Fine-Grained Device Management in an Interactive Media Server

Raju Rangaswami, Zoran Dimitrijevic, Edward Chang, and Shueng-Han Gary Chan

***Abstract*—The use of interactive media has already gained considerable popularity. Interactivity gives viewers VCR controls like slow-motion, pause, fast-forward, and instant replay. However, traditional server-based or client-based approaches for supporting interactivity either consume too much network bandwidth or require large client buffering; and hence they are economically unattractive. In this paper, we propose the architecture and design of an interactive media proxy (IMP) server that transforms noninteractive broadcast or multicast streams into interactive ones for servicing a large number of end users. For IMP to work cost-effectively, it must carefully manage its storage devices, which are needed for caching voluminous media data. In this regard, we propose a fine-grained device management strategy consisting of three complementary components: disk profiler, data placement, and IO scheduler. Through quantitative analysis and experiments, we show that these fine-grained strategies considerably improve device throughput under various workload scenarios.**

***Index Terms*—Data placement, disk profiling, interactive, IO scheduling, media server, streaming, video-on-demand.**

I. Introduction

DUE TO THE proliferation of video content, it has become increasingly important to manage video data effectively to facilitate efficient retrieval. On-demand interactive video streaming for education or entertainment over the Internet or broadband networks is becoming popular [1]–[4]. In true *video-on-demand* (VoD), a video server allocates a dedicated stream to each user so that the user can freely interact with the video by means of VCR controls (such as pause, fast-forward, and instant replay). But such a system becomes expensive in both network and server bandwidth when tens of thousands of concurrent users have to be accommodated. A more scalable solution is to serve multiple requests for the same video with a single broadcast or a few multicast streams [5]–[7].

Due to the nature of broadcast and multicast, however, end users cannot interact with a TV program or a video using VCR-like controls. We propose an interactive media proxy (IMP) server architecture which acts as a dual client/server system to enable interactivity. As a client of broadcasters (or multicasters), the IMP system reduces network traffic to support interactivity. Additionally, it functions as a server by managing streams to enable interactivity for a large number of end users. With interactive capability, students in a virtual classroom or at a library can watch a live lecture at their own pace. A hotel can turn televised programs or movies into interactive ones in its guest rooms. A cable service provider can provide interactive video services to thousands of subscribers. We believe that the IMP architecture is attractive because it not only solves the scalability problem of the traditional server-based VoD model, but also provides a cost-effective solution to user interactivity.

The IMP architecture faces challenging design issues. In addition to the high bandwidth and realtime IO requirements that have been studied extensively [1], [8]–[11], it must deal with requests for writes as well as reads, and adapt to the changing ratio of write requests to read requests (defined as the *write–read ratio*). We explore this aspect further with three application scenarios:

1) *Local-area interactive system*. A local-area interactive system consists of a centralized bank of disk-array servers that provide service to clients in a local area. The local-area interactive system receives broadcast (or multicast) signals and enables interactivity for its clients. This model is clearly more economical than the server-based VoD model in which each viewer, interactive or not, requires a dedicated wide-area network channel. In addition, on the client side, no caching device is required. For a local-area interactive service, there are large numbers of interactive users and relatively few broadcast/multicast channels.
2) *@HOME interactive system*. The @HOME interactive system records broadcast programs on disk, and provides interactive service simultaneously to multiple display devices at home, in a classroom, or at a library. The @HOME service can be configured to provide interaction capability for certain popular channels in order to reduce cost. At the same time, it serves interactive content to a few users at home. Thus, the number of reads and writes are approximately equal and optimization for both is important.
3) *Video surveillance system*. Video surveillance systems are used in a wide variety of settings—shopping malls, casinos, airports, banks, etc. These systems can be made more effective if the surveillance team is provided with the ability to timeshift the surveillance streams. Typically, surveillance cameras are situated in numerous

Manuscript received October 29, 2001; revised June 26, 2002. This work was supported by the SONY/UC DiMI program and by NSF Grant EIA-0080134. The associate editor coordinating the review of this paper and approving it for publication was Dr. Anna Hac.

R. Rangaswami and Z. Dimitrijevic are with the Department of Computer Science, University of California, Santa Barbara, CA 93106 USA (e-mail: raju@cs.ucsb.edu; zoran@cs.ucsb.edu).

E. Chang is with the Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 USA (e-mail: echang@ece.ucsb.edu).

S.-H. G. Chan is with the Department of Computer Science, Hong Kong University of Science and Technology, Kowloon, Hong Kong (e-mail: gchan@cs.ust.hk).

Digital Object Identifier 10.1109/TMM.2003.814722

locations spanning the facility but are monitored by only a few people at a time. Thus, most of the streams serviced are recording (or write) requests.

In this paper, we propose the architecture and design of the IMP system. The IMP system employs various fine-grained device management strategies to address the traditional requirements of high-bandwidth IO and realtime operation, as well as varying write–read request distribution. Specifically, our fine-grained approach makes two key contributions.

1) *Device profiling*. We implement Scsibench [12] to collect detailed disk parameters, which are either not provided or often provided inaccurately by disk vendors. Obtaining an accurate and detailed disk profile enables IMP to manage a device more effectively (discussed next).
2) *Device management*. Using the information obtained from *device profiling*, we perform fine-grained device management to improve the overall disk access efficiency of the IMP system. Our device management scheme also enables the IMP system to adapt to different write–read request distributions through *data organization* and thus cater to a wide range of application scenarios. Our analytical study and experimental results suggest that such a strategy can offer a significant performance improvement over traditional systems.

The rest of the paper is organized as follows. In Section II, we describe our device profiler which performs feature extraction to obtain various device parameters. Section III describes our device management strategies. Section IV provides quantitative analysis. In Section V, we present a case-study using three sample configurations of our IMP system and evaluate our fine-grained device manager. In Section VI, we present related research, and we make concluding remarks in Section VII.

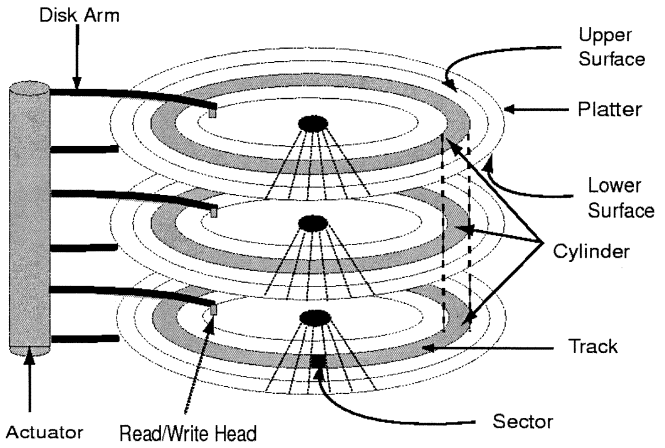


Fig. 1. Disk-drive internals.

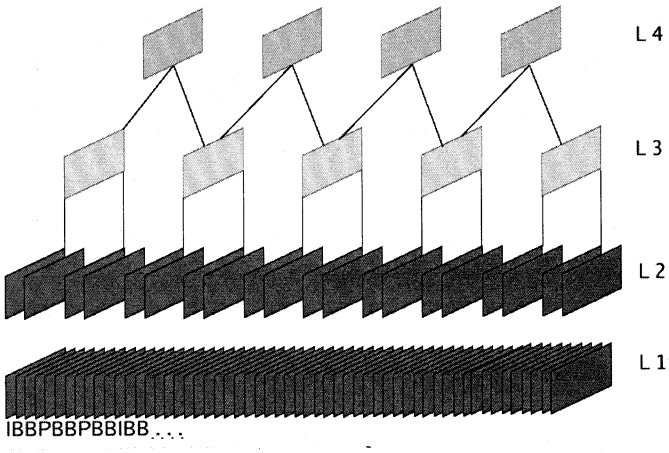


Fig. 2. The Truncated Binary Tree (TBT) formation.

II. Device Profiling

In this section, we present a SCSI disk profiling tool [12] that extracts detailed disk parameters. This parameter profile is used in performing fine-grained device management (Section III) in the IMP system.

