Disk Scheduling for Mixed-Media Workloads in a Multimedia Server[1]

Y. Rompogiannakis[2], G. Nerjes[3], P. Muth[4], M. Paterakis[2], P. Triantafillou[25], G. Weikum[4]

ABSTRACT

Most multimedia applications require storage and retrieval of large amounts of continuous and discrete data at very high rates. Disk drives should be servicing such mixed workloads achieving low response times for discrete requests, while guaranteeing the uninterrupted delivery of continuous data. Disk scheduling algorithms for mixed workloads, although they play a central role in this task, have been overlooked by related multimedia research efforts, which so far have mostly concentrated on the scheduling of continuous requests only. The focus of this paper is on efficient disk I/O scheduling algorithms for mixed workloads in a multimedia storage server. We propose novel algorithms, a taxonomy of relevant algorithms, and study their performance through experimentation. Our results show that our proposed algorithms offer drastic improvements in discrete request average response times, low response-time variability, while serving continuous requests without interruptions.

1 INTRODUCTION

Multimedia storage servers for many applications (such as digital libraries, news-on-demand, teleteaching, etc.) will have to manage mixed-workloads containing continuous data requests (e.g., for video and audio) and discrete data requests (e.g., for text or image data). The performance requirements that have to be met are very stringent and request-type specific. Each continuous data fragment must be delivered within a specified time limit, in order to avoid interruptions in the flow of data, termed hiccups or glitches. At the same time, the server must ensure good performance for discrete requests (such as low response times, low response-time variance, etc.). Offering high performance to both types of requests is a formidable challenge which disk scheduling algorithms must face. However, disk scheduling algorithms for mixed workloads have been largely overlooked by related multimedia research efforts, which so far have mostly concentrated on the scheduling of continuous requests only. With this paper we attempt to fill this gap.

The organization of the paper is as follows. In the rest of this section we briefly review related work, describe the system model, and define the problem at hand. In section 2 we develop new scheduling algorithms, motivate their usefulness, and present a taxonomy. The simulation-based testbed and the performance evaluation of the proposed algorithms are presented in section 3. The paper is concluded in section 4

1.1 Related Work

1.1.1 Disk Scheduling Algorithms

Early scheduling algorithms focused on reducing seek times. The Shortest Seek Time First (SSTF) algorithm [3] achieved this; however, it incurs a high response time variance and is starvation-bound. SCAN scheduling was also proposed in [3]. It serves requests in an elevator-like manner, (as it sweeps the disk in either one or both directions) allowing seek-time optimizations, while reducing the response time variance. In [5] the authors proposed a parameterized generalization of SSTF and SCAN. The V(R) algorithm operates as SSTF except that, every time it changes direction it adds a penalty, dependent on the parameter R and the seek distance. When R=1 (R=0) V(R) reduces to SCAN (SSTF). The performance of these algorithms, as well as other variations of SCAN, such as the LOOK algorithm (which changes scanning direction when no more requests are pending in the current direction) has been studied experimentally in [22] and analytically in [2].

A significant development in modern-disk scheduling was to also target the high costs owing to rotational delays. These algorithms [19, 8] attempted to minimize the sum of seek and rotational delays by favoring, for instance, the request with the Smallest Positioning Time First (SPTF). Recently, disk controllers have been developed which can rearrange the order of submitted requests exploiting their detailed knowledge of the requested blocks' positions and minimize seek and rotational delays [16].

1.1.2 Disk Scheduling Algorithms for Continuous Data Requests

The SCAN algorithm is inapplicable for continuous data requests since it is oblivious of deadlines. The Earliest Deadline First (EDF) algorithm [11] is a natural choice; however, it has poor performance since it does not attempt to reduce the overhead. SCAN-EDF [17] is a hybrid that serves requests in EDF order, but when several requests have the same deadline, they are served using SCAN. Most related recent research has adopted the notion of scheduling with *rounds* [1, 6, 7, 15, 20]. Each continuous data object (stream) is divided into blocks (also called fragments) such that the playback duration of each fragment is some constant time (typically, from one to a few seconds).

