Seek} also includes the rotational latency time. At the end of a T_{Scan} period, the system observes a long seek time (T_{cseek}) attributed to the disk repositioning its arm to the outermost zone. The disk produces m blocks of X during one T_{Scan} period ($m \times B$ bytes). The number of bytes required to guarantee a hiccup-free display of X during T_{Scan} should either be lower than or equal to the number of bytes produced by the disk. This constraint is formally stated as:

$$\mathcal{R}_C \times (T_{cseek} + \sum_{i=0}^{m-1} T_{MUX}(Z_i)) \leq m \times B \tag{1}$$

The amount of memory required to support a display is minimized when the left hand side of Equation 1 equals its right hand side.

During a T_{MUX}, $\mathcal{N}$ blocks are retrieved from a single zone, Z_{Active}. In the next T_{MUX} period, the system references the next zone $Z_{(Active+1)\ mod\ m}$. When a display references object X, the system computes the zone containing X_0, say Z_i. The transfer of data on behalf of X does not start until the active zone reaches Z_i. One block of X is transfered into memory per T_{MUX}. Thus, the retrieval of X requires f such periods. (The display of X may exceed $\sum_{j=0}^{f-1} T_{MUX}(Z_{(i+j)\ mod\ m})$ seconds as described below.) The memory requirement for displaying object X varies due to the variable transfer rate. This is best illustrated using an example. Assume that the blocks of X are assigned to the zones starting with the outermost zone, Z_0. If Z_{Active} is Z_0 then this request employs one of the idle $T_{disk}(Z_0)$ slots to read X_0. Moreover, its display can start immediately because the outermost zone has the highest transfer rate. The block size and $\mathcal{N}$ are chosen such that the data accumulates in memory when accessing outermost zones and decreases when reading data from innermost zones on behalf of a display (see Figure 4). In essence, the system uses buffers to compensate for the low transfer rates of innermost zones using the high transfer rates of outermost zones, harnessing the average transfer rate of the disk. Note that the amount of required memory reduces to zero at the end of one T_{scan} in preparation for another sweep of the zones.

The display of an object may not start upon the retrieval of its block from the disk drive. This is because the assignment of the first block of an object may start with an arbitrary zone while the transfer and display of data is synchronized relative to the outermost zone, Z_0. In particular, if the assignment of X_0 starts with a zone other than the outermost zone (say Z_i, $i \neq 0$) then its display might be delayed to avoid hiccups. The duration of this delay depends on: 1) the time elapsed from retrieval of X_0 to the time that block X_{m-i} is retrieved from zone Z_0, termed $T_{accessZ_0}$, and 2) the amount of data retrieved during $T_{accessZ_0}$. If the display time of data corresponding to item 2 ($T_{display(m-i)}$) is lower than $T_{accessZ_0}$, then the display must be delayed by $T_{accessZ_0} - T_{display(m-i)}$. To illustrate, assume that X_0 is assigned to the innermost zone Z_{m-1} (i.e., $i = m-1$) and the display time of each of its block is 4.5 seconds, i.e., $T_{display(1)} = 4.5$ seconds. If 10 seconds elapse from the time X_0 is read until X_1 is read from Z_0 then the display of X must be delayed by 5.5 seconds relative to its retrieval from Z_{m-1}. If its display is initiated upon retrieval, it may suffer from a 5.5 second hiccup. This delay to avoid a hiccup is shorter than the duration of a T_{scan}. Indeed, the maximum latency observed by a request is T_{scan} when the number of active displays is less than[5] $\mathcal{N}$:

$$\ell \ = \ T_{Scan} \ = \ T_{cseek} + \sum_{i=0}^{m-1} T_{MUX}(Z_i) \tag{2}$$

This is because at most $\mathcal{N}-1$ displays might be active when a new request arrives referencing object X. In the worst case scenario, these requests might be retrieving data from the zone that contains X_0 (say Z_i) and the new request arrives too late to employ the available idle slot. (Note that the display may not employ the idle slot in the next T_{MUX} because Z_{i+1} is now active and it contains X_1 instead of X_0.) Thus, the display of X must wait one T_{scan} period until Z_i becomes active again.

One can solve for the block size by observing from Figure 3 that $T_{MUX}(Z_i)$ can be defined as:

$$T_{MUX}(Z_i) = \mathcal{N} \times (\frac{B}{\mathcal{R}(Z_i)} + T_{W\_Seek}) \tag{3}$$

[5] When the number of active displays exceeds $\mathcal{N}$ then this discussion must be extended with appropriate queuing models.