Fig. 1 depicts the general disk architecture. The main components of a current-day disk drive are as follows: one or more *disk platters* or recording surfaces rotating on a shared *spindle*, a set of *read–write heads* residing on a shared *disk arm*, *disk logic*, *cache memory* with embedded caching and replacement logic, and the *IO bus*. Data is stored on the hard disk by logically organizing it into disk *blocks* (the unit of disk access). Typically, a block corresponds to a disk *sector*. The set of sectors that are on the same magnetic surface and at the same distance from the central spindle form a *track*. The set of tracks at the same distance from the spindle form a *cylinder*. To perform a data-transfer, the disk arm must first move from its current position to the destination cylinder. It must then wait for the desired sector to rotate and reach the disk head before data-transfer can commence. These constitute the disk seek and rotational delays respectively. In Table I, we enumerate disk parameters along with other parameters that are introduced later.

Traditional device management schemes often neglect some of the above disk parameters. Disk abstractions such as SCSI or IDE interfaces hide low-level device characteristics from the operating system and virtualize the access to the device in the form of logical blocks. Such device abstraction makes the task of tuning disk operation to match application requirements (and

TABLE I
IMP System Parameters

Parameter	*Description*
$\mathcal{N}_w$	*Number of broadcast channels (write streams)*
$\mathcal{N}_r$	*Number of interactive requests (read streams)*
ρ	*Write-read ratio*
f	*Fraction of interactive streams (i.e., fast-scans)*
$\mathcal{TR}$	*Minimum disk data-transfer rate*
$\gamma(d)$	*Worst-case latency function to seek d cylinders*
$\mathcal{H}$	*Head-switch time*
$\mathcal{DR}$	*Average display rate of MPEG stream*
$\mathcal{DR}_f$	*Average display rate of a fast-scan stream*
C	*Throughput of the disk (number of requests serviced)*
$\mathcal{T}$	*IO cycle time*
$\mathcal{T}_s$	*Reservation time-slot*
$\mathcal{M}$	*Available system memory*
Z_l	*Logical zone set*
α , β	*Adaptive tree parameters*
κ	*Number of exclusive I-frame sub-streams*
h	*Height of the adaptive tree*
η	*Density of the adaptive tree*
Φ	*IO resolution*

thus improve IO efficiency) difficult. Disk profiling is then necessary in order to overcome the following three shortcomings of traditional schemes.

1) *Inaccurate Information*. Some schemes can be inefficient because they make worst-case (or average-case) assumptions about data-transfer rates and seek-and-rotational delays.
2) *Dynamic Information*: Some are ineffective because they are not aware of internal changes occurring within the device. For instance, a sequentially placed file can be subsequently broken into nonsequential segments due to sector relocations.
3) *Manufacturing Variance*. We observe that even two disks of the same model from the same vendor may exhibit different performance characteristics, and some data (such as the locations of bad sectors) simply cannot be predetermined.

A. Disk Features

Let us now look at disk parameters which enable us to overcome the shortcomings of traditional device management schemes. In addition to disk seek times and rotational delay, we are also interested in the following disk parameters:

- Per-zone data-transfer rate. On a modern disk, the amount of data that can be stored on a track is proportional to the length of the track. Outer tracks can store more data and hence have higher data-transfer rates. A modern disk is partitioned into zones, each comprising multiple cylinders. The zones have different numbers of sectors per track and hence different data-transfer rates. We extract per-zone data-transfer rates to perform bit-rate matching for video streams (Section III).
- Spare/bad sector locations. A disk allocates information about spare sectors for relocating data when a good sector is damaged. A file that was created as a sequential one can be subsequently broken into nonsequential blocks by sector re-mapping. We collect spare/bad sector locations to determine if a logically sequential block of data is indeed sequentially placed on disk.
- Head switch and cylinder switch times. While accessing a block residing on the next track or the next cylinder, the disk arm needs to be repositioned. The head switch and cylinder switch times must be accurately accounted for as a part of the IO cost.

B. Feature Extraction

Our feature extraction tool runs on a Linux system in the user mode, using the SCSI generic interface to the SCSI bus. The SCSI library was implemented from scratch, complying to the SCSI-2 standard [13]. We then developed methods to extract a range of disk features that are of interest. We used some of the strategies presented in similar studies [14]–[16]. All the timing measurements are performed using an Intel Pentium time-stamp-counter register, which provides high precision with the overhead of only one register-read instruction.

We collect disk features in several steps. The first step is to measure the rotational time of the disk. The second step is to obtain detailed logical-to-physical (cylinder, track, sector) block mapping. The output of this operation is the zoning information, together with a list of all tracks that contain a large number of spare or bad sectors. The zoning information of the disk (Seagate ST39102LW) we tested is presented in Table II. T_{size} denotes the number of tracks in a given zone, and $\mathcal{TR}$ denotes the transfer rate in MBps. The disk has 12 tracks per cylinder. $\mathcal{TR}_{max}$ denotes the maximum transfer rate in MBps of each zone calculated using rotational time and track size for that zone. The difference between the transfer rates of outer and inner zones can be substantial. Also, the transfer rate for long sequential reads is lesser than the maximum throughput due to track and cylinder skews. During the course of our experiments, we learned that zoning characteristics can be slightly different even between physical devices of the same model. The actual mappings on disks differ depending on the location of errors on magnetic surfaces detected during the low-level format. Hence, a device manager should extract precise zoning information, and it should periodically validate the logical-to-physical mappings in order to guarantee peak disk performance.[1]

TABLE II
SEAGATE ST39102LW DISK ZONE FEATURES

Zone	*Cylinders*	T_{size}	$\mathcal{TR}$	$\mathcal{TR}_{max}$	$\gamma(1)$	$\mathcal{H}$
1	0-847	254	18.56	21.77	1104	883
2	848-1644	245	18.02	21.00	1108	885
3	1645-2393	238	17.49	20.40	1108	876
4	2394-3097	227	16.70	19.46	1115	890
5	3098-3758	217	15.99	18.60	1115	890
6	3759-4380	209	15.70	17.92	1105	884
7	4381-4965	201	14.83	17.23	1099	875
8	4966-5515	189	13.98	16.20	1124	901
9	5516-6031	181	13.39	15.51	1124	903
10	6032-6517	174	12.89	14.92	1109	886
11	6518-6960	167	12.38	14.32	1119	887

Our tool also measures head switch and cylinder switch times. Table II lists estimated maximum values for head switch time $(\mathcal{H})$ and cylinder switch time $\gamma(1)$ (in microseconds). Finally, we collect information on disk latency, including seek time and rotational delay. Interested readers may refer to [17] which presents a more extensive study of disk profiling. The parameters collected by the disk profiler can assist data placement and IO scheduling, which we discuss next.

III. DEVICE MANAGEMENT

With precise disk information extracted from the disk, data can be placed more intelligently, and IO can be scheduled more efficiently for improving disk throughput. In this section, we describe how IMP takes advantage of an accurate disk profile to perform fine-grained device management for improving system performance. The design of IMP consists of three complimentary device management strategies: *high-level data organization*, *low-level disk placement*, and *IO scheduling*. These com-

[1]If a sector becomes bad and its data is relocated to a spare sector, a logically sequential file is no longer physically contiguous on the disk. The device manager may not be able to relocate the entire file to make it contiguous, but it can be aware of the changed conditions and reserve more time for performing IO.

ponents of IMP work together to improve disk throughput by minimizing both intra-stream and inter-stream seek delay, and by improving the effective data transfer rate.

In this section, we describe each component of IMP in detail. First, we describe the *adaptive tree scheme*, a high-level data organization scheme for reducing intra-stream disk latency and supporting interactive operations efficiently. This organization scheme forms the basis of our low-level disk placement policy. Next, we present *zoning placement* and *cylinder placement*, which are two complimentary low-level disk placement strategies for placing stream data on disk. These schemes improve effective data transfer rate and reduce the inter-stream disk latency. Finally, we present *step-sweep*, an IO scheduling policy, which takes advantage of the above two components and improves the overall disk throughput.

A. High-Level Data Organization

Without loss of generality, we can assume that an MPEG stream[2] consists of m frame-sequences; each sequence has δ frames on average, led by an I frame and followed by a number of P and B frames.