The round length represents an upper bound on the time in which the storage server must retrieve from disk the next fragments for all active continuous displays, or some displays will suffer a glitch. Within a round, it is possible to employ either a round-robin or a SCAN algorithm. The latter performs seek optimizations, resulting in better disk throughput. However, this is achieved at the expense of higher start-up latencies; the display cannot be started immediately after the retrieval of its first block but only after the end of the round. This is done to avoid glitches, since the service order differs from round to round. This limitation is not present when using round-robin scheduling, which also has lower RAM buffer requirements since it does not require the double-buffering scheme required by SCAN between successive rounds. A compromise was achieved with the Group Sweeping Scheduling (GSS) algorithm [22]. GSS groups streams and employs round-robin scheduling for the different groups and SCAN scheduling for the streams' blocks in a group. Thus, when there is only one group GSS reduces to SCAN and when each stream is in its own group GSS reduces to round-robin.

1.1.3 Disk Scheduling Algorithms for Mixed Media Workloads

To our knowledge, the only works with some relevance to mixed-media disk scheduling are [17, 10, 12, 13, 14]. The work in [17] overviewed the performance goals for mixed workload scheduling, which are similar to ours, and studied only how some known algorithms for continuous-request scheduling, such as EDF and SCAN/EDF, affect the response time of discrete (or aperiodic, as they call them) requests. A simple scheduling scheme, called "the immediate server approach" [10] was employed for discrete requests, according to which discrete requests were served in between two successive EDF-selected continuous requests.

Since multimedia disk scheduling research has moved away from EDF-based algorithms, the works in [12, 13] attempted to analytically model and predict the performance of multimedia servers with mixed workloads, when scheduling is based on the notion of rounds. The work in [14] is a preliminary attempt to define the issues and present initial algorithms towards efficient mixed-media disk scheduling. The work reported in this paper is a continuation of our work in [13, 14]. We will present novel scheduling algorithms, which introduce drastic performance improvements. We will also define a taxonomy of related algorithms which we hope will be useful to designers, implementors, and researchers, in putting the problem and its prospective solutions in context.

1.2 System Model and Problem Definition

Our storage server receives a *mixed workload* consisting of continuous and discrete data requests. It employs an admission controller which bounds the number of admitted streams, given the availability of resources (such as RAM buffers, disk bandwidth, etc). We will also adopt the notion of scheduling with rounds, outlined above. In addition, the admission controller will reserve some subround period for discrete requests.

Within this framework, the problem at hand is to derive disk scheduling algorithms that will meet the following performance goals:

1. the displays of continuous objects observe no glitches,
2. discrete requests have small average response times
3. discrete requests do not starve, and discrete requests have a small response time variance.

The disk's workload consists of N concurrent continuous-data requests, C-requests, per scheduling round and also discrete-data requests, D-requests. In each round, the desired continuous blocks are known before hand (they are simply the next blocks of the admitted streams). The requests for these blocks are in a waiting queue (C-queue). D-requests arrive according to a Poisson process with rate λ_D and enter their waiting queue (D-queue). In each round we aim to produce a schedule which meets the above performance goals.

2 MIXED-LOAD SCHEDULING: ALGORITHMS AND A TAXONOMY

The taxonomy categorizes the algorithms along two key dimensions:

1. The number of separate *scheduling phases* in a round. The algorithms are categorized as two-phase scheduling (TPS) algorithms when two separate schedules are produced, one for the C-requests and one for the D-requests, and the phases are non-overlapping in time. One-phase scheduling algorithms (OPS) produce mixed schedules, containing both C- and D-requests. TPS algorithms are specified as $TPS: Phase1\_Alg/Phase2\_Alg$.
2. The number of *scheduling levels*. *Hierarchical scheduling* algorithms for D-requests will be presented. These algorithms are based on defining *clusters* of D-requests. At the higher level, the algorithms are concerned with the efficient scheduling of clusters. At the lower level, the algorithms are concerned with the scheduling of a cluster's requests. Hierarchical algorithms are specified as $\frac{Alg\_High}{Alg\_Low}$, where Alg_High and Alg_Low are the algorithms used at the higher and lower levels, respectively.

In this paper for all algorithms we choose to serve C-requests according to the SCAN policy. The benefits of the SCAN policy are well known and widely adopted even for C-requests and leave little room for improvement. Thus, we will mostly concentrate on how to schedule the D-requests within the round's SCAN order for the C-requests.

2.1 Clustering and Hierarchical Scheduling

We define a *C-interval* as the disk interval between the disk cylinders of any two requested C-fragments that are successive in the current SCAN's direction. All D-requests belonging to the same C-interval form a cluster. Hierarchical scheduling algorithms in essence decompose the problem of scheduling all D-requests into the subproblems of deriving efficient schedules of, first, the clusters and, then, each cluster's requests.