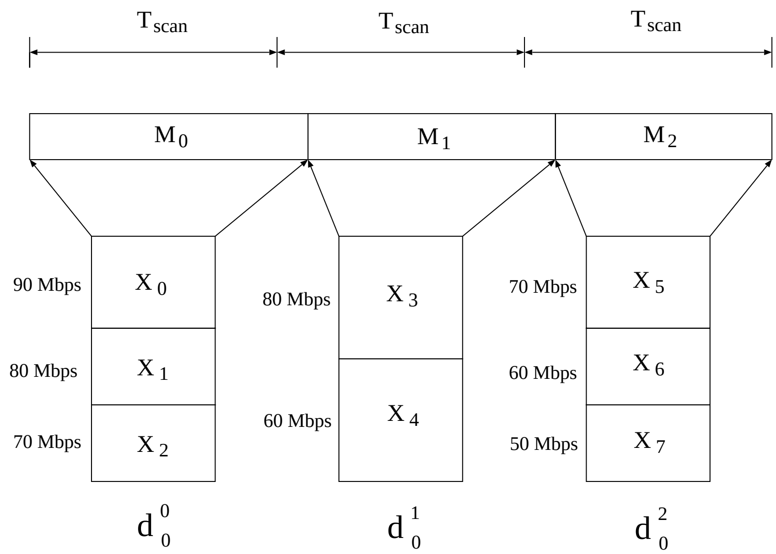


Figure 5: HetFIXB

Substituting this into Equation 1, the block size is defined as:

$$B = \frac{\mathcal{R}_C \times (T_{cseek} + m \times \mathcal{N} \times T_{W\_Seek})}{m - \mathcal{R}_C \times \sum_{i=0}^{m-1} \frac{\mathcal{N}}{\mathcal{R}(Z_i)}} \tag{4}$$

Observe that FIXB wastes disk space when the storage capacity of the zones is different. This is because once the storage capacity of the smallest zone is exhausted then no additional objects can be stored as they would violate a round-robin assignment[6].

2.5.2 Extensions of FIXB (HetFIXB)

With a heterogeneous collection of disks, we continue to maintain a T_{scan} per disk drive. While the duration of a T_{scan} is identical for all disk drives, the amount of data produced by each T_{scan} is different. We compute the block size for each disk model (recall that blocks are equi-sized for all zones of a disk) such that the faster disks compensate for the slower disks by producing more data during their T_{scan} period. HetFIXB aligns the T_{scan} of each individual disk drive with one another such that they all start and end in a T_{scan}.

[6]Unless the number of blocks for an object is less than m. We ignored this case from consideration because video objects are typically very large.

To support $\mathcal{N}$ simultaneous displays, HetFIXB must satisfy the following equations.

$$M = \sum_{i=0}^{K-1} M_i, \text{ where } M_i = m_i \times B_i$$

$$AvgR_i \ : \ AvgR_j \ = \ M_i \ : \ M_j, \ \ 0 \leq i, j < K$$

$$T_{scan} = Tp/K, \text{ where } Tp = \frac{M}{R_C}$$

$$T_{scan_i} = T_{cseek} + \sum_{j=0}^{m_i - 1} N(\frac{B_i}{R(Z_j(D_i))} + seek_i) \leq T_{scan}$$

where $0 \leq i < K$.