To support a K-times speed-up fast-scan, the system displays one out of every K frames. Allowing K to be any positive integer, however, can result in the system requiring high IO and network bandwidth, as well as incurring significant memory and CPU overhead. This is because a frame that is to be displayed (e.g., a B frame) may depend on other frames that are to be skipped (e.g., an I and a P frame). The client/server dual system ends up having to read, stage (in main memory), and transmit to a client many more frames than the client displays causing poor IO resolution. (*IO resolution* is the ratio of the useful data read to the total data read from the hard disk.) We intend to remedy this poor IO resolution problem with a high-level data organization scheme, the *adaptive tree* scheme.

To avoid processing the frames that are to be skipped, we do not include any B frames in a fast-scan, and the playback system displays a P frame only if the corresponding I frame is also included in the fast-scan. This restriction will not allow a user to request a fast-scan of any speed-up. Since VCR or DVD players support only three to five fast-scan speeds, we support only a few selected speeds of fast-scans.

We now describe the adaptive tree data organization scheme in detail. In the adaptive tree approach for high-level MPEG data organization, we use a basic truncated binary tree structure to store MPEG frames. To provide good IO resolution for all playback speeds, we organize MPEG frames in a truncated binary tree structure, as shown in Fig. 2. The levels (Li) of the adaptive tree can be described as follows.

- *Level 1 (the leaf level)*. The original MPEG stream.
- *Level 2*. All I and P frames stored in their playback sequence.
- *Level 3 to* $(\kappa+2)$. Containing only sampled I-frames. κ is the number of sub-streams containing only I-frames. The higher the level, the lower the sampling rate.

[2]The Advanced Television System Committee (ATSC) has adopted MPEG2 as the encoding standard of DTV and HDTV.

Each level of the tree forms a substream of the original video stream and is stored as a sequential file on the disk. The placement of the different tree levels on the disk will be addressed in the low-level placement (Section III-B). To service a request, only one level of the tree is read from the disk with good IO resolution.

Let S_i denote the set of frames that belong to the i^{th} level. An example of a five-level organization is

$$S_1 = \{I_1, B_1, B_2, P_1, B_3, B_4, P_2, B_5, B_6, I_2 \cdots\}$$
$$S_2 = \{I_1, P_1, P_2, I_2, P_3, P_4, I_3 \cdots\}$$
$$S_3 = \{I_1, I_2, I_3, I_4 \cdots\}$$
$$S_4 = \{I_3, I_6, I_9, I_{12} \cdots\}$$
$$S_5 = \{I_6, I_{12}, I_{18}, I_{24} \cdots\}.$$

When a nine-times fast-scan is requested, we can read in S_3 to achieve perfect resolution. (We assume that an I-frame leads a nine-frame sequence.) When a 54-times fast-scan is requested, we read in S_5 to achieve perfect IO resolution.

This adaptive tree approach trades storage space for IO efficiency. The precise tradeoff can be controlled by fine-tuning the following two parameters.

- *Height* (h). The height parameter describes the number of levels (or files) in which the data is organized. Height is a static parameter and it controls the number of fast-scan streams supported. It can be expressed as $h = \kappa + 2$.
- *Density* (η). Density is a tunable system parameter whose value can range from zero to one. A smaller η value eliminates some tree levels and decreases the tree density. For example, at $\eta = 1/3$, only one out of every three levels of the tree (from the leaf level up) are retained. The higher the density, the higher the IO resolution (favorable) and storage cost (not favorable).

Another factor that affects the disk bandwidth requirement of a fast-scan request is the playback frame rate. A typical DVD player plays a fast-scan stream at 3 to 8 fps (frames per second), instead of 24 to 30 fps, the regular playback speed. A fast-scan stream needs to be displayed at a lower rate so that the viewer can comprehend the content and react in time. Due to its high IO resolution and lesser frame rate, a fast-scan stream under the adaptive tree scheme typically requires lesser disk bandwidth than that of a regular playback stream.

B. Low-Level Disk Placement

Now, we describe how we physically place each stream S_i on disk. Here, our goals are to maximize the effective data transfer rate and minimize disk latency. Our low-level data placement scheme achieves these goals by employing two independent solutions. We utilize *zoning* information to perform *zoning placement* to achieve higher transfer rates, and we perform *cylinder placement* in order to minimize disk latency.

1) Zoning Placement: In the previous section, we described the adaptive tree scheme for MPEG data organization, which splits a stream into high and low bandwidth sub-streams. Higher bandwidth streams require higher data transfer rates to reduce overall data transfer time. From our experiments in disk feature

extraction in Section II, we realize that inter-zone bandwidth variations can be as great as 50% (see Table II). Zoning placement performs bit-rate matching of streams to zones.

In order to map streams to disk zones, we first create a set of logical disk zones, Z_l, from the physical zones of the disk. Let these logical zones be numbered from 1 *to* $|Z_l|$, from the outermost zone to the innermost. We map the physical zones of the disk to one of the $(|Z_l| = \lceil h \cdot \eta \rceil)$ *logical* zones, where the η and h are the density and height parameters of the adaptive tree. Each sub-stream is mapped to one of these logical zones in which it is stored.

First, we assume that all streams (and sub-streams) have equal popularity to be accessed. (We will relax this assumption shortly.) Using adaptive tree levels, we divide each video stream into $|Z_l|$ sub-streams of differing bit-rates as described earlier. We group the similar bit-rate sub-streams from different video streams and place them in the same stream group S_i to obtain $S_1, S_2, \cdots S_{|Z_l|}$ such stream groups. Then the objective function to place each stream group S_i in a unique logical zone Z_j, can be written as

$$\mathcal{O} = \min \sum_{i=1}^{|Z_l|} \frac{\mathcal{DR}_i}{\mathcal{TR}_j} \tag{1}$$

where $\mathcal{DR}_i$ denotes the common display rate of all substreams grouped in set S_i, and $\mathcal{TR}_j$ denotes the transfer rate of zone Z_j in which the sub-stream group S_i is placed. However, since all streams do not enjoy the same popularity at all times, *is this the best objective function for optimal placement?*

If we can predict the future with high accuracy, we can possibly rewrite the objective function as

$$\mathcal{O} = \min \sum_{i=1}^{|Z_l|} \frac{\mathcal{DR}_i}{\mathcal{TR}_j} P(i) \tag{2}$$

where $P(i)$ denotes the access probability of the i^{th} substream set. Unfortunately, predicting $P(i)$ is difficult. For instance, a sports highlight enjoys only a burst of interest.

However, if we look at the problem from a different perspective, zoning placement can provide a bound for the worst-case IO cycle time. A lower worst-case bound is useful because it conserves memory space for staging the streams.

[Lemma 3.1] (Zoning Placement): Using $i = j$ in objective function (1) to place streams achieves the lowest worst-case bound for the total IO time for servicing any $\mathcal{N}$ requests.

The formal proof appears in [17]. Here, we illustrate the idea using an example.

[Example 3.2] (Zoning Placement): Suppose an MPEG file is organized into three sub-streams, and their bit-rates are

- S_1 (regular speed stream): 6 Mbps;
- S_2 (ten times fast-scan stream): 4 Mbps;
- S_3 (30 times fast-scan stream): 3 Mbps.

Suppose a disk has three zones (Z_1, Z_2, and Z_3), and their transfer rates are 150, 100, and 75 Mbps, respectively. Suppose each zone can store only one stream, and the system services three requests in one IO cycle. Placing S_1 in Z_1, S_2 in Z_2, and S_3 in Z_3 gives the system the lowest worst-case data transfer time. To sustain one second of playback for each stream, the total worst-case data transfer time is $3 \times (6/150) = 120$ ms. It is easy to see that if we place S_1 in either Z_2 or Z_3, the total worst-case data transfer time increases.

Although we introduce zoning in the context of storing MPEG video data, we can see that the above principles can be applied to general purpose multimedia file servers that support multiple data types such as images, audio, and video as well as traditional non realtime data. For instance, a text file can be stored in a low bandwidth disk zone, a 256 Kbps *mp3* file can be stored in a medium bandwidth zone, and a 19.2 Mbps HDTV stream can be stored in the high bandwidth zone to maximize disk throughput.