We propose two lower-level scheduling algorithms. Their motivation is as follows. Given a non-trivial set of C-requests, clustering results in a number of small C-intervals (such that any seek within the C-interval is a "short seek" [18]). Applying seek-optimizing algorithms (such as SCAN) in such small intervals (i) does not result in major savings, and (ii) focuses at the wrong overhead component. To give a concrete example, consider a disk drive with 1500 cylinders and a set of 10 requested C-fragments, randomly distributed over the disk's surface. Any seek within the defined 150-cylinder-wide C-intervals will be a "short seek", the cost of which, given current technology, is only a few (e.g., less than six) milliseconds. However, in such drives the rotational delay can be significantly higher than the seek cost (e.g., up to 8 milliseconds, for the drives with the currently fastest rotating speed). Furthermore, the transfer time can actually be significantly higher than both of the above costs (e.g., a 200KB image in a disk with 10MB/sec transfer rate, requires a transfer time of 20 milliseconds). Given, as mentioned in the introductory section, the capabilities of modern disk controllers, further optimizations, such as the ones that follow, are possible.

The CI-SATF (C-Interval Shortest Access Time First) Algorithm

According to this algorithm:

1. The D-queue is partitioned into *interval queues*, one per C-interval. The D-requests are distributed to the appropriate interval queues.
2. Within each interval queue the D-requests are sorted according to their total disk access costs (seek, rotation, and transfer times).
3. CI-SATF serves D-requests according to their order in the interval queue.

The CI-OPT (C-Interval Optimal) Algorithm

CI-SATF is a greedy algorithm, which may not produce optimal schedules. CI-OPT looks at all D-requests in the interval queue, considers all possible schedules, and determines the one with the minimum cost. It is well known that such optimality problems are reducible to the Traveling Salesman Problem (TSP) [4, 9]. For this reason, we place an upper bound on the number of cluster members, (e.g., five, thus leading to the CI-OPT(5) algorithm). According to this algorithm:

1. The interval queues are constructed as in CI-SATF, except that when one is to contain more than five D-requests, it is recursively subdivided into two equal-sized queues.
2. The interval queues are then sorted in the order computed by CI-OPT.

The overall (hierarchical) algorithm serves the clusters of D-requests in some order (which in this paper is SCAN) and within each cluster (C-interval) serves the D-requests according to the order in the corresponding interval queue as produced by CI-SATF or CI-OPT(5).

In addition, before serving a D-request within a cluster, the algorithms check if there will be enough time to serve all remaining C-requests (using SCAN). If so, the D-request is served; else, it is postponed until the next round. Note that this check can rapidly be performed using the known detailed models for the seek cost, the rotational delays, and the transfer times.

2.2 Two Phase Scheduling

In two phase scheduling (TPS) schemes we serve C-requests and D-requests separately into disjoint time periods (C-period and D-period) within each round. Each schedule can be produced by any known or novel algorithm. In this paper we will assume that at the beginning of each round a schedule for the N C-requests is constructed according to the SCAN policy, and then in the remaining time (until the end of the current round) the D-requests are served according to either the SCAN discipline or a hierarchical scheduling discipline. Thus, we will focus on $TPS: SCAN / SCAN$ and on $SCAN / \frac{SCAN}{CI-OPT(5)}$, respectively. For the latter (hierarchical) algorithm, the clusters which will be visited in SCAN order during the second phase are defined as in the CI-OPT(5) algorithm, except that instead of a C-interval, the whole disk is considered.

2.3 One Phase Scheduling

In one phase scheduling schemes C-requests and D-requests are served together in an interleaved manner. D-requests that arrive during the service period of C-requests, do not necessarily have to wait until the end of that period. We propose and study the following one-phase scheduling algorithms.

FAMISH (Fair Mixed-scan ScHeduling)

D-requests are ordered in the D-queue based on their arrival time. FAMISH ensures that:

- All C-requests will be served within the current round.
- No D-request D_i will be served in an earlier round than D_j, if D_j is ahead of D_i in the D-queue.

In particular, FAMISH will:

1. construct a SCAN schedule for the C-requests for this round.
2. incorporate the D-request at the head of the D-queue, in its proper position in the current SCAN schedule, and
3. calculate whether serving this SCAN schedule (which includes this D-request) will result in a hiccup for one or more C-requests in the SCAN.
4. if not, the iteration continues from step (2).
5. if so, the iteration stops, this D-request is removed from the current SCAN for this round and the requests in

the resulting SCAN are served.

In addition to $OPS: FAMISH$, we will also consider the $OPS.\frac{SCAN}{CI - SATF}$ and $OPS.\frac{SCAN}{CI - OPT(5)}$ algorithms.

3 EXPERIMENTATION AND PERFORMANCE RESULTS

3.1 Experimental Testbed

Our simulation results have been extracted by modeling a modern disk drive. The parameters of the disk drive are given in Table 1.

Cylinders	1,449
Revolution Speed	7,200 RPM
Rotation Time	8 ms
Number of Zones	1
Transfer Time	10 MBps

Table 1: Disk Characteristics

The seek time modeling as a function of the distance is given in Eq. 1 (for the HPC2200A drive).

$$Seek(d) = \begin{cases} 3.45 + 0.59\sqrt{d}, ms \;\; if \;\; 1 \le d \le 616 \\ 10.8 + 0.012d, \; ms \;\; otherwise \end{cases} \qquad (1)$$

The rotational delay for each D-request is computed by keeping track of the starting disk sector within the cylinder containing the D-request. Given the constant angular velocity of the disk and the seek distances, along with the accurate seek cost model above, we can calculate which sector will be passing beneath the disk head at the end of the seek and thus compute the actual rotational delay. The transfer time is given from the disk's transfer rate and the block size.

The C-requests sizes are Gamma distributed with mean 200000 bytes and standard deviation 100000 bytes. The values for C-requests reflect typical data characteristics for MPEG-1 data. The sizes of D-requests are typically smaller and obey a normal distribution with mean 70000 bytes and standard deviation 10000 bytes

The arrival of D-requests is driven by a Poisson process with arrival rate λ_D and it is assumed that arriving D-requests are uniformly distributed over the disk.

All simulations consist of five independent runs of 20,000 rounds each (which are enough for the algorithms to reach steady-state). During each run: the number of C-requests is held constant and the results for a specific performance metric are calculated with cumulative data. For example, the steady-state D-request average response time is obtained from the ratio of total response time of D-requests to the total number of serviced D-requests.

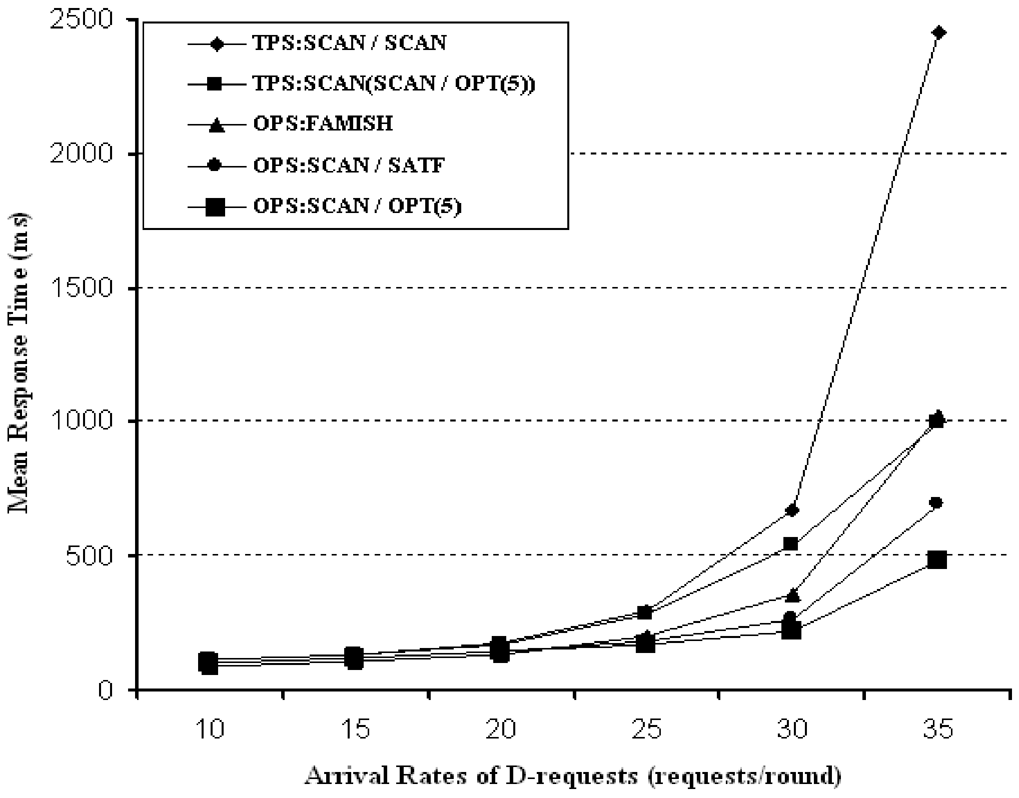

Figure 1: The Mean Response Time of D-requests for N=10

The final value for a metric is calculated as the mean of that metric under the five runs

3.2 Performance Metrics

We measure the *Mean Response time* of the D-requests, as the sum of the request waiting time (in a queue) and its service time. The service time is the sum of the seek, rotation, and transfer time.

3.3 Results

We compare the performance of the above scheduling algorithms. All algorithms prevent glitches (all N C-requests are completed before the end of the round). The round's duration is set to one second. In Figure 1 we present the simulation results for the mean response time of D-requests for different arrival rates when $N = 10$. Serving 10 C-requests per round according to the SCAN algorithm and without intermixing D-requests in intervals between successive D-requests, consumes about 35% of the total round time.

The major conclusions are that:

1. $TPS : SCAN / SCAN$ has consistently the worst performance.
2. $TPS : SCAN \frac{SCAN}{CI - OPT(5)}$ performs worse than OPS algorithms for low and medium loads. But for very high loads it outperforms $OPS : FAMISH$ in the average response time (and its variability, although not shown for space reasons). This emphasizes the usefulness of hierarchical scheduling: its local optimizations achieve smaller average response times, while the high-level fair scheduler ensures small variability around the mean.
3. $OPS : FAMISH$ performs very well for low and medium D-request arrival rates. For heavier workloads the hierarchical OPS algorithms considerably outperform it.

4. $OPS : \frac{SCAN}{CI - SATF}$ and $OPS : \frac{SCAN}{CI - OPT(5)}$ perform better as the value of λ_D increases since it gives a greater chance for the local optimizations to work. For heavier workloads (i.e., when $\lambda_D \geq 35$ $OPS : FAMISH$ is more than 250% worse compared to $OPS : \frac{SCAN}{CI - OPT(5)}$.
5. $OPS : \frac{SCAN}{CI - OPT(5)}$ performs, as expected, better than $OPS : \frac{SCAN}{CI - SATF}$ as λ_D increases.
6. Although not shown, for space reasons, for $\lambda_D > 35$, for some algorithms the system enters an unstable state. For instance, for $\lambda_D > 35$, $TPS : SCAN / SCAN$ leaves at the end of the experiment more than 2% of the D-requests "stranded" in the D-queue. For the other algorithms for $\lambda_D < 38$ the "stranded" D-requests are less than 1%. Again, the hierarchical algorithms enter into unstable states at greater λ_D values.

4 CONCLUSIONS

We have considered the problem of disk scheduling for mixed-media workloads. Despite the fact that such workloads are typical of many applications, related research efforts have largely overlooked this problem. We have presented several algorithms which aim to ensure the hiccup-free display of continuous objects, while ensuring low average response times and low response-time variance for discrete requests. We have contributed a taxonomy of related algorithms, organized along two key dimensions: the number of separate scheduling phases and the number of scheduling levels.

We have implemented the proposed algorithms in a detailed simulation testbed. We have shown that the hierarchical scheduling algorithms we propose can offer very drastic improvements, up to several hundred percent, in the average response times of D-requests. Also, these impressive response times do not occur at the cost of high response-time variance, thus achieving fair D-request schedules.

On-going work includes the design and evaluation of alternative clustering techniques for the hierarchical algorithms, alternative higher-level scheduling algorithms and efficient heuristics for TSP which can avoid the upper bound of five D-requests per cluster and replace the OPT(5) algorithm. Also, we plan to investigate to what extend similar hierarchical algorithms, in which the higher level algorithms effectively avoid starvation problems introduced by the lower-level optimizations, are suitable for (file-system or database-system) disk workloads.