To illustrate, assume a configuration consisting of 3 disks, see Figure 5. Assume the average transfer rates of disks, $AvgR_0 = 80$ Mbps, $AvgR_1 = 70$ Mbps, and $AvgR_2 = 60$ Mbps respectively. When $R_C = 4$ Mbps, 1.5 Mbytes of data ($M = 1.5$ MB) is required every 3 seconds ($T_p = 3$ sec) to support a hiccup-free display. Based on the ratio among the average transfer rates of disk models, $M_0 = 0.5715$ MB, $M_1 = 0.5$ MB, and $M_2 = 0.4285$ MB. Thus, $B_0 = M_0/m_0 = 0.19$ MB, $B_1 = M_1/m_1 = 0.25$ MB, $B_2 = M_2/m_2 = 0.14$ MB. An object X is partitioned into blocks and blocks are assigned into zones in a round-robin manner. When a request for X arrives, the system retrieves X_0, X_1, and X_2 ($M_0 = 3 \times B_0$ amount of data) from D_0 during the first T_{scan}. A third of M (0.5 MB) is consumed during the same T_{scan}. Hence, some amount of data, 0.0715 MB, remains un-consumed in the buffer. In the next T_{scan}, the system retrieves X_3 and X_4 ($M_1 = 2 \times B_1$ amount of data) from D_1. While the same amount of data (0.5 MB) is retrieved and consumed during this T_{scan}, the accumulated data (0.0715 MB) still remains in the buffer. Finally, during the last T_{scan}, the system retrieves X_5, X_6, and X_7 ($M_2 = 3 \times B_2$ amount of data) from D_2. Even though the amount of data retrieved in this T_{scan} (0.4285 MB) is smaller than the amount of data displayed during a T_{scan} (0.5 MB), there is no starvation because 0.0715 megabytes of data is available in the buffer. This process is repeated until the end of display.

2.6 Heterogeneous Deadline Driven (DD)

With this technique, a client issues block requests, each tagged with a deadline. Each disk services block requests with the Earliest Deadline First policy. In [10], we showed that the assignment of blocks to the zones should be independent of the frequency of access to the blocks. Thus, blocks are assigned to the zones in a random manner. The size of the blocks assigned to each disk model is

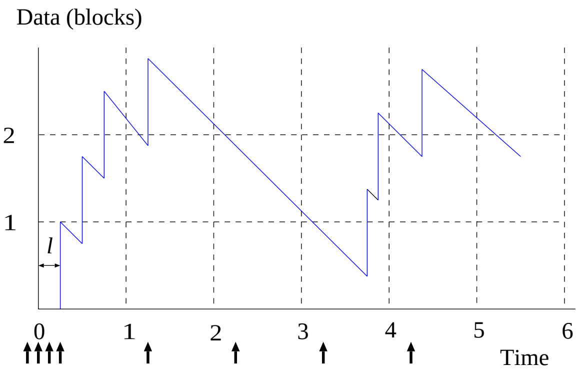


Figure 6: Deadline driven

different. They are determined based on the average weighted transfer rate of each disk model. Let WR_i denote the weighted average transfer rate of disk model i:

$$WR_i = \sum_{j=0}^{m_i-1} [S(Z_j(D_i)) \times R(Z_j(D_i)) / \sum_{k=0}^{m_i-1} S(Z_k(D_i))]$$

$$WR_i \ : \ WR_j \ = \ B_i \ : \ B_j, \ \ 0 \leq i,j < K$$

Assuming $B_i \geq B_j$ where $i < j$ and $0 \leq i,j < K$, an object X is divided into blocks such that the size of each block X_i is $B_{i \ mod \ K}$. Blocks with the size of B_i are randomly assigned to disks belonging to model i. A random placement may incur hiccups that are attributed to the statistical variation of the number of block requests per disk drive, resulting in varying block retrieval time. Traditionally, double buffering has been widely used to absorb the variance of block retrieval time: while a block in a buffer is being consumed, the system fills up another buffer with data. However, we generalize double buffering to N buffering and prefetching N-1 buffers before initiating a display. This minimize the hiccup probability by absorbing a wider variance of block retrieval time, because data retrieval is N-1 blocks ahead of data consumption.

We assume that, upon a request for a video clip X, a client: (1) concurrently issues requests for the first N-1 blocks of X (to prefetch data), (2) taggs the first N-1 block requests with a deadline equivalent to display time of a block, (3) starts display as soon as the first prefetched block arrives. For example, in Figure 6, first three blocks are requested at the beginning. Then, the next block request is issued immediately after a block in the buffer is consumed. Obviously, there are other ways of deciding both the deadline of the prefetched blocks and when to initiate display blocks. In [10], we analyzed the impact of these alternative decisions and demonstrated that the combination of the above two choices enhances system performance.