2) Cylinder Placement: IMP is characterized by N_w broadcast streams which need to be stored on disk and N_r interactive user streams. Careful stream placement on disk tracks can minimize seek and rotational overheads. Using zoning placement, we combine similar bit-rate streams in the same logical zone. Cylinder placement describes how to place data for different streams which share a zone.

In the IMP system, write streams are deterministic in terms of start time, duration, and average data-rate. Read streams depend on user interactions and are inherently nondeterministic and unpredictable. The key idea of the *cylinder placement* strategy is to exploit the deterministic nature of write streams and use a best-effort approach for reads. For each stream, we allocate a group of adjacent cylinders of size c on the disk. Each consecutive write stream is allocated the next c cylinders on disk adjacent to the c allocated cylinders for the previous stream. When any write stream uses up its allocated c cylinders, a new set of $c \cdot \mathcal{N}_w$ free cylinders within the same zone and adjacent to the previous cylinder set is allocated. The write streams are stored in the newly allocated cylinders starting from the next IO cycle. Cylinder placement maintains the same relative cylinder distance between the stream pairs so that the scheduling order can be preserved across IO cycles. This minimizes *IO variability*. Minimizing IO variability is crucial for minimizing memory requirements [1]. Cylinder placement might lead to some fragmentation of disk space. However, we observe that for high bandwidth applications, the disk is bandwidth-bound rather than storage-bound. Hence, some storage can be sacrificed for the sake of improving disk throughput. Placing write streams in adjacent cylinder groups has the following advantages.

1) The seek overhead for switching from one write stream to the next write stream requires the disk to seek c cylinders, typically a number less than 50. From our experiments in calculating disk seek time, we note that this overhead is almost equal to the minimum seek time for a single cylinder.
2) The strategy of reserving cylinder groups for individual streams greatly reduces the probability that a single read operation may be nonsequential.

C. IO Scheduling

For a system to perform optimally under a given disk placement scheme, a closely coupled scheduling algorithm should be

in place. Our IO scheduling algorithm, *step-sweep*, is designed to work with the other two components of IMP.

Given a constant amount of memory and disk bandwidth, step-sweep is designed to 1) *maximize throughput* and 2) *minimize response time*. The step-sweep algorithm operates as shown in Fig. 3. The disk-arm services IO requests zone by zone, starting from the outermost zone. In each zone, it first services the write requests by sweeping in the direction toward the center of the disk. Read requests in the zone are then scheduled in a predetermined order which does not vary from one IO cycle to the next. The largest seek overhead in the case of read requests is restricted to the size (in tracks) of a single zone. The disk arm then moves to the adjacent inner zone and repeats this sequence. When the disk arm has serviced all the requests in the innermost zone, it gets repositioned to the outermost zone of the disk. This completes one cycle of IO requests. This cycle is repeated over and over again. A formal algorithm for the step-sweep scheduling policy is given as follows.

Procedure: *Step-sweep*
- **Variables**:
  i: Logical zone variable
  Z_l: Set of logical disk zones
- **Execution**:
  1. Initialize $i = 1$
  2. Service write requests in zone Z_i by sweeping in the direction toward the center of the disk.
  3. Service read requests in zone Z_i. These read requests are serviced in a predetermined order to reduce IO variability.
  4. Set $i = (i+1)\%|Z_l|$
  5. Go to step 2

The design of step-sweep is based on the following considerations. First, we know that the write streams are deterministic in nature. The system has control on where it can place the write streams on disk. Using *Cylinder Placement*, the seek overheads for write streams can be minimized without suffering from IO variability at the expense of some fragmentation of disk space. Thus, step-sweep schedules write streams optimally. Read streams, on the other hand, are unpredictable. Using simple sweep scheduling for read streams might result in large IO variability. Using step-sweep, the service order for read streams in a zone is predetermined to minimize IO variability [1], [11]. Step-sweep achieves its design objectives in the following manner.

1) To maximize throughput, we need to minimize the IO cycle time as well as the memory use per stream. We define *IO cycle time* as the time required to complete a single round of IO for each stream serviced by the system. Reducing IO cycle time allows the system to service more users in a given period of time. Step-sweep reduces IO cycle time by minimizing seek overhead. When seek overhead is reduced, the system needs to prefetch less data to fulfill the realtime playback requirement for each stream, and this reduces data transfer time. Step-sweep minimizes memory use per stream by minimizing IO variability.

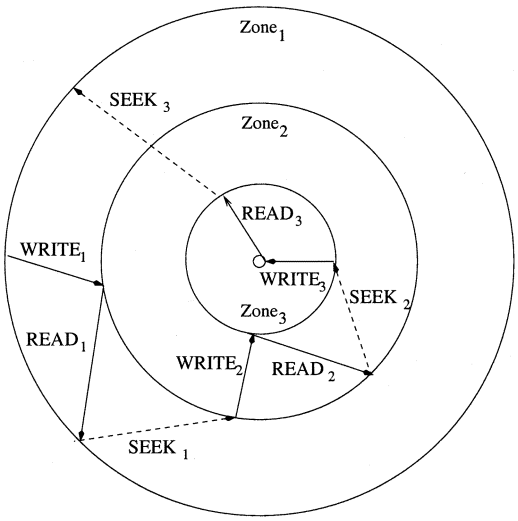


Fig. 3. Step-sweep IO scheduling.

2) To minimize response time, we use a reservation timeslot, T_s, within each IO cycle time T. We use this reservation slot to schedule "unexpected" new requests with minimum delay. These unexpected requests might also include requests for non-realtime data. Each new request can be serviced as soon as the current nonpreemptible IO operation is finished. Thus, the reservation time slot T_s can be used up in small chunks throughout the IO time cycle.

IV. Quantitative Analysis

In this section, we present results obtained from our quantitative analysis. Our quantitative model is developed assuming cylinder placement and step-sweep IO scheduling.

Let the IO cycle time be denoted by $\mathcal{T}$. This is the time required to complete a single round of IO for each stream serviced by the system. This IO cycle is repeated over and over again in the system. Let the disk support $\mathcal{N}_w$ broadcast channels, which perform simultaneous writes, and $\mathcal{N}_r$ timeshift streams, which perform reads. Suppose a fraction f of the timeshift streams are fast-scans and the ratio of write to read requests is ρ. Let C be the maximum throughput of the disk. Also, let $\mathcal{T}_s$ denote the worst case reservation slot time

If $\mathcal{T}_{Z_i}$ is the total time required to service all requests in zone i, and $\mathcal{T}_s$ is the reservation time slot, we can express $\mathcal{T}$, the cycle time, as

$$\mathcal{T} = \sum_{i=1}^{p} \mathcal{T}_{Z_i} + \mathcal{T}_s \tag{3}$$

where $p = |Z_l|$ is the number of logical zones used. In each zone, the scheduler first schedules the write requests, then the read requests. The disk arm is then moved to the beginning of the next zone by seeking to it. If the worst-case zone seek requires $\mathcal{T}_{zone}$ time, then

$$\mathcal{T}_{Z_i} = \mathcal{T}_{W_i} + \mathcal{T}_{R_i} + \mathcal{T}_{zone}. \tag{4}$$

Now, we proceed to quantify the write and read times in each logical zone. In each logical zone, Z_i, the disk performs $\mathcal{N}_w$

write requests. Thus, the time required to complete write operations in zone Z_i is

$$\mathcal{T}_{W_i} = \mathcal{N}_w \cdot \left[\gamma(d_w) + \frac{\mathcal{T} \times \mathcal{DR}(x)}{\mathcal{TR}_i}\right] \quad (5)$$

where d_w is the average seek distance for write operations and $\mathcal{TR}_i$ is the average transfer rate of logical zone i. $\mathcal{DR}(x) = \mathcal{DR}$ for writing regular playback streams and $\mathcal{DR}(x) = \mathcal{DR}_f$ for fast-scan streams. The total read time in the entire cycle is given by

$$\sum_{i=1}^{p} \mathcal{T}_{R_i} = \mathcal{N}_r \cdot \left[\gamma(d_r) + \mathcal{T} \cdot \left(\frac{(1-f)\cdot \mathcal{DR}}{\mathcal{TR}_1} + \frac{f \cdot \mathcal{DR}_f}{avg_{i=2}^{p}(\mathcal{TR}_i)}\right)\right]. \quad (6)$$

Substituting (4), (5), and (6) in (3), we can obtain a closed form solution for cycle time $\mathcal{T}$.