REFERENCES

[1] S. Berson, S. Ghandeharizadeh, R.R. Muntz and X. Ju. Staggered Striping in Multimedia Information Systems. Proc. of the Intern. Conf. on Management of Data (SIGMOD), Minneapolis, Minnesota, pp. 79-90, 1994.
[2] Coffman, Jr and M. Hofri. Queueing Models of Secondary Storage Devices. Stochastic Analysis of Computer and Communication Systems. Ed. Hideaki Takagi, North-Holland, 1990.
[3] P.J. Denning. Effects of scheduling on file memory operations. Proc. AFIPS Conf., April 1967, pp. 9-21.
[4] M.R. Garey and D.S. Johnson. Computers and Intractability: A guide to the theory of NP-Completeness. W.H. Freeman, 1979.
[5] R. Geist and S. Daniel. A continuum of disk scheduling algorithms. ACM Transactions on Computer Systems, Feb. 1987, pp.77-92.
[6] D.J. Gemmel, H.M. Vin, D.D. Kandlur, P.V. Rangan, and L.A. Rowe. Multimedia Storage Servers: A Tutorial. IEEE Computer, Vol.28, no.5, May 1995, pp.40-49.
[7] S. Ghandeharizadeh, S.H. Kim and C. Shahabi. On Disk Scheduling and Data Placement for Video Servers. ACM Multimedia Systems, 1996.
[8] D. Jacobson and J. Wilkes. Disk scheduling algorithms based on rotational position. Techical Report, HPL-CSP-91-7, Feb 1991.
[9] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys. The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, 1986.
[10] T.H. Lin and W. Tarng. Scheduling periodic and aperiodic tasks in hard real time computing systems. Proc. Intern. ACM SIGMETRICS, Conf., 1991.

[11] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard real time environment. J. ACM, Jan 1973, vol. 20, no. 1, pp. 46-61.
[12] G. Nerjes, P. Muth, G. Weikum. Stochastic Performance Guarantees for Mixed Workloads in a Multimedia Information System. Proc. IEEE Intern. RIDE Workshop, April 1997.
[13] G. Nerjes, Y. Rompogiannakis, P. Muth, M. Paterakis, P. Triantafillou, and G. Weikum. Scheduling strategies for mixed workloads in multimedia information servers. Proc. IEEE Intern. RIDE Workshop, Feb. 1998
[14] G. Nerjes, Y. Rompogiannakis, P. Muth, M. Paterakis, P. Triantafillou, and G. Weikum. On mixed-Workload Multimedia Storage Servers with Guaranteed Performance and Service Quality. 3rd Intern. Workshop on Multimedia Information Systems, Sept. 1997.
[15] B. Ozden, R. Rastogi and A. Silberschatz. Disk Striping in Video Server Environments. In Proc. of the Intern. Conf. on Multimedia Computing and Systems (ICMCS), June 1996.
[16] Quantum Corp., Storage Basics, ORCA. www.quantum.com/src/storage/_basis.
[17] A.L.N. Reddy and J.C. Wyllie. I/O Issues in a Multimedia System. IEEE Computer, March 1994, pp. 69-74.
[18] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. IEEE Computer, March 1994, pp. 17-28.
[19] M. Seltzer, P. Chen, and J. Ousterhout. Disk Scheduling revisited. Proc. 1990 USENIX Technical Conf, pp.313-323.
[20] P. Triantafillou and C. Faloutsos. Overlay Striping and Optimal Parallel I/O in Modern Applications. Parallel Computing, March 1998, (24) pp 21-43.
[21] B.L. Worthington, G.R. Ganger, and Y. Patt. Scheduling algorithms for modern disk drives.In Proc. of the 1994 ACM SIGMETRICS Conference, pp 241-251.
[22] P.S. Yu, M.S. Chen and D.D. Kandlur. Grouped sweeping scheduling for DASD-based multimedia storage management. ACM Multimedia Systems, 1(3): 99-109, 1993.

---

1 This research has been supported by the Esprit Long Term Project, no. 9141, HERMES.

2 Yannis Rompoyannakis, Michael Paterakis, and Peter Triantafillou are with the Multimedia Systems Institute of Crete and the Dept. of Computer and Electronics Engineering at the Technical University of Crete, Greece.

3 Guido Nerjes, is with Dept. of Computer Science at ETH Zurich, Switzerland.

4 Peter Muth and Gerhard Weikum are with the Dept. of Computer Science at the University of Saarland, Germany.

5 Peter Triantafillou is the contact author: Chania, 73100, Greece. Tel.no: +30.821.37230. FAX: +30.821.37202

Email: peter@cha.forthnet.gr, peter@ced.tuc.gr