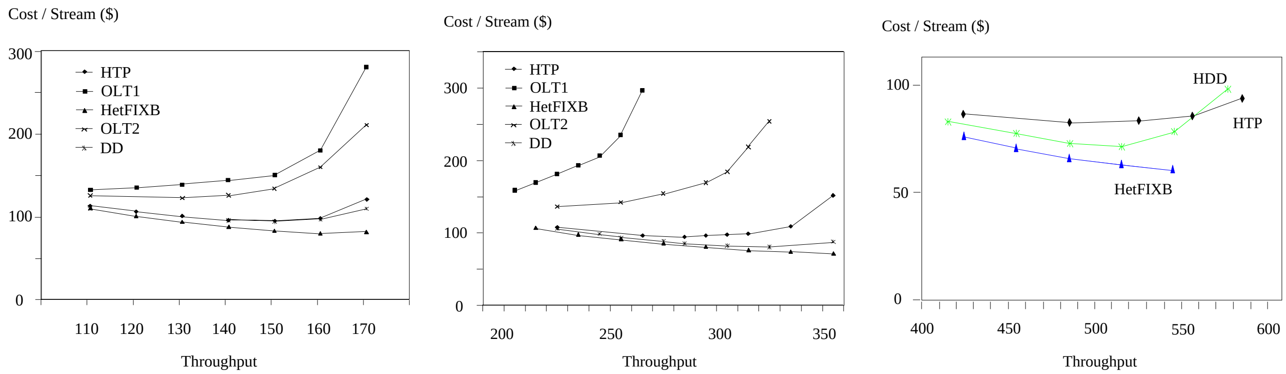
3 Performance Evaluation

In this section, we quantify the performance tradeoffs associated with alternative techniques. While OLT1, OLT2, TP and HetFIXB were quantified using analytic models, DD was quantified using a simulation study. We conducted numerous experiments analyzing different configurations with different disk models from Quantum and Seagate. Here, we report on a subset of our results in order to highlight the tradeoffs associated with different techniques. In all results presented here, we used the three disk models shown in Table 1. Both Barracuda 4LP and 18 provide a 7200 rpm while the Cheetah provides a 10000 rpm. Moreover, we assumed that all objects in the database require a 4 Mbps bandwidth for their continuous display.

Figure 7 shows the cost per stream as a function of the number of simultaneous displays supported by the system (throughput) for three different configurations. Figure 7.a shows a system that is installed in 1994 and consists of 10 Barracuda 4LP disks. Figure 7.b shows the same system two years later when it is extended with 10 Cheetah disks. Finally, Figure 7.c shows this system in 1998 when it is extended with 10 Barracuda 18 disks. To estimate system cost, we assumed: a) the cost of each disk at the time when they were purchased with no depreciation cost, and b) the system is configured with sufficient memory to support the number of simultaneous displays shown on the x-axis. We assumed that the cost of memory is \$7/MB, \$5/MB, and \$3/MB in 1994, 1996, and 1998, respectively. Additional memory is purchased at the time of disk purchases in order to support additional users. (Once again, we assume no depreciation of memory.) While one might disagree with our assumptions for computing the cost of the system, note that the focus of this study is to compare the different techniques. As long as the assumptions are kept constant, we can make observations about the proposed techniques and their performance tradeoff.

In these experiments, OLT2 constructed logical zones in order to force all disk models to consist of the same number of zones. This meant that OLT2 eliminated the innermost zone (zone 10) of Barracuda 4LP, splitting the fastest three zones of Cheetah into six zones, and splitting the outermost zone of Barracuda 18 into two. Figure 7.c does not show OLT1 and OLT2 because: a) their cost per stream is almost identical to that shown in Figure 7.b, and b) we wanted to show the difference between HetFIXB, DD, and HTP.

Figure 7 shows that HetFIXB is the most cost effective technique, however, it supports fewer simultaneous displays as a function of heterogeneity. For example, with one disk model, it provides a throughput similar to the other techniques. However, with 3 disk models, its maximum throughput is lower than those provided by DD and HTP. This is dependent on the physical characteristics



a. One disk model (homogeneous) b. Two disk models (heterogeneous) c. Three disk models (heterogeneous)

Figure 7: Throughput and cost per stream

of the zones because HetFIXB requires the duration of T_{scan} to be identical for all disk models. This requirement results in fragmentation of the disk bandwidth which in turn limits the maximum throughput of the system. Generally speaking, the greater the heterogeneity, the greater the degree of fragmentation. However, the zone characteristics ultimately decide the degree of fragmentation. One may construct logical zones in order to minimize this fragmentation, see [9]. This optimization is not reported because of strict space limitations imposed by the call for paper. It raises many interesting issues that are not presented here. Regardless, the comparison shown here is fair because our optimizations are applicable to all techniques.