To fulfill realtime data requirements, each stream, read or write, allocates two buffers. The size of one buffer must be large enough to sustain the playback before another buffer is replenished. The memory requirement for a fast-scan stream is given by $2 \cdot \mathcal{T} \cdot \mathcal{DR}_f$ whereas that for write streams and regular-speed read streams is given by $2 \cdot \mathcal{T} \cdot \mathcal{DR}$. The total memory usage cannot exceed the available memory $\mathcal{M}$. We can thus quantify the memory requirement as

$$\mathcal{M} \geq 2 \cdot \mathcal{T} \cdot \mathcal{DR} \times \left[\mathcal{N}_w + (1-f)\cdot \mathcal{N}_r + f \cdot \mathcal{N}_r \cdot \frac{\mathcal{DR}_f}{\mathcal{DR}}\right]. \quad (7)$$

Given $\mathcal{M}$, $\mathcal{DR}$, $\mathcal{DR}_f$, $\mathcal{TR}$, $\gamma(d)$, f, Φ, and the write–read request ratio, ρ, we can use (3) and (7) to estimate the throughput of the disk.

V. System Evaluation

In this section, we evaluate the performance of the IMP system. First, we define the performance metric. Given a system with fixed resources, we would like to maximize the number of video streams that can be supported simultaneously. Hence we measure the system performance in terms of the number of streams supported by the disk. We refer to this performance metric as *disk throughput*. We performed the following experiments to evaluate the disk throughput.

System Configuration Evaluation. We performed a case study on three sample configurations of the *adaptive tree* data organization scheme (Section III). We studied the effect of the following three system parameters on the throughput ($\mathcal{N}$):

- available system memory ($\mathcal{M}$);
- write–read ratio (ρ);
- fraction of interactive streams (f).

Keeping two of the above parameters fixed, we examined the effect of changing the third.

Data Management Strategy Evaluation. We examined the individual as well cumulative effect of the following fine-grained device management strategies on the throughput ($\mathcal{N}$):

- zoning placement;
- cylinder placement;
- step-sweep IO scheduling.

For experimental evaluation of the IMP system, we used disk trace support provided by our Scsibench tool (see Section II). Scsibench supports the execution of disk traces using primitive disk commands like *seek*, *write*, and *read*. It also provides for accurate timing measurement. Using Scsibench, we performed trace executions on the Seagate ST39102LW disk presented in Table II. In some cases, where trace executions were not possible, we evaluate the system using analytical estimation.

A. System Configuration Evaluation

In this section, we report a case study of three sample configurations of the *adaptive tree* data-organization scheme (Section III), each of which is designed to optimize on a subset of the design parameters so as to perform optimally for a specific class of target applications, introduced in Section I.

1) *Truncated Binary Tree or TBT* ($\eta = 1$). This is the normal configuration in which all levels of the truncated binary tree are stored. The advantage of this scheme is that we have prepared streams for supporting each fast-scan speed and hence can achieve 100% IO resolution and minimum disk latency at the same time. However, due to replication of data, storage cost increases. A *local-area interactive service* supporting a large number of clients could use this configuration.
2) *Partial TBT or PTBT* ($\eta = 0.5$). In the PTBT configuration, the tree is partially dense. With $\eta = 0.5$, we store only alternate levels of the tree. Thus, in this configuration, there are prepared streams only for some of the fast-scan speeds. A fast-scan stream that is not directly accessible is created by selectively sampling the frames in another fast-scan stream, which serves a lower speed. Such streams suffer from a degraded IO resolution. The number of files to be written into (i.e., the number of seeks by the disk-arm) for each stream is $\lceil \eta \cdot h \rceil$. This scheme serves as a middle-ground between the SEQ and TBT configurations and could be used by *@HOME* type applications.
3) *Sequential or SEQ* ($\eta = (1/h)$). In the sequential configuration, only one out of h levels is stored to obtain a tree with density $(1/h)$. The sequential configuration stores only Level 1 of the tree. Higher levels of the tree are simply not stored. The goal of this scheme is to reduce seek overhead for writes, thus conserving memory use in write-intensive applications. However, this scheme suffers from very poor IO resolution for fast-scans. This scheme would be practical for a *video surveillance* type application.

Thus, each scheme aims to optimize a different subset of the design parameters. In Tables III and IV, we summarize each configuration's pros (with positive signs) and cons (with negative signs). IOR represents the IO resolution.

1) Available System Memory: In this section, we compare the memory requirement for each of the three sample configurations of the adaptive tree data-organization scheme using disk traces. These disk traces were generated so as to mimic the IO load of a media server. The trace executions were performed using the data management strategies of the IMP system.

TABLE III
SCHEME SUMMARY FOR READ OPERATIONS

Scheme	*Seek Overhead*	*IOR*	*Storage*
Sequential	++	– –	*N/A*
TBT	++	++	*N/A*
PTBT	++	+	*N/A*

TABLE IV
SCHEME SUMMARY FOR WRITE OPERATIONS

Scheme	*Seek Overhead*	*IOR*	*Storage*
Sequential	++	*N/A*	++
TBT	– –	*N/A*	– –
PTBT	–	*N/A*	–

Assuming a given amount of main memory, first we calculated the maximum IO cycle time expendable to support N users. Next, we obtained the actual IO cycle time from disk runs. If the disk runs were shorter than the analytically computed IO time cycle, we could conclude that the disk can support N users. If not, we would repeat the above experiment with $(N-1)$ users. We continued this iterative process till we obtain a feasible N, so that all the user streams are hiccup-free. We repeat the same experiment by assuming different memory sizes. For the following traces conducted using the Scsibench tool, we assumed that the fraction of user requests that are for fast-scan streams is 0.2. We assumed that the peak data consumption and input rates are 6.4 Mbps (the Standard DTV broadcast rate). Further, we assumed that the frame-rate for fast-scans was 5 fps (please refer to Section III for why we chose a lower frame-rate for fast-scan streams). We will use the same numbers for all subsequent experiments unless others are specified explicitly.

Fig. 4(a)–(c) presents the disk throughput for the SEQ, PTBT, and TBT configurations, respectively, against varying memory size (M) and for different values of the write–read ratio ($\rho = 0.25,\ 0.5,1,2,4$). We note that for the SEQ scheme, we need as little as 32 MBytes of memory in order to maximize disk throughput. For the PTBT and TBT schemes the corresponding numbers are 128 MBytes and 64 MBytes respectively. Looking at these figures from a different perspective, a mere 32 MBytes of main memory is sufficient to drive the disk throughput to almost 90% of the maximum achievable value. This is the case because for such an amount of memory, the system can buffer enough data so that all disk accesses are in large chunks. When the disk is accessed in large chunks, disk latency is much less in comparison to the time spent in data transfer. At this point, the bottleneck is the raw data transfer rate of the hard drive, which can support only a fixed maximum number of high bandwidth streams.

2) Write–Read Ratio: In this section, find out the variation in disk throughput under different write–read ratios using disk traces. We also compare the IMP system against the traditional Unix-like file-system that tries to store file data sequentially on disk and present the improvement in performance. Also, we present analytical results for wider ranges of the write–read ratio.

Fig. 5(a) compares the relative performance of the IMP system in each of the sample configurations: SEQ, PTBT, and TBT. Since the SEQ scheme is optimized for a large number of writes, it achieves a high throughput for write-intensive loads. PTBT performs optimally in midranges and TBT performs well for read-intensive loads. For different values of ρ, different configurations achieve the highest throughput, thus defining a distinct *ideal* region.

