

On Coordinated Display of Structured Video

Martha L. Escobar-Molano
University of South Florida

Shahram Ghandeharizadeh
University of Southern California

This article presents a taxonomy of resource scheduling techniques that enable a multidisk hardware platform to support coordinated displays. It describes three techniques that use the available disk bandwidth to minimize latency and memory requirements. A simulation study quantifies the trade-offs associated with these techniques.

Representing video follows one of two approaches:¹ stream-based or structured. With the stream-based approach, a video clip consists of a sequence of frames displayed at a prespecified rate (30 frames per second, for example). The structured approach represents a presentation as a collection of objects (such as 3D representations of a dinosaur) with spatial and temporal constraints (for example, the dinosaur's positions and time of their appearances) along with their rendering features (light intensity, viewpoint, and so on). The structured approach permits both information reuse and development of effective query processing techniques and is useful for producing animated sequences. Disney's *Toy Story* and the television show *Reboot*, for example, were generated using the structured approach.²

One optimal resource scheduling technique³ guarantees a coordinated display of structured presentations for single-disk architectures. We extend this method to multidisk architectures. Our target hardware platform (illustrated in Figure 1) consists of a fixed amount of memory (DRAM) and D homogeneous magnetic disk drives that contain the objects participating in a structured presentation. The unit of transfer from each disk has a fixed size and is termed a *page*. An object can be smaller or larger than a disk page; an object x that is larger than a disk page is represented as a collection of pages. It is first partitioned into $k = \lceil \text{size of } x / \text{size of a page} \rceil$ disk pages; then these pages are assigned to the D disks. The system may cluster

several small objects in a single disk page. Each disk can perform independent reads and writes. The D disk drives may retrieve D pages into D different memory frames at the same time.

To render a real-time coordinated display of a structured presentation, the system must retrieve the objects that constitute a scene and display them according to prespecified temporal and spatial constraints. Since these objects are contained in a set of pages, a *display schedule* defines which pages must be in memory at what time to satisfy the temporal constraints. The storage manager processes the display schedule to generate a resource schedule that satisfies the temporal constraints. Based on this schedule, the system delivers data from disk to memory. The rendering software processes the pages in memory to display the presentation. We assume that the disk bandwidth is the bottleneck resource (that is, the CPU is fast enough to process the data as long as the required data is in memory) since, in general, CPU speed is increasing much faster than disk bandwidth.

An unbalanced placement of objects across the disk drives with respect to a display schedule might cause a subset of disks to become temporary bottlenecks. An uncoordinated display results when the resource schedule cannot render an object memory-resident to support the display schedule. We can detect these bottlenecks in advance because when the display is activated, both the display schedule and data placement are prespecified. The system may resolve these bottlenecks in several ways. Some techniques manipulate object placement across the disks during display to resolve bottlenecks. Our simulation study results indicate that such techniques can significantly reduce the amount of memory a display requires.

Previous studies have investigated techniques that ensure a continuous display of stream-based presentations.^{4,5} These studies conceptualize a presentation as a file read sequentially at a prespecified rate. They also assume a data layout that ensures a regular disk reference pattern—that is, reading the first block of a presentation from disks 0, 1, and 2 during the first time cycle, reading the second block from disks 3, 4, and 5 during the second time cycle, and so on.

The disk reference pattern of a structured presentation, on the other hand, consists of objects referenced and displayed based on a prespecified schedule. A schedule may reference a single object at different times, and several presentations may share the object. These conceptual differences ren-

Definition of Terms

m	Number of time intervals in a video
t	Duration of a time interval
Time interval i	Period $[i, i + 1]$
Instant i	The beginning of time interval i
S_i	Pages in memory at instant i
P_i	Pages containing objects displayed during interval i
F_j^d	Pages retrieved from disk d onto memory during interval i
F_i	Pages retrieved from disk onto memory during interval i : $\sum_{d=0}^{D-1} F_j^d$
K_i	Pages swapped out during interval i
R_j^a	Read page a from disk j
U_j^d	Pages written to disk d during interval i
U_i	Pages written to disk during interval i : $\sum_{d=0}^{D-1} U_j^d$
W_j^a	Write page a to disk j
D	Disk drives in the system
B	Maximum number of pages read by a drive during an interval
C	Number of memory frames

der the above-referenced studies irrelevant for structured presentations. A stream-based study would relate only if it assumed that a movie's frames are randomly dispersed across the available disks and a frame might be displayed at different times during a regular display.

In an earlier study,³ we described a resource scheduler that realizes a coordinated display for a structured presentation assuming a single disk architecture. That scheduler minimized both start-up latency and required memory. Here we extend it by focusing on a multidisk hardware platform. We demonstrate here that our earlier scheduling technique is not appropriate for a multidisk architecture. The placement of objects across multiple disks impacts both the memory required to support a display and its incurred latency. We thus propose scheduling techniques that manipulate data placement for the target application and hardware platform.

Coordinated display

To simplify the discussion without loss of generality, we partition time into fixed units, *time intervals*. The duration of each interval is t . A collection of objects, represented as a set of disk pages, is displayed during each interval. *Time*

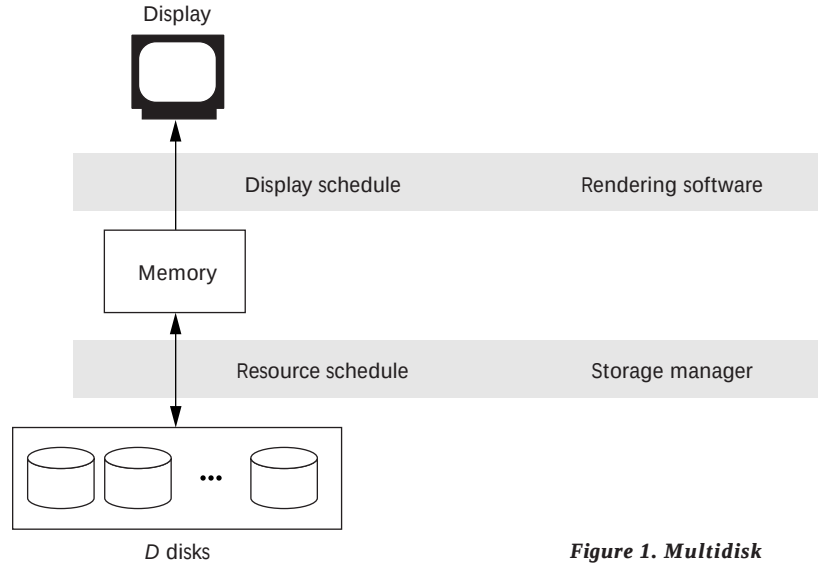


Figure 1. Multidisk system architecture.