OLT1 provides inferior performance as compared to the other techniques because it wastes a significant percentage of the available disk bandwidth. To illustrate, Figure 8 shows the percentage of wasted disk bandwidth for each disk model with each technique when the system is fully utilized (the trend holds true for less than 100% utilization). OLT1 wastes 60% of the bandwidth provided by Cheetah and approximately 30% of Barracuda 18. Most of the wasted bandwidth is attributed to these disks sitting idle. Cheetahs sit idle because they provide a 10,000 rpm as compared to 7200 rpm provided by the Barracudas. Barracuda 4LP and 18 disks sit idle because of their zone characteristics. In passing, while different techniques provide approximately similar cost per performance ratios, each wastes bandwidth in a different manner. For example, both HTP and HetFIXB provide approximately similar cost per performance ratios, HTP wastes 40% of Cheetah's bandwidth while HetFIXB wastes only 20% of the bandwidth provided by this disk model. HTP makes up for this limitation by harnessing a greater percentage of the bandwidth provided by Barracuda 4LP and 18.

Figure 9 shows the maximum latency incurred by each technique as a function of the load imposed on the system. In this figure, we have eliminated OLT1 because of its prohibitively high latency (One

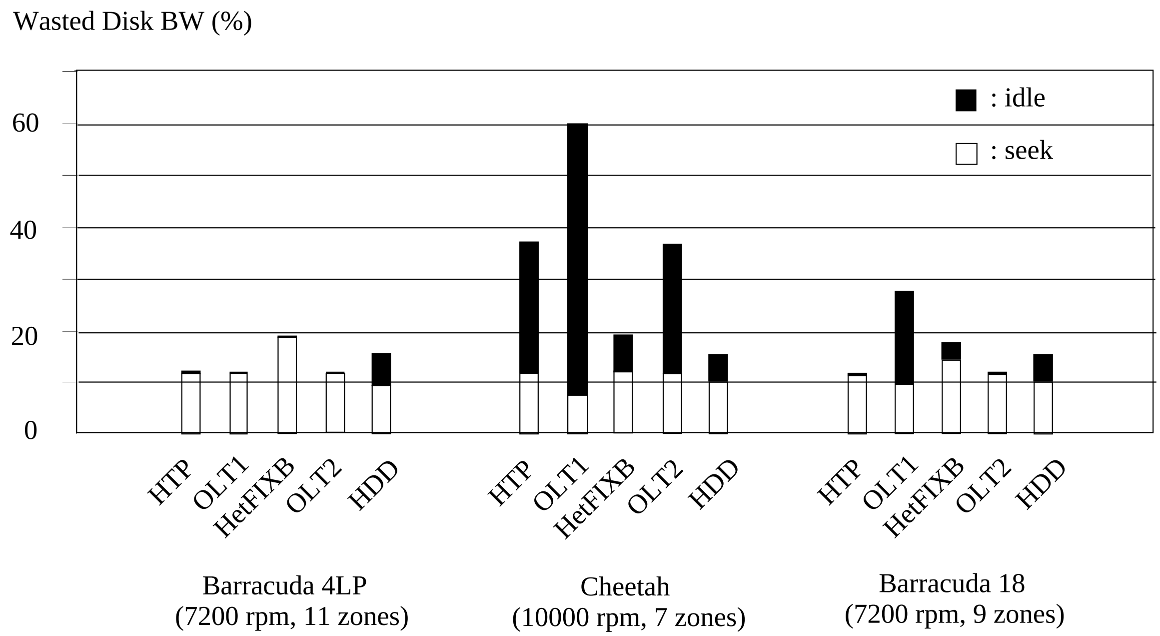


Figure 8: Wasted disk bandwidth

conclusion of this study is that OLT1 is not a competitive strategy.) The results show that HetFIXB provides the worst latency while DD's maximum latency is below 1 second. This is because HetFIXB forces a rigid schedule with a disk zone being activated in an orderly manner (across all disk drives). If a request arrives and the zone containing its referenced block is not active then it must wait until the disk head visits that zone (even if idle bandwidth is available). With DD, there is no such a rigid schedule in place. A request is serviced as soon as there is available bandwidth. Of course, this is at the risk of some requests missing their deadlines. This happens when many requests collide on a single disk drive due to random nature of requests to the disks. In these experiments, we ensured that such occurrences impacted one in a million requests, i.e., a hiccup probability is less than one in a million block requests.

OLT2 and HTP provide a better latency as compared to HetFIXB because they construct fewer logical disks [2, 12]. While OLT2 constructs a single logical disk, HTP constructs 10 logical disks, and HetFIXB constructs 30 logical disks. In the worst case scenario (assumed by Figure 9), with both HTP and HetFIXB, all active requests collide on a single logical disk (say $d_{bottleneck}$). A small fraction of them are activated while the rest wait for this group of requests to shift to the next logical disk (in the case of HetFIXB, they wait for one T_{scan}). Subsequently, another small fraction is activated on $d_{bottleneck}$. This process is repeated until all requests are activated. Figure 9 shows the incurred latency by the last activated request.

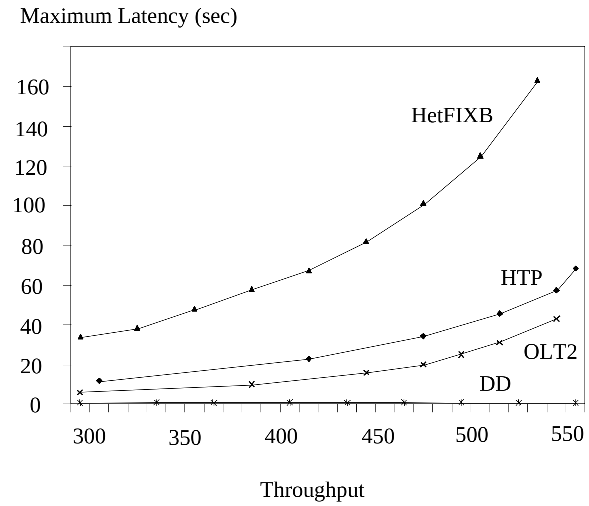


Figure 9: Maximum startup latency

With three disk models (Figure 7.c), OLT1 and OLT2 waste more than 80% of disk space, HTP and DD waste approximately 70% of disk space, while HetFIXB wastes 44% of the available disk space. However, this does NOT mean that HetFIXB is more space efficient than other techniques. This is because the percentage of wasted disk space is dependent on the physical characteristics of the participating disk drives: number of disk models, number of zones per disk, track size of each zone, storage capacity of individual zones and disk drives. For example, with two disk models (Figure 7.b), HetFIXB wastes more disk space when compared with the other techniques.

4 Conclusion and Future Directions

In this study, we quantified the tradeoff associated with alternative multi-zone techniques when extended to a configuration consisting of heterogeneous disk drives. Ignoring OLT1, our principle result is that no single strategy dominates on all metrics: throughput, startup latency, cost per simultaneous display, and wasted disk space. All proposed techniques strive to distribute the load of a single display evenly across the available disk bandwidth in order to prevent formation of bottlenecks. They belong to a class of algorithm that is commonly termed non-partitioning. An alternative approach might have been to partition resources into clusters and treat each cluster as an independent server. For example, with a configuration consisting of 3 disk models, we could have constructed three servers and assigned objects to different servers with the objective to distribute the workload of the system as evenly as possible [7]. The system would replicate popular clips across multiple servers in order to prevent formation of bottlenecks [13, 6]. Using this approach, one could optimize system parameters (such as block size) for each configuration independently in order to maximize the performance of each subserver. This is ideal for static workloads that do not change overtime. However, for dynamic workloads, one must employ detective techniques that monitor the frequency of access to objects and replicate popular objects in order to prevent formation of bottlenecks. In [26], we utilize a simulation model to show that this approach is generally inferior to a non-partitioning scheme. This is because detective techniques must wait for formation of bottlenecks prior to eliminating them [5].

In addition, recently we have quantified fault-tolerant characteristics of a general non-partitioning scheme for heterogeneous single-zone disk drives [25]. While extensions of HTP to incorporate results of [25] are trivial, it is not clear how HetFIXB and DD are impacted by the design of [25]. This is another future research direction of this study.

References

[1] J. Al-Marri and S. Ghandeharizadeh. An Evaluation of Alternative Disk Scheduling Techniques in Support of Variable Bit Rate Continuous Media. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, Valencia, Spain, March 23-27, 1998.

[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*, pages 79–89, 1994.

[3] S. Berson, L. Golubchik, and R. R. Muntz. A Fault Tolerant Design of a Multimedia Server. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 364–375, 1995.

[4] Y. Birk. Track-pairing: a novel data layout for vod servers with multi-zone-recording disks. In *IEEE International Conference on Multimedia Computing and System*, June 1995.

[5] H. Chou and D. J. DeWitt. An Evaluation of Buffer Management Strategies for Relational Database Systems. In *Proceedings of the International Conference on Very Large Databases*, 1985.

[6] A. Dan, M. Kienzle, and D. Sitaram. A Dynamic Policy of Segment Replication for Load-balancing in Video-on-Demand Computer Systems. In *ACM Multimedia Systems*, number 3, pages 93–103, 1995.

[7] A. Dan and D. Sitaram. An Online Video Placement Policy based on Bandwidth toSpace Ratio (bsr). In *Proceedings of ACM SIGMOD*, 1995.

[8] C. Freedman and D. DeWitt. The SPIFFI Scalable Video-on-Demand System. In *SIGMOD Conference*, 1995.

[9] 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.

[10] S. Ghandeharizadeh and S.H. Kim. Design of Multi-user Editing Servers for Continuous Media. *To appear in the Multimedia Tools and Applications Journal.*

[11] S. Ghandeharizadeh and S.H. Kim. Striping in Multi-disk Video Servers. In *High-Density Data Recording and Retrieval Technologies*, pages 88–102. Proc. SPIE 2604, October 1995.

[12] S. Ghandeharizadeh, S.H. Kim, W. Shi, and R. Zimmermann. On Minimizing Startup Latency in Scalable Continuous Media Servers. In *Proceedings of Multimedia Computing and Networking*, pages 144–155. Proc. SPIE 3020, Feb. 1997.

[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] E. Grochowski. Disk Drive Price Decline. In *IBM Almaden Research Center*, 1997.

[15] S.H. Kim and S. Ghandeharizadeh. Design of Multi-user Editing Servers for Continuous Media. In *International Workshop on the Research Issues in Data Engineering (RIDE'98)*, Feb. 1998.

[16] M. Leung, J. C. Lui, and L. Golubchik. Buffer and I/O Resource Pre-allocation for Implementing Batching and Buffering Techniques for Video-On-Demand Systems. In *Proceedings of the International Conference on Data Engineering*, 1997.

[17] R. Muntz, J. Santos, and S. Berson. RIO: A Real-time Multimedia Object Server. *ACM Sigmetrics Performance Evaluation Review*, 25(2), Sep. 1997.

[18] G. Nerjes, P. Muth, and G. Weikum. Stochastic service guarantees for continuous data on multi-zone disks. In *the 16th Symposium on Principles of Database Systems (PODS'97)*, May 1997.

[19] B. Ozden, R. Rastogi, and A. Silberschatz. Disk Striping in Video Server Environments. In *IEEE International Conference on Multimedia Computing and System*, June 1996.

[20] D. A. Patterson. Terabytes >> Teraflops (or Why Work on Processors When I/O is Where the Action is? In *Key note address at the ACM-SIGMETRICS Conference*, 1993.

[21] M.F. Mitoma S.R. Heltzer, J.M. Menon. Logical data tracks extending among a plurality of zones of physical tracks of one or more disk devices. In *U.S. Patent No. 5,202,799*, April 1993.

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

[23] H.M. Vin, S.S. Rao, and P. Goyal. Optimizing the Placement of Multimedia Objects on Disk Arrays. In *IEEE International Conference on Multimedia Computing and System*, May 1995.

[24] J. Wong, K. Lyons, D. Evans, R. Velthuys, G. Bochmann, E. Dubois, N. Georganas, G. Neufeld, M. Ozsu, J. Brinskelle, A. Hafid, N. Hutchinson, P. Iglinski, B. Kerherve, L. Lamont, D. Makaroff, and D. Szafron. Enabling Technology for Distributed Multimedia Applications. In *IBM System Journal*, 1997.

[25] R. Zimmermann and S. Ghandeharizadeh. Continuous Media Placement and Scheduling in Heterogeneous Disk Storage Systems. In *Submitted to ACM Transactions of Database Systems*.

[26] R. Zimmermann and S. Ghandeharizadeh. Continuous Display Using Heterogeneous Disk-Subsystems. In *Proceedings of ACM Multimedia Conference*, Nov. 1997.