Next, we compare performance improvement of the IMP system over the traditional approach to storing and retrieving data. A traditional operating system like Unix is optimized for sequential access. However, for an interactive video application, access to data is not always sequential. For instance, a fast-scan stream needs to access only key frames from the entire file. In addition to providing interactive capability, the IMP system provides "fine-grained" device management strategies for adapting to changing request workload. Fig. 5(b) shows that for different workload configurations, IMP either outperforms or equals the performance of a traditional file-system. Throughput gains can be as much as 100% using IMP.

Since trace driven executions restrict the available parameter space for the write–read ratio, we perform further evaluation of the three sample configurations by analytical estimation. We use parameters for the Seagate ST39102LW disk, presented in Table II, and seek-curves obtained using our disk profiler, to perform our analysis. In Fig. 6, the x-axis represents the write–read ratio, and the y-axis shows the disk throughput achievable by the configurations. We can see clearly that for different values of ρ, different configurations achieve the highest throughput, thus defining a distinct *ideal* region. This means that the density of the adaptive tree has to be configured dynamically for optimal performance.

B. Data Management Strategy Evaluation

In this section, we perform an evaluation of our three fine-grained data management strategies: *zoning placement*, *cylinder placement*, and *step-sweep IO scheduling*. We compare the IMP system to a traditional Unix-like file-system that uses *sweep* scheduling. Most modern operating systems use sweep algorithm for disk IO scheduling, in which the disk arm moves from the outermost cylinder to the innermost, servicing requests along the way. Then, we add our device management strategies, one at a time, to examine the marginal improvement in throughput due to each strategy. Finally, we present the cumulative effect of these disk management strategies, in increasing the throughput of the system.

In order to make a fair comparison, we give the traditional system the benefit of using the adaptive tree data organization scheme to achieve good IO resolution. In addition, for each of the following evaluations, we assume that the system is dynamically tuned to the "ideal" adaptive tree configuration under changing write–read ratios. In essence we try to capture the effect of fine-grained device management of the IMP system in the form of *zoning placement*, *cylinder placement*, and *step-sweep* IO scheduling on throughput improvement.

1) Zoning Placement: Fig. 7(a) compares the throughput of the baseline sweep with sweep using zoning placement. Zoning placement improves throughput for read-intensive loads by as much as 65%. For write-intensive loads, our adaptive tree

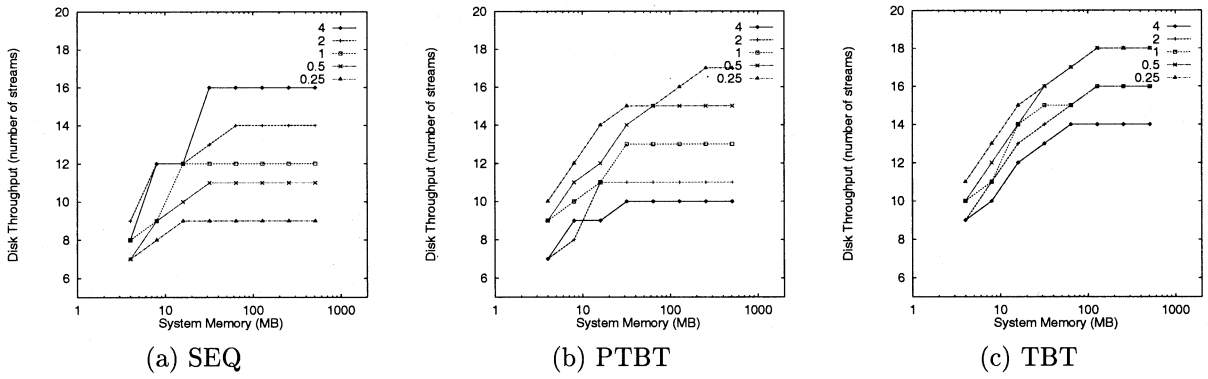


Fig. 4. Disk throughput for adaptive tree configurations.

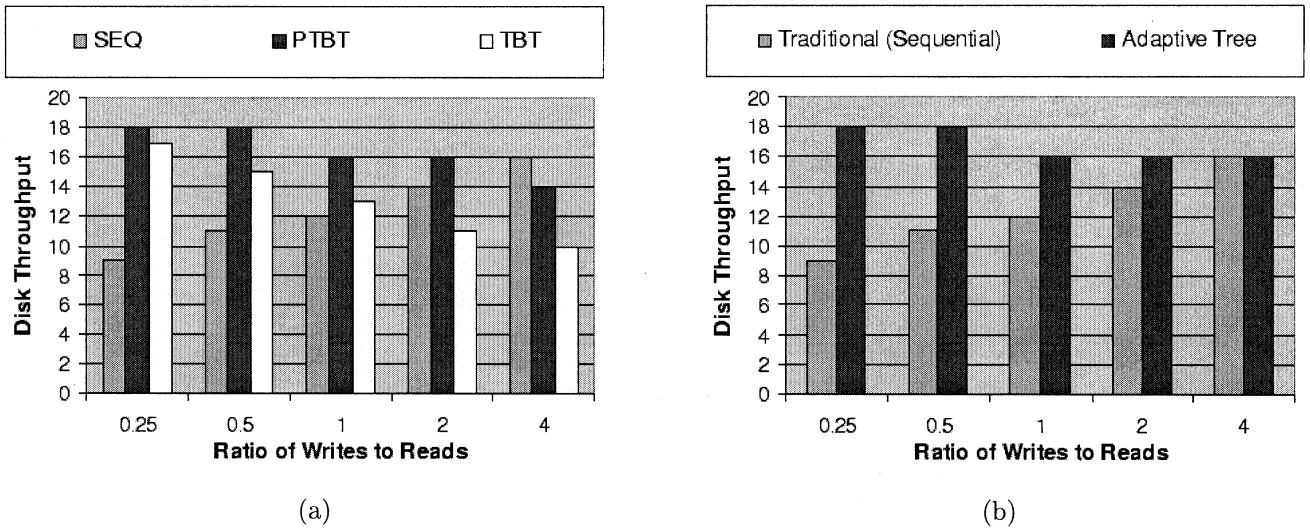


Fig. 5. Throughput comparison. (a) Different configurations of the adaptive tree scheme. (b) Improvement over traditional sequential placement of data.

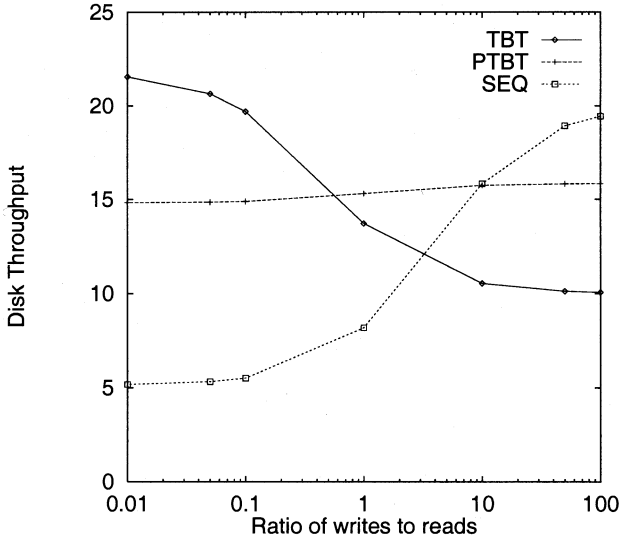


Fig. 6. Throughput for the three sample configurations.

scheme adopts the ideal sequential (SEQ) configuration and hence zoning has no effect.

2) Cylinder Placement: Fig. 7(b) compares the throughput of the baseline sweep with and without cylinder placement. Cylinder placement improves write performance by reducing the seek overhead for write operations. It increases the throughput by about 10% for write-intensive loads.

3) Step-Sweep IO Scheduling: Next, we examine the marginal effect of step-sweep algorithm in comparison with traditional sweep. Step-sweep increases throughput by decreasing memory use. It reduces memory use by minimizing IO variability and seek overhead. Fig. 7(c) shows that without step-sweep scheduling, there is a 5 to 10% degradation in throughput.

4) Cumulative Effect: Finally, Fig. 7(d) shows the cumulative effect of our fine-grained device management strategies. The IMP system offers a performance gain of as much as 75% over traditional systems. This highlights the importance of performing fine-grained storage management using the IMP system.