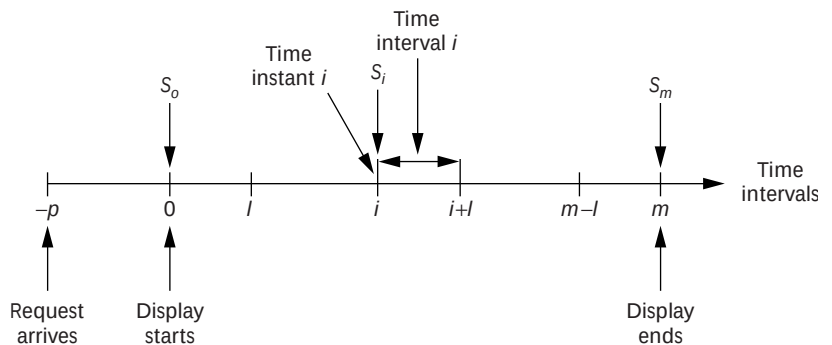


Figure 2. Time interval and time instant.

instant i denotes the beginning of a time interval i ; we represent a display schedule as a sequence of time instants (see Figure 2). Each time instant i is tagged with a collection of disk pages, P_i , displayed during interval i . We assume that the system can support a coordinated display if the set P_i is memory resident at instants i and $i + 1$, $0 \leq i < m$.

To reduce the observed startup latency and required memory, the scheduling techniques we discuss here overlap the display and retrieval of disk pages and manipulate the placement of data on disks. P_i should be retrieved into memory during interval $i - 1$ to minimize the memory required. However, this ideal situation might be infeasible at times because the pages that constitute P_i might be unevenly dispersed across the disks, exhausting the bandwidth of one or more disks (while other disks are idle) such that they fail to retrieve the set P_i during an interval. Note that

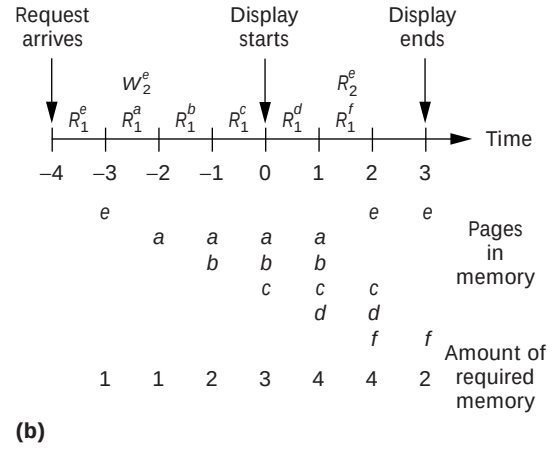
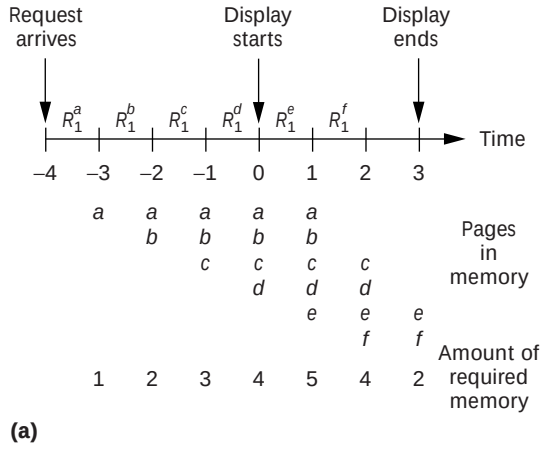


Figure 3. Resource schedules (a) without and (b) with manipulation of the data placement.

in this scenario the disks' total bandwidth suffices; the primary limitation is the placement of data which, in combination with the display schedule, results in a bottleneck.

The system may pursue two solutions to avoid bottlenecks:

1. retrieve some pages of P_i during earlier intervals such as $i - 2$ or $i - 3$ (*prefetched* pages), or
2. manipulate data placement prior to interval i to distribute P_i more evenly across disks.

Our *resource scheduler* consumes a display schedule and produces a schedule of

1. page retrievals from disks to memory,
2. page writes to modify data placement, and
3. page discards (from memory) to accommodate new retrievals.

Similar to a display schedule, a *resource schedule* is represented as a sequence of instants. The duration of an interval for the resource schedule matches that of the display schedule. Associated with instant i is

1. a collection of pages retrieved from each of the D disks during interval i , denoted F_i^d ,
2. a collection of pages written to each of the D disks during interval i to change the placement of data, denoted U_i^d , and
3. a collection of pages discarded from memory to accommodate these retrievals, denoted K_i .

The set of pages retrieved and discarded during an interval are disjoint. Otherwise, the system would perform wasteful work.

The number of pages that can be retrieved from or written to a disk during an interval is bounded (denoted as B). When computing the resource schedule, the scheduler ensures that the number of pages retrieved from or written to a disk during interval i does not exceed B . We define the state of memory (pages occupying memory frames) at each instant i , S_i , based on the current set of pages that occupy memory, those discarded from memory, those flushed to disks, and those retrieved from different disks. Given a system with D disks, the state of memory at each instant i is defined as

$$S_i = \left(S_{i-1} - K_{i-1} - \left(U_{i-1}^0 \cup \dots \cup U_{i-1}^{D-1} \right) \right) \cup \left(F_{i-1}^0 \cup \dots \cup F_{i-1}^{D-1} \right)$$

The number of pages in S_i should be lower than or equal to the C frames that constitute the memory.

Consider a display schedule that consists of three intervals: $P_0 = \{a, b\}$, $P_1 = \{c, d\}$, and $P_2 = \{e, f\}$. Assume that the system consists of four disks ($D = 4$), each with the bandwidth to retrieve or write one disk page during an interval ($B = 1$). Assuming all the referenced pages reside on disk 1, Figure 3a shows a resource schedule that supports a coordinated display without changing the data placement (that is, only retrievals and discards). R_1^a denotes that page a is retrieved from disk 1. In this figure, a negative instant corresponds to page retrievals performed prior to the display. A page might be retrieved during the interval prior to its display (such as f) or prefetched at an earlier interval (such as a). For

example, five frames of memory are allocated at instant 1 (a, b, c, d, e), while the display schedule dictates that only four should be allocated (a, b, c, d). Page e is prefetched for later use and increases the system's memory requirements.

This example illustrates how an unbalanced schedule of references to disks might result in formation of bottlenecks on some disks, requiring the system to prefetch pages, while other disks remain idle. In our example, while four disks could accommodate the retrieval of four pages, the system is forced to prefetch pages because they all reside on disk 1. To remedy this, the scheduler could construct resource schedules that use the idle disk bandwidth to minimize the number of prefetched pages.

Figure 3b shows one such schedule. The system reads page e from disk 1 during interval -4 and replicates or migrates it to disk 2 (denoted W_2^e) during interval -3 . This allows the system to free the memory frame occupied by e at instant -2 and use disk 2 to retrieve e during interval 1 to satisfy the display schedule. This schedule requires only four memory frames at instant 1 (a, b, c, d). If a page is replicated, its original copy is termed primary while its other copies are termed secondary. A disadvantage of constructing secondary copies of a page is that it wastes both disk bandwidth and disk space. Once the display is completed, the system may either keep the primary and secondary copies of e or maintain one copy of e . In the first case (and the second when e is migrated or the primary copy of e on disk 1 is deleted), a subsequent display of the schedule would incur a three-fold interval startup latency because there is no need to replicate or migrate e .

Resource schedules

Computing a resource schedule that supports the coordinated display of a structured presentation and minimizes latency is an NP-hard problem.² Figure 4 shows a taxonomy of the possible heuristics for resource scheduling, categorized into memory-based, replication-based, and migration-based. Memory-based (Figure 3a) employs prefetching to support a coordinated display. It has no impact on data placement across the available disks. Replication-based (Figure 3b) detects bottlenecks and identifies those pages that cause them. It resolves bottlenecks by constructing copies of pages during earlier intervals. With transient replication, the system deletes the secondary copies at the end of a display to restore the prior placement of data. In the example shown in Figure 3b, tran-

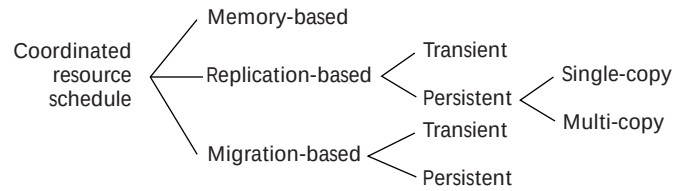


Figure 4. Taxonomy of possible approaches to resource scheduling.

sient replication would delete the secondary copy of e from disk 2 at the end of the display.

With persistent replication, the display of a presentation might produce a new data organization. This technique is categorized into single-copy and multicopy. With single-copy persistent replication, disk space is a scarce resource and the system maintains only one copy of each page at the end of display. In the example of Figure 3b, this technique might delete the primary copy of e from disk 1, making the secondary copy on disk 2 its primary copy. With multicopy persistent replication, the system maintains the secondary copies on disk.

With migration, when the system migrates e to disk 2 during interval -3 , it deletes the primary copy of e from disk 1. This is useful when disk space is scarce and cannot accommodate secondary copies during the display. Migration does not create additional copies of a disk page during the display, whereas single-copy replication-based techniques might create additional copies during the display and then remove them at the end. With persistent migration, the new assignment of e persists at the end of display. With transient migration, the system remembers the original placement of data across the disks and restores it at the end of the display. This would require additional disk operations at the end of a display, potentially delaying other presentations.

Memory-based

Given a display schedule and the current data placement across the D disks, this technique extracts the display schedule for each disk based on the pages that reside on that disk. It invokes the greedy scheduler¹¹ using the display schedule of each disk to compute a resource schedule that consists of retrievals and discards (for each i , $U_i^0 = \dots = U_i^{p-1} = \emptyset$) for that disk. The union of these D resource schedules yields a final resource schedule. The disk with the longest startup latency (p) determines the display's overall latency.

For any data placement, this schedule minimizes required memory because the greedy scheduler minimizes the memory requirement at each instant i for a single disk.³ By minimizing S_0 , the

```

IncLatency
  if  $i = p - 1$ 
    then  $p = p - 1$ ;
    for  $d = 0$  to  $D - 1$  do  $F_{p-1}^d = 0$  endfor

Stage 1
   $L = \emptyset$ ;  $S_m = P_{m-1}$ 
  for  $i = m - 1$  to  $0$  do
    if  $i = 0$  then  $RS_i = P_i$ 
    else  $RS_i = P_i \cup P_{i-1}$ 
    targets =  $\emptyset$ ; availbw = 0
    for  $d = 0$  to  $D - 1$  do in[d] = 0 endfor
    for each  $a \in S_{i+1} - RS_i$  do
      if there exists drive  $d$  with a copy of  $a$ 
        such that in[d] <  $B$ 
        then residence( $a$ ) =  $d$ ;
        in[d] = in[d] + 1
        else residence( $a$ ) =  $d$  for some  $d$ 
        with a copy of  $a$ 
    endfor
    for  $d = 0$  to  $D - 1$  do
      Let  $N_i^d$  be the sequence of pages in
         $S_{i+1} - RS_i$  whose residence is  $d$ ,
        sorted by last( $\bullet$ ,  $i$ ).
       $F_i^d = \{\text{the first } B \text{ pages in } N_i^d\}$ 
      if  $|F_i^d| < B$ 
        then targets = targets  $\cup \{d\}$ ;
        availbw = availbw +  $B - |F_i^d|$ 
    endfor
    Let Disc be the sequence of pages in
       $S_{i+1} - RS_i - (F_i^0 \cup \dots \cup F_i^{D-1})$  sorted by
      last( $\bullet$ ,  $i$ )
    Xfer = {the first availbw pages in Disc}
    for each page  $a$  in Xfer do
       $L = L \cup \{a, i, \text{residence}(a), \text{targets}\}$ 
    endfor
     $S_i = RS_i \cup [(S_{i+1} - RS_i) - (F_i^0 \cup \dots \cup F_i^{D-1}) - Xfer]$ 
    if  $|S_i| > C$  then print("insufficient
      memory"); exit
    endfor
    /* schedule temporarily the retrievals of
       $S_0$  to count them in the computation supply
      */
     $p = 0$ ; for  $d = 0$  to  $D - 1$  do  $F_{p-1}^d = 0$  endfor
    for each  $a \in S_0$  do
      Let  $i$  be the largest time interval
        smaller than  $0$  such that  $|F_i^d| < B$  for
        some  $d$  with a copy of  $a$ 
       $F_i^d = F_i^d \cup \{a\}$  S0
    endfor

    IncLatency
    Add  $a$  to each memory state  $S_{i+1}, \dots, S_{-1}$ 
    endfor

Stage 2
  /* schedule temporarily the read of each
  replication to count them in the
  computation supply */
  for each  $e = \langle a, ti, d, \text{targets} \rangle \in L$  do
    Let  $i$  be the largest time interval
      smaller than  $ti$  such that  $|F_i^d| < B$ 
     $F_i^d = F_i^d \cup \{a\}$  S1
    IncLatency
  endfor
  /* designate one target disk to each
  replication in  $L$  */
   $L' = \emptyset$ 
  for  $n = 1$  to  $|L|$  do
    Let  $d_{free}$  be the disk with the lowest
    demand to supply ratio
    Let  $e = \langle a, ti, d, \text{targets} \rangle$  be an element in
     $L$  such that:
     $d_{free} \in e(\text{targets})$  and
    for all  $e' \in L$  such that  $d_{free} \in$ 
     $e'(\text{targets})$ :
       $|e(\text{targets})| < |e'(\text{targets})|$  or
       $|e(\text{targets})| = |e'(\text{targets})|$  and the
      drive  $d' \in e(\text{targets}) \cup e'(\text{targets})$ 
      with the highest demand to supply
      ratio is in  $e(\text{targets})$ 
     $L = L - \{e\}$ ;  $L' = L' \cup \{a, ti, d, d_{free}\}$ ;
     $F_{ti}^{d_{free}} = F_{ti}^{d_{free}} \cup \{a\}$ 
    if  $|F_{ti}^{d_{free}}| \geq B$  then for each  $e' =$ 
       $\langle a', ti, d', \text{targets}' \rangle \in L$ : targets' =
      targets' -  $\{d_{free}\}$ 
    /* schedule temporarily the write of  $e$  to
    count it in the computation of supply */
    Let  $i$  be the largest time interval
      smaller than  $ti$  such that  $|F_i^{d_{free}}| < B$ 
     $F_i^{d_{free}} = F_i^{d_{free}} \cup \{a\}$  S2
    IncLatency
  endfor
  Undo the scheduling performed in statements
  S0, S1 and S2

```

Figure 5. Macro to increase latency and stages 1 and 2 of the replication-based heuristic.

greedy scheduler minimizes the incurred latency. However, this technique fails to display an object when S_i exceeds C . In this case, the system must employ a technique that manipulates the data placement, such as the replication technique described below.

The greedy scheduler (the optimality proof of which exists elsewhere¹) consumes a display schedule ($\{P_0, \dots, P_{m-1}\}$) and disk bandwidth (B) as input to produce a sequence $\{S_0, \dots, S_m\}$ of memory states from which we derive the retrievals (F_i^d) and discards (K_i). The scheduler traverses the display schedule backwards (starting with instant $m - 1$) to determine the disk pages to retrieve during each

interval. It assumes that only pages referenced during the final interval are in memory at the end of the display ($S_m = P_{m-1}$). To minimize prefetching, it strives to retrieve during interval i all pages in memory at instant $i + 1$ (S_{i+1}) that are not referenced during intervals i and $i - 1$ ($P_i \cup P_{i-1}$). If all pages in $S_{i+1} - (P_i \cup P_{i-1})$ cannot be retrieved during interval i because of insufficient disk bandwidth, the scheduler pushes the retrieval of some pages to an earlier interval. Pages a in $S_{i+1} - (P_i \cup P_{i-1})$ with the lowest last(a, i) (most recent instant at which they are referenced) are retrieved during interval i . The other retrievals are pushed to earlier intervals. For each interval i and page a , we define last(a, i)

to be the most recent instant j ($j \leq i$) at which page a was referenced. More formally, let $\text{last}(a, 0) = -\infty$ and for each $i > 0$ define

$$\text{last}(a, i) = \begin{cases} i & \text{if } a \in P_{i-1} \\ \text{last}(a, i-1) & \text{otherwise} \end{cases}$$

Replication based

This three-stage technique detects the bottlenecks created by unbalanced data placement and schedules replications to resolve them. During the first stage (see Figure 5), it constructs a schedule of retrievals for each disk d and a list L of replications. It strives to retrieve all pages in $S_{i+1} - (P_i \cup P_{i-1})$ that reside on d during interval i . If there is not enough bandwidth on d during i to retrieve these pages, it adds to L a replication for each page in excess of B . Each replication in L consists of a page identifier, the interval when the bottleneck occurs (i), the bottleneck disk (d), and a list of candidate disks with idle disk bandwidth during interval i . If there is no idle disk bandwidth during interval i , the system prefetches the page at an earlier interval ($i-1$, $i-2$, and so on).

The second stage shown in Figure 5 selects a target from the candidate disks for each element e in L . This stage computes a demand and supply value for each disk drive. While the demand for a disk quantifies how frequently that disk might be used as the target for containing a secondary copy of a page, its supply measures the disk's available bandwidth. If $e(\text{targets})$ denotes the candidate disks that can serve as target for e and $|e(\text{targets})|$ denotes the number of disks in that set, then

$$\text{demand}(d) = \sum_{e \in L \text{ and } d \in e(\text{targets})} \frac{1}{|e(\text{targets})|}$$

The reciprocal weights the demand of a candidate disk based on its degree of freedom, or how many other disks can substitute for d . For example, an element of L that has one target candidate (d) makes the demand for d higher than if it could choose between four possible target disks. If $|F_j^d|$ denotes the number of pages retrieved from d during interval j , then

$$\text{supply}(d) = \sum_{-p \leq j \leq m-1} (B - |F_j^d|)$$

A disk almost completely used during each interval will have a low supply value. A disk that participates as a possible candidate for many page replications might have a high demand value

depending on the number of disks that compete with it to serve as potential candidates.

Next, the replication-based method computes the demand-to-supply ratio of each disk. A disk with a high ratio has the most constraints and should be used with care because it could potentially resolve bottlenecks. Our heuristic picks the disk with the lowest demand-to-supply ratio (least constrained), d_{free} , and identifies the elements of list L that contain d_{free} as a possible candidate to resolve bottlenecks. It chooses the element with the fewest target candidate disks; several may qualify if their number of target candidate disks is identical and this is the smallest value. From these, the heuristic chooses the element with the most constrained other candidate disk (the highest demand-to-supply ratio). The satisfying element is modified to have d_{free} as the designated drive to contain the secondary copy, and a retrieval from d_{free} is scheduled during the interval when the bottleneck occurs. The supply and demand values for d_{free} are adjusted and this process is repeated until a target drive is designated for each element in L .

In the final stage (see Figure 6, next page), this heuristic extends the schedule of retrievals and discards of stages 1 and 2 with appropriate read and write operations to construct secondary copies. Each page to be replicated now has a source with its primary copy and a target disk to contain its secondary copy, d_{source} and d_{target} respectively. Moreover, this page must be replicated prior to the interval that the bottleneck occurred, $i_{\text{bottleneck}}$. This stage strives to construct secondary copies during the display (between 0 and $i_{\text{bottleneck}}$) to minimize startup latency. Moreover, the number of intervals elapsed from when a replicated page is read from d_{source} to the time this page is written to d_{target} is minimized to decrease the required memory. This heuristic discards the secondary copies constructed during the display if it is in transient mode. In multicopy persistent mode, it maintains the newly created secondary copies for future displays. In single-copy persistent mode, it selects which copy to maintain by monitoring the number of references to different copies during the display and deleting the least-referenced copies.

Hybrid

With a hybrid approach, the system uses either memory or replication to resolve a bottleneck. Hybrid employs replication as soon as the display reaches a prespecified memory threshold. Memory requirements below the threshold imply that solv-

Stage 3

```

/* schedule reads and writes of
replications and retrievals of pages
in  $S_0$  */
 $L'' = \emptyset$ ;  $s'_0 = S_0$ ;  $s''_0 = \emptyset$ ;  $i = m$ 
while  $L' \neq \emptyset$  or  $s'_0 \neq \emptyset$  do
/* schedule eligible reads of
replications */
for each replication  $e' = \langle a, ti, d_{source}, d_{target} \rangle \in L'$  whose read has not been
scheduled
and its write has been scheduled at
interval  $j > i$  and  $|F_i^{d_{source}}| < B$  do
 $F_i^{d_{source}} = F_i^{d_{source}} \cup \{a\}$ ;  $L' = L' - \{e'\}$ 
IncLatency
 $L'' = L'' \cup \{\langle a, ti, d_{source}, d_{target}, j \rangle\} \cup$ 
 $\{\langle a, ti, d_{source}, i, d_{target}, j \rangle\}$ 
endfor
/* schedule eligible writes of
replications */
for each replication  $e' = \langle a, ti, d_{source}, d_{target} \rangle \in L'$  whose write has not been
scheduled and  $ti > i$  and  $|F_i^{d_{target}}| < B$  do
 $F_i^{d_{target}} = F_i^{d_{target}} \cup \{a\}$ ;  $L'' = L'' \cup$ 
 $\{\langle a, ti, d_{source}, d_{target}, i \rangle\}$ 
IncLatency
endfor
/* schedule retrievals of pages in  $s'_0$  */
if  $i < 0$  then
for each  $a \in s'_0$  do
if  $|F_i^d| < B$  for some drive  $d$  with a
copy of  $a$  then
 $F_i^d = F_i^d \cup \{a\}$ ;  $s'_0 = s'_0 - \{a\}$ ;  $s''_0 =$ 
 $s''_0 \cup \{\langle a, d, i \rangle\}$ 
IncLatency
endif
endif
 $i = i - 1$ 
endwhile

```

```

/* reduce the memory requirements */
/* swap writes of replications with
retrievals of pages in  $S_0$  */
for each  $\langle a, ti, d_s, i, d_{target}, j \rangle \in L''$  and
 $\langle b, d_{target}, k \rangle \in s''_0$  and  $0 > j > k$  and  $k > i$ 
do  $L'' = L'' - \{\langle a, ti, d_s, i, d_{target}, j \rangle\} \cup$ 
 $\{\langle a, ti, d_s, i, d_{target}, k \rangle\}$ 
 $s''_0 = s''_0 - \{\langle b, d_{target}, k \rangle\} \cup \{\langle b, d_{target}, j \rangle\}$ 
 $F_j^{d_{target}} = F_j^{d_{target}} - \{a\} \cup \{b\}$ ;
 $F_k^{d_{target}} = F_k^{d_{target}} - \{b\} \cup \{a\}$ 
endfor
/* swap read of a replication with a write
of another replication */
for each  $e_1 = \langle a, ti_1, d, i_1, d_{target}, j_1 \rangle \in L''$ 
and  $e_2 = \langle b, ti_2, d_{source}, i_2, d, j_2 \rangle \in L''$  such
that
 $e_1 \neq e_2$  and  $i_2 < i_1$  and  $i_1 < j_2$  and  $j_2 < j_1$  do
 $L'' = L'' - \{\langle a, ti_1, d, i_1, d_{target}, j_1 \rangle\} \cup$ 
 $\{\langle a, ti_1, d, j_2, d_{target}, j_1 \rangle\}$ 
 $L'' = L'' - \{\langle b, ti_2, d_{source}, i_2, d, j_2 \rangle\} \cup$ 
 $\{\langle b, ti_2, d_{source}, i_2, d, i_1 \rangle\}$ 
 $F_{i_2}^d = F_{i_2}^d - \{a\} \cup \{b\}$ ;  $F_{i_2}^d = F_{i_2}^d - \{b\} \cup \{a\}$ 
endfor
reschedule reads of replications as close
as possible to the corresponding writes
reschedule writes of replications as close
as possible to the corresponding reads
compute the memory requirements of
retrievals of pages in  $S_0$ 
for each replication  $e$  in  $L''$  do
compute the memory requirements of  $e$ 
if the memory requirements of  $e$  exceed
the available memory
then reschedule  $e$  during an earlier
period so that they do not exceed the
available memory
endfor

```

Figure 6. Stage 3 of the replication-based heuristic.

ing a bottleneck with replications might impose unnecessary demands for both disk space and disk bandwidth; this method therefore resolves the bottleneck using memory. On the other hand, memory requirements above the threshold invoke replication of pages to resolve bottlenecks and reduce excessive memory demands that might increase latency. Therefore, it would be better to solve the bottlenecks with replications.

This technique resembles the heuristic described earlier; the only difference is that during stage 1, it employs prefetching if the amount of memory used is less than or equal to the threshold. Otherwise, it employs replication. When constructing the resource schedule, hybrid might toggle between the replication- and memory-based techniques depending on when the memory used reaches the specified threshold.

Evaluation

We quantified the trade-off associated with memory-based, replication-based, and hybrid

scheduling techniques using a simulation study. Each technique's performance depends on both data placement across the disks and the display schedule. To evaluate the techniques, we computed a theoretical minimum amount of memory and latency required to support a coordinated display. This conceptualizes D disks as a single disk that can support the retrieval of $B \times D$ pages during each interval, thus rendering data placement irrelevant. It employs the greedy algorithm¹¹ to compute both minimum required memory and incurred latency. This technique is theoretical because it might be impossible to employ the aggregate bandwidth of D disks with arbitrary schedules and data layouts.

Simulation model

The simulation model consisted of four modules (illustrated in Figure 7):

1. a *display schedule generator* that produced a presentation's disk page reference pattern,

2. a *greedy scheduler* that computed the theoretical minimum,
3. a *data placement generator* that assigned the pages constituting the database to the disk drives, and
4. a *multidisk resource scheduler* which generated a resource schedule that satisfied the temporal constraints imposed by the display schedule.

The resource scheduler implemented the memory-based, replication-based, and hybrid techniques along with their persistent (single and multi-copy) and transient operation modes.

For this evaluation, the display schedule generator produced a 24-minute sequence of scenes (Presentation 1). Each scene consisted of a collection of background objects and a collection of motions. A motion was represented as a sequence of deltas to an object (the first posture), where a delta is implemented as either a computation or a page retrieval that describes the required changes. Applied to an object, deltas provided the illusion of movement. The duration of each scene (dur_{scene}) was selected randomly from a collection of possible values. These values were generated by analyzing the duration of several scenes for the animations *Lion King* and *Reboot*. The number of pages constituting the first interval of a scene (that is, containing background objects and the first motion postures) were generated randomly using a value in the range of [1, 60]. We based this on the number of objects appearing simultaneously in these two animations. We assumed that the size of an object was smaller than a disk page. The specification of a large background object in the display schedule was equivalent to that of several small background objects. Similarly, a large object's motion was equivalent to several small object motions. The number of motions ($n_{motions}$) is a fraction of the number of pages referenced during the first interval. This fraction is a randomly chosen real value between 0 and 1 to represent scenes that range from static background to those that change rapidly. Deltas can translate into either disk retrievals or computations. A percentage (p) of the total number of deltas as retrievals was randomly generated assuming that a delta per motion per interval is required and that this per-

centage is distributed uniformly. We computed the number of deltas that translate into disk retrievals as $p \times n_{motions} \times dur_{scene}$. Finally, we assigned page identifiers to all pages in the first interval and all pages corresponding to deltas. We assumed that the disk placement of all pages in the database (both referenced and nonreferenced by Presentation 1) distributed them evenly across the D disk drives.

Figure 8 shows the number of page references per interval for Presentation 1. It represents the cardinality of each set P_i as a function of i . In our sample presentation, most page references occurred during the first few scenes of the movie. However, a page might be referenced during consecutive intervals (page $a \in P_i, P_{i+1}, P_{i+2}$, and so on). Therefore it is retrieved only once (before instant i).

We analyzed an alternative presentation (Presentation 2) that was the reflection of the one shown in Figure 8. In Presentation 2, most page references occurred during the last few scenes. Presentations 1 and 2 referenced 65,002 pages. The

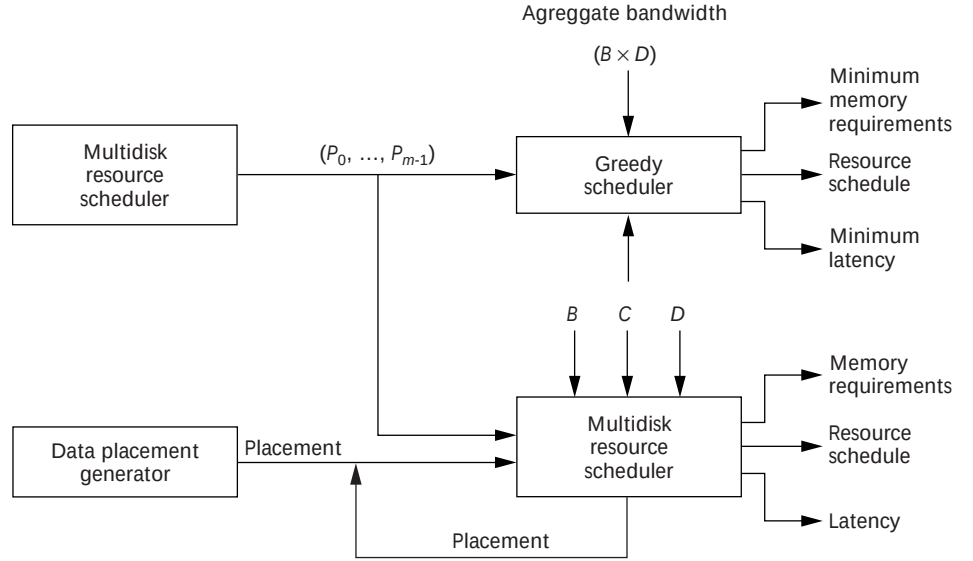


Figure 7. Simulation model.

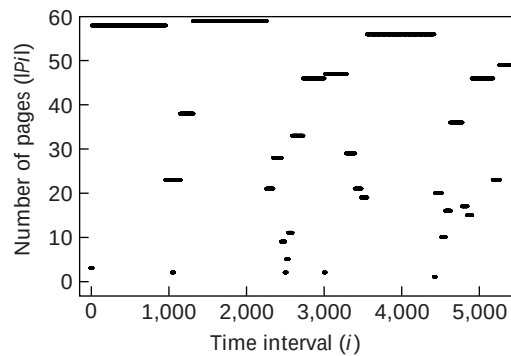


Figure 8. Number of pages referenced during each interval by Presentation 1.

Table 1. Performance of the three heuristics for Presentation 1 using seven disks.

Technique	Display	Latency	Resource Requirements						
			Memory		Bandwidth		Percent Overhead		
			Max	Avg	Max	Avg	D. Band	Mem	D. Space
Theoretical Minimum	N/A	1	117	78.79	59	33.82	N/A	N/A	N/A
Memory-Based	N/A	12	3155	445.41	63	33.63	N/A	N/A	N/A
Replication-Based, Persistent									
Multicopy	1st	94	703	175.79	63	38.84	14.39	57.50	22.66
	2nd	1	126	79.19	63	34.00	0.55	0.62	23.75
	3rd	1	118	78.79	59	33.83	0.01	0.00	23.77
Single-copy	1st	94	703	175.79	63	38.84	14.39	57.50	0
	2nd	85	1024	187.36	63	37.18	10.43	59.62	0
	8th	70	1024	181.53	63	36.84	9.36	58.08	0
Hybrid, Persistent									
Multicopy	1st	12	1024	263.58	63	34.92	3.38	23.73	4.92
	2nd	1	461	158.07	63	33.82	0	0	4.92
Single-copy	1st	12	1024	263.58	63	34.92	3.38	23.73	0
	2nd	2	1024	259.96	63	34.61	2.31	25.81	0

Table 2. Measurements obtained for Presentation 1 using eight disks.

Technique	Display	Latency	Resource Requirements						
			Memory		Bandwidth		Percent Overhead		
			Max	Avg	Max	Avg	D. Band	Mem	D. Space
Theoretical Minimum	N/A	1	117	78.79	59	33.82	N/A	N/A	N/A
Memory-Based	N/A	1	164	84.58	67	33.82	N/A	N/A	N/A
Replication-Based, Persistent									
Multicopy	1st	1	187	86.21	71	37.14	8.92	10.54	12.65
	2nd	1	122	78.83	63	33.90	0.21	0.11	12.89
	3rd	1	117	78.79	59	33.82	0	0	12.89
Single-copy	1st	1	187	86.21	71	37.14	8.92	10.54	12.65
	2nd	1	199	88.70	69	36.09	6.29	12.45	0
	3rd	1	155	82.45	68	35.99	6.01	5.75	0

Table 3. Measurements obtained by memory-based and hybrid-persistent techniques for Presentation 1 in a system with 10 and 12 drives. Both techniques yielded identical measurements.

Drives	Latency	Resource Requirements						
		Memory		Bandwidth		Percent Overhead		
		Max	Avg	Max	Avg	Disk Band.	Mem	D. Space
10	1	129	79.55	65	33.82	0	0	0
12	1	125	79.02	65	33.82	0	0	0

display schedule consisted of 5,400 intervals, each 8/30 seconds long—the stream-based display of 8 frames assuming 30 frames per second. Our target system was configured with a 32-Kbyte disk page size. Its 32-Mbyte buffer pool was partitioned into 1,024 memory frames ($C = 1,024$). The number of

disks in the system varied from 1 to 16 ($1 \leq D \leq 16$). Each disk supported 68-Mbps transfer, 17-ms seek time, and 8.33-ms latency. Thus, a disk could read nine random pages during an interval ($B = 9$).

Results

In comparing how each technique (memory-based, replication-based, and hybrid with a memory threshold of 512 frames) supported display, we measured startup latency, memory requirement, disk bandwidth, and disk space requirements. Notice that the theoretical minimum and the memory-based results are independent of the amount of available memory (C). Indeed, these

Table 4. Measurements obtained for Presentation 1 using 14 disks.

Technique	Display	Latency	Resource Requirements						
			Memory		Bandwidth		Percent Overhead		
			Max	Avg	Max	Avg	D. Band	Mem	D. Space
Theoretical Minimum	N/A	1	117	78.79	59	33.82	N/A	N/A	N/A
Memory-Based	N/A	1	124	78.86	65	33.82	N/A	N/A	N/A
Replication-Based, Persistent									
Multicopy	1st	1	121	78.85	66	33.96	0.39	0.17	0.55
Single-copy	1st	1	121	78.85	66	33.96	0.39	0.17	0

Table 5. Measurements obtained for Presentation 1 in a system with 16 drives.

Technique	Display	Latency	Resource Requirements						
			Memory		Bandwidth		Percent Overhead		
			Max	Avg	Max	Avg	D. Band	Mem	D. Space
Theoretical Minimum	N/A	1	117	78.79	59	33.82	N/A	N/A	N/A
Memory-Based	N/A	1	123	78.82	62	33.82	N/A	N/A	N/A
Replication-Based, Persistent									
Multicopy	1st	1	121	78.82	62	33.89	0.18	0.08	0.26
	2nd	1	117	78.79	59	33.82	0	0	0.26
Single-copy	1st	1	121	78.82	62	33.89	0.18	0.08	0
	2nd	1	121	78.82	62	33.88	0.17	0.07	0
	3rd	1	121	78.82	62	33.88	0.17	0.07	0

two techniques dictate a value for C . The theoretical minimum defines a lower bound of memory required to support a coordinated display, and memory-based defines the minimum memory required to display a presentation without modifying data placement.

Our results also show the performance of the replication-based and the hybrid scheduling techniques in persistent mode when the presentation is displayed repeatedly. The measurements for transient mode correspond to those of the first displays in persistent mode. Both replication-based and hybrid are dependent on C . To schedule a replication, the system verifies that there is bandwidth available to read and write the page and memory available to stage the page between read and write operations. Therefore, assuming that disk space is not a limitation, if the minimum memory requirement of a coordinated display (theoretical minimum) is lower than the system's memory (C), the replication-based and hybrid resource schedules can support the display.

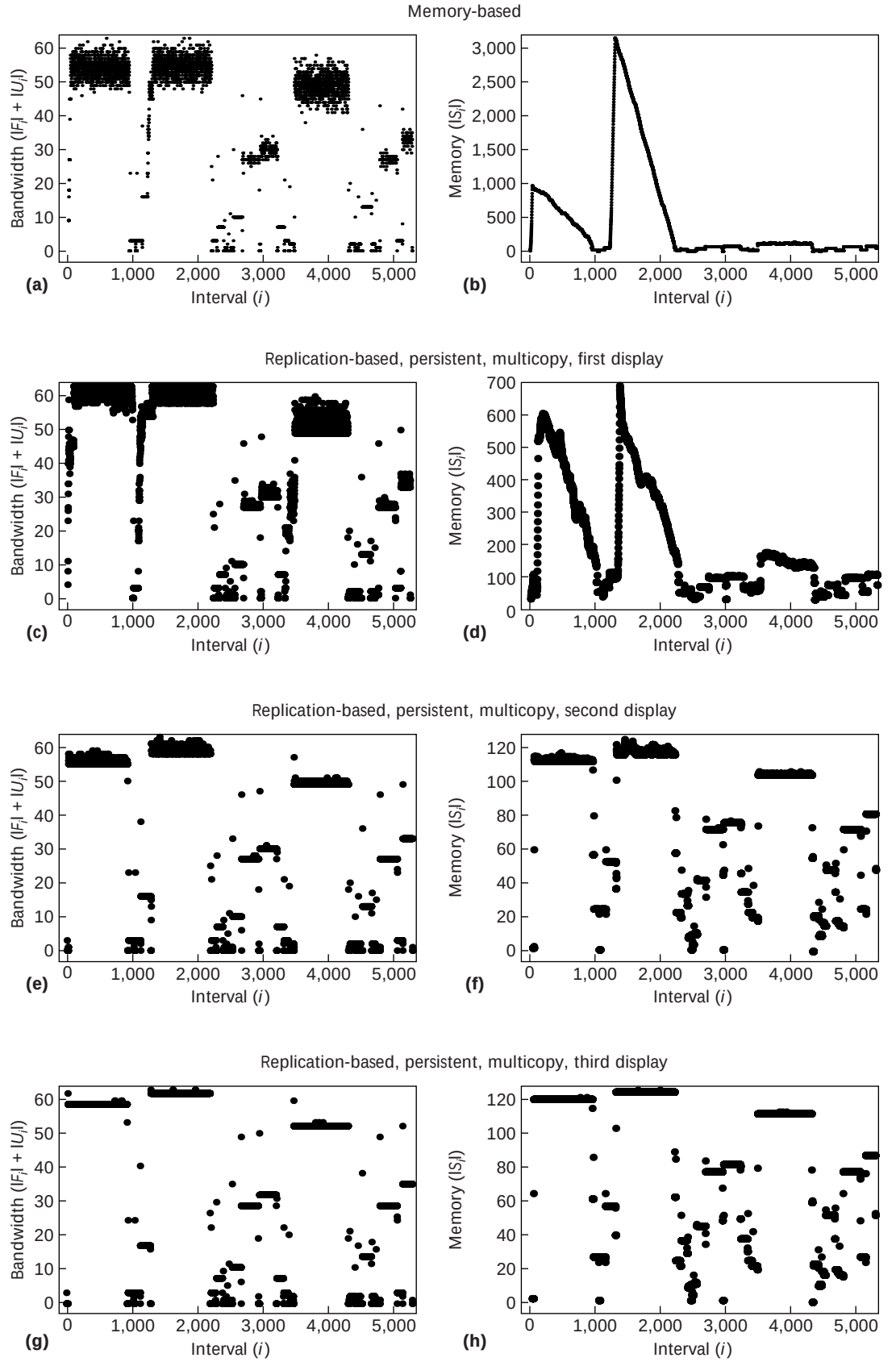
Tables 1, 2, 3, 4, and 5 quantify

1. the display's start-up latency (p), measured in time intervals,

2. the maximum ($\max\{|S_i| \mid -p < i \leq m\}$) and average ($(\sum_{i=-p+1}^m |S_i|)/(m+p)$) memory frames required by each resource schedule,
3. the maximum ($\max\{|F_i| + |U_i| \mid -p \leq i < m\}$) and average ($(\sum_{i=-p}^{m-1} |F_i| + |U_i|)/(m+p)$) disk pages each resource schedule reads or writes during a time interval,
4. the percentage of disk bandwidth used to construct secondary copies over the total disk bandwidth ($(\sum_{i=-p}^{m-1} |F_i| + |U_i|)$),
5. the percentage of memory used to contain pages during construction of their secondary copies over the total number of memory frames used ($(\sum_{i=-p+1}^m |S_i|)$), and
6. the percentage of disk space needed for the secondary copies over that for the primary copies (65,002).

We first present the results of our simulations assuming a system configuration with seven disk drives. We then present the results for the other configurations ($1 \leq D \leq 16$ and $D \neq 7$).

Figure 9. Resource requirements to display Presentation 1 using different heuristics in a system with seven disks.



Seven-disk system. Figure 9 profiles the disk bandwidth and memory requirements of alternative resource schedules to support the display of Presentation 1 for a system configured with seven disk drives. The profiles in the left column correspond to the disk bandwidth requirements (pages read or written during each interval) and the profiles on the right to the memory requirements (pages in memory at each instant). The profiles in the top row (Figures 9a and 9b) represent the memory-based resource schedule requirements, the next two (Figures 9c and 9d) those of the multicopy persistent replication-based schedule for the presentation's first display, the next two (Figures 9e and 9f) those of the multicopy persistent replication-based schedule for the presentation's second display, and the last two (Figures 9g and 9h) those of the multicopy persistent replication-based schedule for the presentation's third display.

As Figure 9b indicates, the memory-based technique might have excessive memory requirements (3,155 memory frames) to support a coordinated display. Unbalanced placement of pages referenced by a presentation during interval i increases the memory requirement during previous intervals j ($j < i$). Similarly, unbalanced placement during $i - 1$ increases the memory requirement during previous intervals j ($j < i - 1$). These increases might be cumulative if consecutive intervals result in bottlenecks.

The start-up latency for both replication-based and hybrid techniques depends on the intervals when the bottlenecks occur. Presentations whose reference pattern results in bottlenecks at the beginning of the display might incur a higher latency than those whose schedule results in a bottleneck at the end. Presentation 2 has a low disk bandwidth requirement at the beginning of its display. This enables the system to replicate pages during display, resulting in a lower start-up latency (two intervals for Presentation 2 and 94 intervals for Presentation 1 when displayed the first time with replication-based scheduling).

In a system that allows multiple page copies, the replication-based technique in persistent mode approximates the theoretical minimum as a presentation displays repeatedly. After three consecutive displays of Presentation 1, this technique's profiles were almost identical to those of the theoretical minimum, and the resulting replicated data required an additional 23.77 percent of the disk space (see Table 1). This is because the secondary copies eliminated some bottlenecks in

subsequent displays, reducing the display's memory and disk bandwidth requirements along with its incurred start-up latency. Note that the maximum amount of memory required for the first display of Presentation 1 (703 frames) decreased by a factor of six for the second display of the presentation.

With the single-copy replication-based technique in persistent mode, the disk bandwidth overhead also decreases as a function of the number of displays. This technique maintains the page copy most frequently referenced by the display and deletes the others. This change in data placement might decrease the number of bottlenecks in subsequent displays, reducing the number of page replications from 15,348 for the first display to 10,637 for the second display and 9,429 for the eighth display (reflected as a decrease in the percentage of disk bandwidth overhead; see Table 1). Replications might increase both memory requirements and memory overhead in subsequent displays, however. The data placement changes might cause bottlenecks at different times, forcing the replications to be scheduled during a period of scarce resources.

Moreover, the time a page remains memory resident prior to being written might increase because this heuristic strives to minimize start-up latency. This increases the schedule's memory requirements. Table 1 shows the memory requirement growth from 703 in the first display to 1,024 in the second display.

The hybrid technique requires less disk bandwidth (columns Bandwidth, Avg and Percent Overhead, D. Band in Table 1) and disk space (Percent Overhead, D. Space) than the replication-based technique. Instead of solving all bottlenecks with replications that increase both disk bandwidth and space requirements, hybrid prefetches pages until it reaches a threshold (50 percent of C in our experiments). Replication requires 4.92 percent extra disk space for the first display of the hybrid technique and 22.66 percent for the first display of replication-based. Hybrid reaches a stable state sooner than replication-based. The second display using the hybrid technique constructs no additional secondary copies, while the third display using the replication-based technique continues to construct secondary copies.

Trends. With fewer than seven disks (Table 6, next page), none of the proposed techniques can support a coordinated display of Presentation 1 or 2 because there is insufficient disk bandwidth and

Table 6. Lower bounds of latency and memory requirements to display Presentation 1 on a system with less than seven disks.

D	Theoretical Minimum		Memory-Based	
	Latency	Memory	Latency	Memory
1	4,765	45,314	4,765	45,314
2	1,883	37,014	1,899	37,014
3	941	28,699	1,871	28,708
4	470	20,401	941	20,401
5	187	12,103	473	15,229
6	1	3,805	192	9,609

memory (the theoretical minimum memory requirement exceeds $C = 1,024$). We could increase the memory capacity (C) to exceed the theoretical minimum memory requirement (Table 6, third column). However, this might be insufficient to support the display with a memory-based resource schedule (for example, for $D = 6$, $C = 4,000 < 9,609$). In this case, the system has to use the replication-based or hybrid technique. Another alternative is to increase the number of disks in the system (to seven disks or more) so that the theoretical minimum memory requirement is lower than the memory capacity.

Besides supporting a coordinated display, a resource schedule might need a bounded latency. If the latency of the theoretical minimum (second column of Table 6) exceeds this, then we must add disks to satisfy the latency constraint.

To illustrate, suppose that an application imposes a limit of two minutes (450 intervals) on the latency of Presentation 1. The system then must have at least five disks; but a memory-based schedule would not satisfy the latency constraint on a system with five disks (latency = 473). We thus should use replication-based or hybrid schedules instead to change data placement and satisfy this constraint.

Twelve disks or more provide abundant bandwidth, rendering the differences between techniques negligible (Tables 4 and 5). Increasing the number of disks has the following effects:

1. It decreases the memory requirements of the memory-based technique (fifth column in Table 6; Memory columns in Tables 2, 3, 4, and 5).
2. It decreases the disk bandwidth requirements of the replication-based technique (Bandwidth and Percent Overhead, D. Band columns in Tables 2, 4, and 5).

3. It might decrease the memory requirements of the replication-based technique (Memory columns in Tables 2, 4, and 5).

Additional disks reduce the possibility of bottlenecks. For replication-based schedules, the memory requirement depends not only on the number of prefetches but also on replication scheduling. In reducing a display's start-up latency, our heuristic might increase its memory requirements.

Hybrid behaves the same as the memory-based resource schedule if the memory requirement is below the threshold. In our experiments, the memory requirement of the memory-based technique for a system with at least eight disks is lower than the threshold (512 memory frames). Then hybrid behaves like memory-based for a system with eight to 16 drives.

Summary

Our study shows that the replication-based technique supports coordinated displays that the memory-based technique cannot (because it requires excessive memory). Moreover, the replication-based technique in multicopy persistent mode approximates the theoretical minimum for required memory and latency as a function of the number of times a presentation is displayed. The memory-based technique, however, minimizes disk resource requirements. The memory threshold of the hybrid technique was lower than the memory requirement of memory-based for some system configurations, and higher for other configurations. If the threshold exceeds this memory requirement, then hybrid scheduling is identical to memory-based scheduling.

Hybrid uses memory and replications to resolve bottlenecks in a controlled fashion and thus allocates resources effectively. This technique compromises between memory and disk resources (bandwidth and space). It requires less disk resources than replication-based and less memory than memory-based. However, memory-based requires less disk resources than hybrid, while replication-based requires less memory and yields lower latency than hybrid after the presentation is displayed repeatedly in multicopy persistent mode. The threshold determines how much memory to trade for disk resources. As the threshold approaches zero, hybrid starts to approximate replication-based. (Similarly, as it approaches 100 percent, it approximates memory-based.) The threshold should therefore be tuned based on system resource availability.

Future research directions

In the future we intend to address the following:

1. integration of fault tolerance into the scheduling techniques,
2. development and evaluation of migration-based techniques,
3. resource scheduling techniques for multiuser environments where several users display different presentations, and
4. resource scheduling that supports the display of presentations with dynamically created display schedules (interactive video games, for example). MM

Acknowledgments

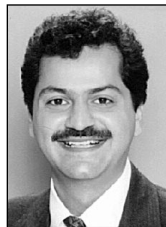
This research was supported in part by NSF grants IRI-9222926, IRI-9258362 (NYI award), and CDA-9216321, and a unrestricted cash/equipment gift from Hewlett-Packard.

References

1. M.L. Escobar-Molano, *Management of Resources to Support Coordinated Display of Structured Presentations*, doctoral dissertation, University of Southern California, Los Angeles, Calif., 1996.
2. S. Bernstein, "Techno-Artists Tooning Up," *Los Angeles Times*, Section F, Nov. 10, 1994.
3. M.L. Escobar-Molano, S. Ghandeharizadeh, and D. Ierardi, "An Optimal Resource Scheduler for Continuous Display of Structured Video Objects," *IEEE Trans. Knowledge and Data Engineering*, Vol. 8, No. 3, June 1996, pp. 508-511.
4. F.A. Tobagi et al., "Streaming RAID—A Disk Array Management System for Video Files," *ACM Conf. on Multimedia*, ACM Press, New York, Aug. 1993, pp. 393-400.
5. S. Berson et al., "Staggered Striping in Multimedia Information Systems," *Proc. ACM SIGMOD*, ACM Press, New York, May 1994, pp. 79-89.



Martha L. Escobar-Molano is an assistant professor in the Department of Computer Science and Engineering at the University of South Florida. She holds a BS from the University of the Andes (Colombia), an MS from the University of Massachusetts at Boston, and a PhD from the University of Southern California, all in computer science. Her research interests include multimedia information systems and databases.



Shahram Ghandeharizadeh is an associate professor of computer science at the University of Southern California. He received his PhD in computer science from the University of Wisconsin, Madison, in 1990. In 1992, he received the National Science Foundation Young Investigator's Award for his research on the physical design on parallel database systems, and in 1995 he received an award from the USC School of Engineering for his research and teaching activities. His primary research interests include design and implementation of multimedia storage managers, parallel database systems, and active databases.

Contact Escobar-Molano at the Computer Science and Engineering Department, University of South Florida, Tampa, FL 33620, e-mail mescobar@csee.usf.edu.