As regards response time, a detailed evaluation is beyond the scope of this paper and will be left to future work. However, it is clear that in case of step-sweep, which serves new requests immediately in a reservation timeslot, the worst-case response time is bounded by the time required to complete the current nonpreemptible IO. This time is typically in the order of tens of milliseconds. Traditional IO schedulers like sweep as well as GSS [11], have worst-case response times of the order of minutes under heavy load, and at least of the order of seconds under average load conditions.

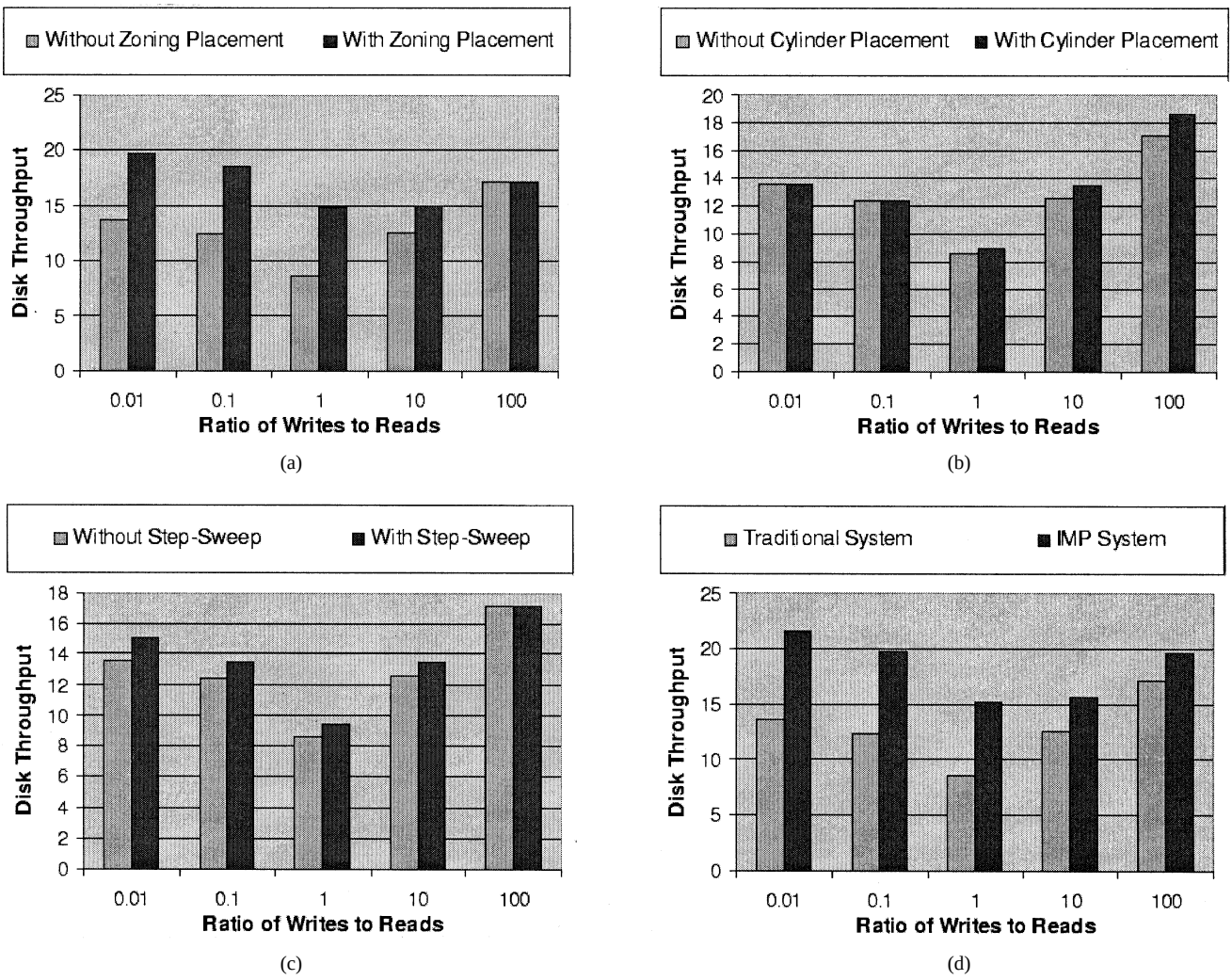


Fig. 7. Throughput improvement due to device management strategies.

C. Observations

We conclude our evaluation section by making the following key observations.

- High-level data organization in the form of the adaptive tree scheme is crucial to maintain disk throughput. The effect of the system configuration on the performance of the IMP system is summarized as follows:
  1) Increasing the system memory beyond a certain threshold does not significantly increase system throughput. At this point, the disk throughput becomes the bottleneck. Beyond this threshold, the disk throughput can only be increased by increasing IO efficiency using the adaptive tree scheme for data organization.
  2) Using the adaptive tree MPEG data organization scheme, the IMP system can optimize for read-intensive as well as write-intensive loads. The achieved throughput is as much as 100% better than that of a traditional file-system for a large range of write–read ratios.
  3) Each sample configuration [SEQ, PTBT, and TBT], of the adaptive tree scheme operates efficiently for a unique sub-configuration defined by the values of write–read ratio (ρ) and interactive stream fraction (f).
- We evaluated the performance gain due to our data management strategies: *zoning placement*, *cylinder placement*, and *step-sweep IO scheduling*. In summary
  1) zoning placement matches stream bit-rates to disk zone transfer rates so as to maximize data throughput of the disk for serving continuous media streams. It improves throughput by as much as 65%;
  2) cylinder placement improves write performance by reducing the seek overhead for write IO's. It increases the throughput by about 10% for write-intensive loads;
  3) step-sweep increases throughput by decreasing memory use. Throughput gains range from 5 to 10%.

  The cumulative effect of the above strategies makes it possible for the IMP system to offer performance gains of as much as 75% over traditional systems.

VI. Related Work

Video broadcasting and multicasting have been used in some cable-based or satellite-based movie-on-demand channels, in which requests for a movie arriving within a period of time are "grouped" (i.e., batched) together and served with a single stream [4], [18], [19]. Schemes have been proposed to support viewing interactivity in a broadcast (or multicast) environment [5], [20], [21]. However, these schemes often double the bandwidth requirement for clients or require a considerable amount of client buffering. In this paper, we propose a client/server dual architecture that manages media data for enabling interactivity between broadcasters/multicasters and clients.

Several schemes have been proposed for supporting fast-scans. These can be classified into three approaches.

1) *Increase playback rate*. This approach increases the playback frame rate for supporting a fast-scan operation. But, this method is impractical, since no TV can display more than 30 fps. Second, the IO bandwidth, memory use, and CPU overhead can be exceedingly high.
2) *Use separate fast-scan streams* [9], [10], [22], [23]. This approach cannot be used in the broadcast (or multicast) scenario because a program is broadcast at a single rate. We can add a unicast channel to deliver fast-scan data, but this provision would require additional network bandwidth.
3) *Skip frames in the regular stream* [24]. The frame-skipping approach, if not designed carefully, can cause low IO-resolution and consequently low system throughput. However, if we break a sequential IO into small IO's to read every k^{th} frame, we can end up with worse, rather than better, IO performance. We believe that skipping frames is nevertheless the right approach for supporting fast-scans under our dual client/server setting. In this study, we have proposed methods to improve IO resolution and to reduce IO overhead in order to implement this approach efficiently.

Data placement and IO scheduling for improving disk throughput have been studied extensively in the context of multimedia file systems [1], [8]–[11], [23]. Most work has been done on read-only systems like VoD, and have not explored simultaneous storage and retrieval of video streams. Even in read-only systems, low-level device optimizations have not been explored. Our work offers a low-level optimized solution to simultaneous storage and retrieval of continuous media. Low-level data placement and IO scheduling strategies can also improve the performance of read-only systems like multimedia data servers. We differentiate our approach from previous efforts in two respects.

1) *Fine-grained* device management: We collect detailed disk parameters directly from a disk to make more effective realtime device management decisions.
2) *Integrated* device management: We provide an integrated strategy, from disk feature extraction, data organization, and data placement, to IO scheduling. We show that tradeoffs often exist between design parameters, and we propose methods to find optimal tradeoffs under different workload scenarios.

Disk profiling has been pioneered by [14]. However, at the time this paper was written, the profiling tool in [14] was not publicly available. The profiling tool that we built is open source [12] and can be easily modified to suit user requirements.

VII. Conclusion

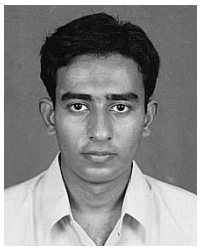
In this paper we have proposed a novel client/server dual architecture and design of IMP, an interactive media proxy that can enable a noninteractive broadcast stream to become an interactive one. Such an architecture is both cost-effective and more practical than the traditional server-based model.

We studied the design parameters of the IMP system and analyzed the tradeoffs between them. To maximize system throughput, we have proposed and evaluated fine-grained device management strategies. After extensive analysis of and experimentation with the IMP system, we conclude our study as follows. High-level data organization in the form of an *adaptive tree* structure is crucial to maintain high throughput under changing load distribution. The data placement strategies of *zoning* and *cylinder placement* employed in IMP improve the throughput of the system significantly. *Step-sweep* IO scheduling further improves the throughput by minimizing seek overhead and IO variability. IMP offers an overall performance gain of as much as 75% over traditional systems, thus demonstrating the importance of performing fine-grained device management for interactive media services.

References

[1] E. Chang and H. Garcia-Molina, "Effective memory use in a media server," in *Proc. 23rd VLDB Conf.*, Aug. 1997, pp. 496–505.
[2] T. Johnson and A. Zhang, "A framework for supporting quality-based presentation of continuous multimedia streams," in *Proc. 4th IEEE Conf. Multimedia Computing and Systems*, June 1996, pp. 169–176.
[3] C. Wang, D. J. Ecklund, E. F. Ecklund, V. Goebel, and T. Plagemann, "Design and implementation of a LoD system for multimedia supported learning for medical students," in *iWorld Conf. Educational Multimedia, Hypermedia & Telecommunications ED-MEDIA*, June 2001.
[4] T. Little and D. Venkatesh, "Prospects for interactive video-on-demand," *IEEE Multimedia Mag.*, pp. 14–24, Fall 1994.
[5] K. C. Almeroth and M. H. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE J. Select. Areas Commun.*, vol. 14, pp. 1110–1122, Aug. 1996.
[6] S.-H. G. Chan and F. Tobagi, "Distributed servers architecture for networked video services," *IEEE/ACM Trans. Networking*, vol. 9, no. 2, pp. 125–136, April 2001.
[7] K. Hua and S. Sheu, "Skyscraper broadcasting: a new broadcasting scheme for metropolitan video-on-demand systems," *ACM SIGCOMM*, pp. 89–101, Sept. 1997.
[8] R. Ng and J. Yang, "Maximizing buffer and disk utilizations for news on-demand," in *Proc. 20th VLDB Conf.*, 1994, pp. 451–462.
[9] P. J. Shenoy and H. M. Vin, "Efficient support for scan operations in video servers," in *Proc. 3rd ACM Int. Conf. on Multimedia*, 1995, pp. 131–140.
[10] W. Tavanapong, K. Hua, and J. Wang, "A framework for supporting previewing and VCR operations in a low bandwidth environment," in *Proc. 5th ACM Multimedia Conf.*, Nov. 1997.
[11] P. S. Yu, M.-S. Chen, and D. D. Kandlur, "Grouped sweeping scheduling for DASD-based multimedia storage management," *Multimedia Syst.*, vol. 1, no. 1, pp. 99–109, Jan. 1993.
[12] Z. Dimitrijevic, D. Watson, and A. Acharya, Scsibench, 2000.
[13] A. N. S. I., SCSI-2 Specification X3T9.2/375R Revision 10L, Jan. 1995.
[14] B. Worthington, G. Ganger, Y. Patt, and J. Wilkes, "Online extraction of scsi disk drive parameters," in *Proc. ACM Sigmetrics Conf.*, 1995, pp. 146–156.
[15] C. Ruemmler and J. Wilkes, "An introduction to disk drive modeling," *Computer*, vol. 2, pp. 17–28, 1994.

[16] N. Talagala, R. H. Arpaci-Dusseau, and D. Patterson, "Microbenchmark-Based Extraction of Local and Global Disk Characteristics," Univ. California, Berkeley, Tech. Rep., 1999.
[17] R. Rangaswami, Z. Dimitrijevic, E. Chang, and S.-H. G. Chan. (2001, Mar.) Fine-Grained Device Mangement for an Interactive Media Server (Extended Version). Univ. California, Santa Barbara, Tech. Rep.. [Online] Available http://www.cs.ucsb.edu/~raju/techreports/device.ps
[18] V. O. K. Li and W.Wanjiun Liao, "Distributed multimedia systems," *Proc. IEEE*, vol. 85, pp. 1063–1108, July 1997.
[19] F. A.Fouad A. Tobagi, "Distance learning with digital video," *IEEE Multimedia Mag.*, pp. 90–94, Spring 1995.
[20] A. Dan, P. Shahabuddin, D. Sitaram, and D. Towsley, "Channel allocation under batching and VCR control in video-on-demand systems," *J. Parallel Distrib. Comput.*, vol. 30, no. 2, pp. 168–179, 1995.
[21] W.-F. Poon and K.-T. Lo, "Design of multicast delivery for providing VCR functionality in interactive video-on-demand systems," *IEEE Trans. Broadcasting*, vol. 45, pp. 141–148, Mar. 1999.
[22] M.-S. Chen and D. D. Kandlur, "Stream conversion to support interactive video playout," *IEEE Multimedia*, vol. 3, pp. 51–58, Summer 1996.
[23] P. Shenoy and H. M. Vin, "Efficient support for interactive operations in multi-resolution video servers," *ACM Multimedia Syst.*, vol. 7, no. 3, May 1999.
[24] W.-C. Feng, F. Jahanian, and S. Sechrest, "Providing VCR functionality in a constant quality video-on-demand transportation service," *ICMCS*, June 1996.



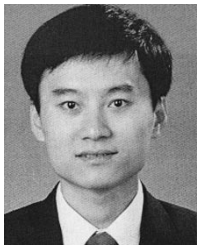
Raju Rangaswami received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Kharagpur, India, in 1999. Currently, he is pursuing the Ph.D. degree at the University of California, Santa Barbara (UCSB). During 1999–2000, he was a Dean's Fellow at UCSB. His research interests include multimedia applications, file systems, operating systems, and storage.

Zoran Dimitrijević received the Dipl.Ing. degree in electrical engineering from the School of Electrical Engineering, University of Belgrade, Yugoslavia, in 1999. Currently, he is pursuing the Ph.D. degree at the University of California, Santa Barbara (UCSB). During 1999–2000, he was a Dean's Fellow at UCSB. His research interests include operating systems, multimedia applications, file systems, and computer architecture.

Edward Chang received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA, in 1999.

He is an Assistant Professor, at the Department of Electrical and Computer Engineering, University of California, Santa Barbara. His research interests include multimedia databases and interactive TV.

Dr. Chang is a recipient of the IBM Faculty Partnership Award in 2000, 2001, and 2002, and the NSF Career Award in 2002.



Shueng-Han Gary Chan received the B.S.E. degree (highest honors) in electrical engineering from Princeton University, Princeton, NJ, in 1993 and the Ph.D. degree in electrical engineering with a minor in business administration from Stanford University, Stanford, CA, in 1999.

He is currently an Assistant Professor with the Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, and an Adjunct Researcher with Microsoft Research Asia, Beijing, China. He was a Visiting Assistant Professor in networking with the Department of Computer Science, University of California, Davis, from September 1998 to June 1999. During 1992–1993, he was a Research Intern at the NEC Research Institute, Princeton. His research interests include multimedia networking, high-speed and wireless communications networks, and Internet technologies and protocols.

Dr. Chan was a William and Leila fellow at Stanford University during 1993–1994. At Princeton, he was the recipient of the Charles Ira Young Memorial Tablet and Medal, and the POEM Newport Award of Excellence in 1993. He is a member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa.