Hard Disk Drive for HD Quality Multimedia Home Appliance *

Jongmin Gim Jaehyeok Chang Hyungwon Jung Youjip Won
Dept. of Electrical and Computer Engineering
Hanyang University, Korea
{jmkim|syia|mulldog|yjwon}@ece.hanyang.ac.kr

Junseok Shim Youngseon Park
Storage Lab
Samsung Electronics, Korea
{junseok.shim|ys_park}@samsung.com

Abstract

In this work, we develop a novel audio and video (A/V) disk which is not only stronger for fragmentation but also getting higher bandwidth for multimedia home appliances. For our work, multimedia workload characteristics and disk overheads in multimedia home appliance are analyzed and then the concept of extent is redefined. If track size is only multiple size of extents, every I/O operation do not have track switch overhead. We suggest three ways of implementing our model in a real disk drive and merits and demerits of each scheme are analyzed. Disk performance can show various patterns according to input workloads generated by filesystem. Designing fragmentation model is based on EXT3 filesystem and Disksim 3.0 are used for the simulation based performance evaluation. A state-of-the-art disk layout is added and many parameters are modified to reflect actual response time pattern of real disk in Disksim. Through our experiments, our model shows a significant improvement in bandwidth from 5% to 25% fragmentation ratio.

keywords: extent, multimedia, A/V disk drive, disksim

1. Introduction

1.1. Motivation

With a great advancement of multimedia contents and disk technology, multimedia home appliance provides services with bandwidth requiring high definition quality and uses disk drives as a secondary storage system. However, general purpose of most disks used in home appliances is computing. Audio-Video (A/V) disk has a different requirements from a general purpose of computing disk such as power consumption, short start-up latency, small buffer size and contiguous streaming service. Especially, one of the most significant requirements of A/V disk is supporting multi-sessions. Although a bandwidth of modern disk is bigger than any other bandwidths of contents, it is not optimized to support contents when several sessions are working at the same time.

In this work, we suggest more optimized A/V disk for high quality multimedia contents. Any electronic devices for high bandwidth multimedia contents (e.g. HDTV and PVR) support a bandwidth of about 19.2Mbps. For these devices, supporting a single session of multimedia contents is not a problem. Rather, disk bandwidth may not enough to support multi-sessions. According to the bandwidth of multimedia contents, a data size of multimedia file to be transmitted in a single Input/Output (I/O) in a disk drive can be decided during a single period.

For developing an A/V disk drive, application layer and filesystem layer need to support disk drive operation to enhance its performance. As for several storage systems, companies have their own multimedia applications, multimedia filesystem, and multimedia hard disks. If we have all layers for application, the layer would send optimized IOs to filesystem, and filesystem would send optimized IOs to device driver of disk. Finally, a disk can do work properly in specific A/V systems. Another merit is simplifying IO path through deleting unnecessary parts in the specific multimedia systems. It will improve the performance, and therefore save costs by deleting unused hardware/software modules.

Our work figures out an optimal extent size in specific multimedia systems. An extent means a set of logical adjacent blocks in a file system, but we redefine it as physical

*This research is in part supported by KOSEF through National Research Lab (R0A - 2007 - 000 - 20114 - 0) at Hanyang University

978-0-7695-3243-1/08 $25.00 © 2008 IEEE
DOI 10.1109/ICCSA.2008.48

adjacent sectors in a disk. Disks use an extent as a basic unit of an IO. We set a track size to the size of multiple extents. Since this measure can reduce the track switch time, more bandwidth is gained in the transfer time. Filesystem also uses extents as a basic IO unit. It means extent size should be a multiple size of blocks. Applications determine the IO size from the extent size. Especially, it is closely related with an application buffer size. Through integrating all layers, this could be optimized for the home appliances. We assume that filesystem layer and application layer are optimized to use extent. Also, we develop A/V disk drives.

1.2 Related Work

There are many factors in the disk overhead. Among them, Seek time is primary factor in storage system. The Seek time means the time for moving of a disk arm to target a cylinder. Since the seek time is the most heavy part in the disk overhead, there have also been many researches. Some researches tried to reduce this by using a disk scheduling policy[14, 6].

There were methods to enhance disk performance by reducing fragmentation[3]. The fragmentation occurs along with many read/write operations in a filesystem layer. A single IO access is transferred from applications, through a filesystem, to a disk. The fragmentation can divide the single IO into several sub IOs, and it can make an additional overhead. In reducing fragmentation in a disk layer, there is a method using re-write policy[5]. In a writing operation, if a single IO were separated due to physical space, this method searches an empty track, and re-write after merging the IO. This method can reduce for a single file, but it could not enhance disk performance in a multi-session. Since an A/V system always takes multisessions into account, this method cannot be properly working in the A/V system.

There is another method that reduces seek time with proper data placement. In response to the time of seek model[8], the response time is represented $a + b\sqrt{x}$ until reaching to a specific threshold. From the equation, a and b denote disk constant, x denotes cylindrical distance. This research insists that there is a flat linear section in the short seek distance. If data could be placed in the flat linear section, it could perform well in multisessions.

Power consumption[16] is significant in A/V systems. A multimedia playback can consume power more than general computing systems since it can access disks contiguously at regular intervals. One research[15] reduced power consumption by switching standby disk state when the disk was not used. If the IO size is large enough, the number of disk accesses can be reduced. With a longer period, disks can stay longer in a standby mode.

There would be no track switch in a single IO operation [10]. Since this method approaches to a filesystem layer, implementation was very difficult. First, the filesystem knows all disk geometry[4, 12]. In case of SCSI disk, it can be somewhat easy to extract the disk geometry since they have IO command for finding disk geometry[9, 11]. On the other hand, IDE disk has only a method of extracting the disk geometry by using the response time of seek. With this method, it takes a very long time to extract disk geometry and it is not even accurate.

2 Multimedia Workload

Application with multimedia playback basically have two processes. First process is multiplexing and demultiplexing, the next process is buffering. Multimedia players get data from disks to process buffer, then buffer is filled with data. If players play back multimedia files, de-multiplexing processes translate data, which are displayed in buffer. Players generally use double buffering for contiguous playback, multimedia data have different file size and bandwidth depending on the types of contents. The file size of multimedia contents in five years[1] is increased about double times, and the bandwidth is growing extremely. In case of mp3 file, it has a very low bandwidth about 128Kbits/sec, MPEG2 has a medium quality bandwidth about from 4Mbits to 6Mbit, and HDTV has a very high bandwidth about 19.39Mbits/sec[13]. As for contents bandwidth, a storage device needs to have bandwidth to support it. For modern 7200rpm disk drives, there are no problems to transfer multimedia data because they have very high bandwidth about 100MB/sec. Since disk bandwidth is evaluated by sequential IO operation, overheads are not taken account. Random workloads can make diverse transfer times because they need many physical arm movements, electrical head switches, and waiting time of a targeted block underlying arm.

Multimedia contents are not a sequential IO operation. A multimedia player uses only fixed size of multimedia IO data in a single period. In the case of a single session in a single period, a disk arm moves very short distance since most single files are allocated adjacently. On the other hand, multi-sessions IO operations can make a large overhead because places of required data could not be predicted. For contiguous playback multimedia contents, retrieval data needs to be larger than data for playback. Let b denote file system block size, n_i denote the number of blocks in i^{th} session, r_i denote playback rates in i^{th} session, and $T(s)$ denote period. In the A/V system, Eq. 1 needs to be in a sufficient condition.

$$\vee i, bn_i \geq r_i T(s) \tag{1}$$

Since r_i is related with contents bandwidth, $T(s)$ can decide the minimum n_i. With growing $T(s)$ size, the minimum n_i

can have larger size. Period $T(s)$ should be longer than total transfer time. Eq. 2 shows multimedia contents requirements for a contiguous playback. $f(bn_i)$ means transfer time for bn_i, and $O(s)$ means overhead time for bn_i data, respectively.

$$T(s) \geq \sum_{i=1}^{n} f(bn_i) + O(s) \quad (2)$$

3 Storage overheads for A/V system

We develop tools to analyze disk drives. The first tool is a track boundary extractor, the other tools are aimed at seeking profiles and skewing patterns. Table 1[1] shows five disks for our experiments. Those five disks do not have any special commands to extract disk drive geometry. We only use a response time measurement and this measurement will be shown in the next paper.

	Vendor	Cap	RPM	Heads	Int
Disk1	Samsung	120G	5400	4	PATA
Disk2	Samsung	300G	7200	6	PATA
Disk3	WD	320G	7200	4	PATA
Disk4	Seagate	320G	7200	4	SATA
Disk5	Hitachi	320G	7200	4	SATA

Table 1. Specifications of five disk

If a client issues an IO request, it would require seek time[8] for searching target cylinder and rotation latency[6, 7] to wait before a targeted sector has arrived to a disk head. We define seek time and rotational latency[6] overhead as $T_{overhead}$. Data transfer time is composed of transfer time and bus transfer time.

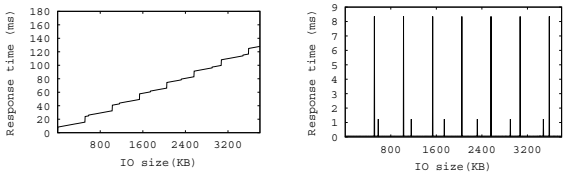
When only a single file playback is used in home appliances, we can have many methods to reduce seek time. But a single file playback is not a problem for modern disk drives since it has much bigger bandwidth about 60 to 100MB/s. Data transfer time is composed of transfer time and bus transfer time. When data blocks for IO are in 2 or more tracks, it can make track switch time overhead and that can be included in the transfer time. It can be a head switch but since it happens rarely, our model does not consider the head switch time.

Let b denote file system block size, n_i denote number of blocks requested in i^{th} session, B_{max} denote max bandwidth of disk, and ε denote transfer overhead, i.e. track switch time. Transfer time for i^{th} session can be Eq. 3.

$$f(bn_i) = \frac{bn_i}{B_{max}} + \varepsilon \quad (3)$$

[1]In table 1, Cap: capacity and Int: interface.

If the IO was not in 2 tracks, there is no track switch. In case of big transfer time such as a sequential IO operation, track switch overhead is not significant, but it can be significant in random workloads.



(a) Response time (b) Difference graph

Figure 1. Response time of Disk2

Fig. 1 (a) shows the track switch time overhead in Disk2. In this graph, we use 256KB filesystem read-ahead, with the disk look-ahead disable, O Direct disable, and check the response time according to growing IO size. Disk2 specific details are illustrated in the table 1. In Fig. 1 (a), a large jump means an IO command to be splitted, and a very small jump means a track switch overhead. In order to see the track switch more clearly, Fig. 1(b) is a difference graph of Fig. 1(a). There is a very large gap between the response time of 512KB IO size and 516KB IO size. It is the IO command split which makes about 8ms response time margin. On the other hand, when there is a very small gap in the graph it is a track switch overhead which makes about 1.2ms, and it shows uniform distribution in the same zone. The track switch is a small overhead, but it can be significant when IO size is in a single track or two tracks.

	disk1	disk2	disk3	disk4	disk5
ts (ms)	1.57	1.17	0.86	1.28	1.56
1 rev. (ms)	11.11	8.33	8.33	8.33	8.33
ts/rev.(%)	14.13	14.04	10.32	15.36	18.72

Table 2. track switch overheads for five disks

Table 2[2] shows track switch overhead ratio comparing with 1 revolution time when disk drive read data with the size of single track or two tracks. Track switch overheads effect bandwidth performance for five disk drives ranging from 14% to 18%. Track switch time is relatively significant when disk reads about single track.

[2]In table 2, ts: track switch time and rev.: revolution time.

4 Aligning Extents to track

4.1 Performance Model

The overheads caused by a single IO request are composed of seek time latency, rotational latency, and track/head switch time. In our system, a single IO can be a single extent or several extents. A multimedia player requires a single IO request per a single period. It means that it is not a sequential read since the multimedia player could not access data contiguously. We can reduce track switch overhead if extents for IO request were in a single track. Between the IO request and period, there is a relationship with bandwidth of multimedia data. A multimedia file can be expressed as a sum of IOs in every period, and we can expect bandwidth enhancement by reducing the track switch time when every IO was aligned with track. Let size of track denote t_s, and track to track switch time denote δ. We can convert from Eq. 3 to Eq. 4 if we consider probability of track switch occurrence.

$$f(bn_i) = \frac{bn_i}{B_{max}} + \frac{bn_i}{t_s} \times \delta \tag{4}$$

We do not need to consider track switch time when every IO is aligned with a track. But, if IO request size were bigger than track size, the track switch occurs. At least, every IO request is aligned by track, we can reduce one track switch in the last track. Eq. 5 shows we can reduce track switch time, compared to Eq. 4.

$$f_{EAT}(bn_i) = \frac{bn_i}{B_{max}} + \lfloor \frac{bn_i}{t_s} \rfloor \times \delta \tag{5}$$

Since a data transfer performance can be evaluated by track switch time and transfer time, gained bandwidth through reducing track switch time can be reduced when IO size is larger.

Another point is the buffer size. A multimedia application could access data from a disk with the same size of the buffer's. We can reduce the number of times of disk access if we set large buffer size. With a longer period, the application gets more data in one period. Nevertheless, startup latency for waiting to be filled with buffer can be longer. This is critical to end users who are using this application. End users generally want to see the multimedia contents as soon as they press the start button of the multimedia application. As buffer size gets much smaller, overhead would be much more larger. It is attributable to the increased number of times of the disk access. Finally, minimum buffer size has a significant meaning to playback multimedia data. An application has a different buffer size whether an IO is aligned with a track or not. Since one of the major points of our goal is to reduce buffer size, we need to calculate minimum buffer size. Let's denote number of track switches in Eq. 3 as q_i. To actualize track switch, we can replace Eq. 3 with Eq. 6.

$$f(bn_i) = \delta q_i + \frac{bn_i}{B_{max}} \tag{6}$$

It can be substituted $f(b_{ni})$ of Eq. 2 for Eq. 7.

$$T(s) \geq O(s) + \sum_{i=1}^{s} (\delta q_i + \frac{bn_i}{B_{max}}) \tag{7}$$

Since our goal is to find n, Eq. 7 should be the form of n. In vector space, optimal $T^*(s)$ is

$$T^*(\mathbf{s}) = O(\mathbf{s}) + \delta\mathbf{q} + \frac{b\mathbf{n}}{B_{max}} \tag{8}$$

Next, substitute $T^*(s)$ in Eq. 8 to $T(s)$ in Eq. 1, then put n on the left hand side of Eq. 8

$$\mathbf{n} = \frac{(O(\mathbf{s})\mathbf{I} + \delta\mathbf{q})\mathbf{r}}{b(\mathbf{I} - \frac{\mathbf{r}}{B_{max}})} \tag{9}$$

Finally, we convert domain from vector space to scalar space

$$\| n \| \geq \frac{(O(s) + \delta \sum_{i=1}^{s} q_i) \sum_{i=1}^{s} r_i}{\frac{b}{B_{max}}(B_{max} - \sum_{i=1}^{s} r_i)} \tag{10}$$

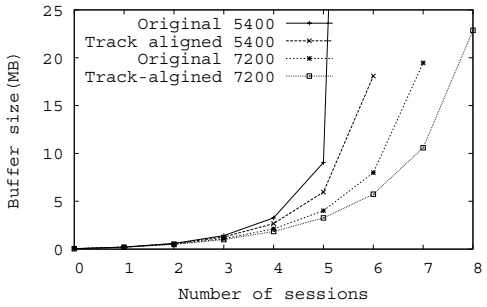


Figure 2. Minimum buffer requirements

In Fig. 2, we set the transfer rate as 25MB/s, track switch time as 2ms, media bandwidth as 19.2Mbps. In every legend, number means hard disk RPM. According to the increasing number of sessions, buffer requirements are also increasing. Especially, when the number of sessions are 6, the original disk cannot support sessions. Since original disk has additional values in a denominator comparing with Eq. 10, the slope of graph can be much sharper.

5 Experiments

5.1 Disk simulator for Extent model

Since we do not have a disk for extents aligned track model, we need to develop a disk simulator for a perfor-

mance experiment. Disksim [2] is a good performance evaluation tool which can extract a lot of hard disk drive information. We evaluate our model via trace driven simulation. However, Disksim only supports Traditional Layout and cylinder serpentine(Fig. 3) which modern disk drives do not use.

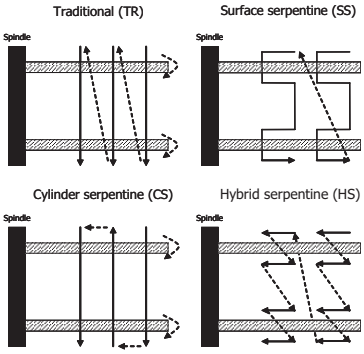


Figure 3. Four types of disk layout

To experiment disk drive performance, we implement additional disk layout to make the same Disk1 which has hybrid serpentine layout (Fig. 3) to Disksim. In order to implement hybrid serpentine layout, we modify disk model as part of Disksim 3.0. Especially, we add dm_translate_ltop, dm_translate_ptol, and dm_get_track_boundaries functions which have 140 lines code. The 12 models supported by Disksim use real disk parameter which is extracted by a real disk. However, we have no method to extract ATA disk parameters. We set proper disk value, and compare real disk response time with Disksim response time, until response time of Disksim becomes similar with that of the real disk drive. It is all too hard because there are too many parameters that need to be modified. We compare response time of real disk and response time of our disk model to modify parameters.

To verify whether or not the model of Disksim is exacted from the real disk drive model, we issue same workloads which are generated by BlkTrace to Disksim and Disk1 drive. BlkTrace is a block layer IO tracing mechanism which provides detailed information about request queue operations up to user space. To make workload extracting program, we mount Disk1 with 4 same size partitions, and disable Read-lookahead operation using by HDPARM, because read-ahead can interrupt extracting regular workload. We generate workload to collect Disk-Level IOs which occurred when they read every 512KB multimedia file. We modify Disksim parameters to get similar response time of Disk1 after issuing workloads to Disksim.

Fig. 4 shows the correctness of response time between Disksim model and disk drive. Graphs of the two models' response time look very similar, and average response time

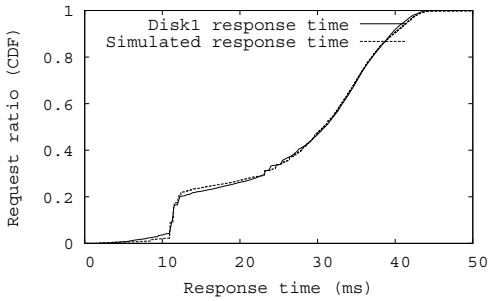


Figure 4. Comparison of response time of Disksim and Disk1

of each model is 27.61ms and 27.74ms, respectively.

5.2 Experimental setup

In order to conduct an experiment, we use kernel 2.6.22 Linux machine, with Intel Pentium 4 3.0GHz for CPU. The test is done by Disksim 3.0 which has hybrid serpentine layout since there is no disk that has our model. Disksim simulator is patched with 3.0 version and complied by gcc 3.4 version. We make an assumption that hard disk have data which are adopted our model, and issues workloads, which are generated by workload generating tool, to Disksim, and analysis response characteristic. Workload to input Disksim is ASCII format, and a single request has parameters of arrival time, device number, block number, request size, and request flags.

In setup for experiments, we use HDTV bandwidth in 4 multi-sessions, and IO size is 512KB. Disk1 has 128KB extent size with HDTV bandwidth, and the number of sessions is 4. Since original Disk1 does not use extent, there is no extent size. File size is 1GB in every session, Total sectors are 261934392 in original disk, and extent model has 216879104. Extent aligned track model makes a spare area, total sector size is smaller than original Disk1 geometry.

5.3 Experiment results

We assume that a single file is written continuously. there are 4 multi-sessions, every session is located at the most outer diameter, 2/3 diameter, 1/3 diameter, and the most inner diameter. We only tested read operations, and every IO size is 512KB. Disksim reads IOs until playback of every session will be finished. We tested 3 disk models. The first is a zone-map change model. This method changes frequency to align track size as multiple extent size. The second is remaining sectors which we do not use. If an extent is aligned to tracks, spare areas could be created. We

allocate spare areas to the end of the tracks. The third is a skew correction method. This method compensates skew values for the reason a disk head does not pass the spare areas.

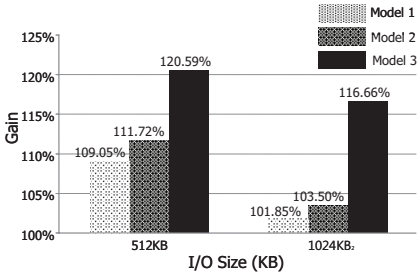


Figure 5. Performances of three layout models in 15% fragmentation ratio

Fig. 5 shows the performance results of 3 models. According to the figure, 100 %gain means response time of original Disk1 layout. Our 3 models have all positive performance compared to the original Disk1 layout. Since the third model gets additional gain which did not rotate spare areas, it shows the best performance. Also, 15% fragmentation means ratio of IO command split occurrence. The third method gets 20% bandwidth gain when IO size is 512KB, and gets 16% when IO size is 1024KB. Fig. 5 shows that we could get more performance gain when IO size is 512KB. Since 1024KB IO size is larger than that of single track in Disk1, our disk model also should be in 2 tracks. It can make at least 1 track switch. Finally, overhead ratio comparing with original Disk1 model, extent aligned track model might be reduced.

6 Conclusion

In this paper, we propose developing A/V disk drives for HD quality home appliances. First, we suggest track aligned extent disk model. This model can reduce application buffer size, startup latency, IO command split, and so forth. Especially, we suggest 3 implementation models of aligning extents to track. Model1 is the simplest method, but it has low performance gain. Model3 shows the best performance gain, but its weakness is difficulties of implementation. Model3 requires compensating for skew values in every zone. Especially, since Model3 has spare sectors in every track, defect management can be done in the same track. It does not require seek time overhead, a defected disk will show better performance than the original disk. Since our model does not consider disk on-board cache effect, there is a chance that the disk cache itself could be involved in the performance. Our next work will essentially be about the disk cache effect.

References

[1] N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch. A five-year study of file-system metadata. *In Proceedings of the 5th USENIX Conference on File and Storage Technologies*, pages 31–45, 2007.

[2] J. S. Bucy and G. R. Ganger. *The DiskSim Simulation Environment Version 3.0 Reference Manual*. School of Computer Science, Carnegie Mellon University, 2003.

[3] W. Davy. Method for eliminating file fragmentation and reducing average seek times in a magnetic disk media environment. *US 5808821*, September 15, 1998.

[4] A. Di Marco. The geometry of commodity hard-disks. Technical report, Technical Report DISI-TR-07-07, DISI-Universita di Genova, July, 2007.

[5] R. M. Duvall and J. M. Claar. Dense edit re-recording to reduce file fragmentation. *US 6182200*, January 30, 2001.

[6] L. Huang and T. Chiueh. Implementation of a rotation latency sensitive disk scheduler. *SUNY at Stony Brook Technical Report*, 2000.

[7] C. R. Lumb, J. Schindler, G. R. Ganger, D. F. Nagle, and E. Riedel. Towards higher disk head utilization: extracting free bandwidth from busy disk drives. *In Proceedings of the 4th conference on Symposium on OSDI*, pages 1–7, 2000.

[8] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *In Proceedings of Computer, IEEE*, 27(3):17–28, 1994.

[9] J. Schindler and G. R. Ganger. Automated disk drive characterization. *In Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 112–113, 2000.

[10] J. Schindler, J. L. Griffin, C. R. Lumb, and G. R. Ganger. Track-aligned extents: matching access patterns to disk drive characteristics. *In Proceedings of the Conference on File and Storage Technologies*, 2002.

[11] J. Schindler, S. W. Schlosser, M. Shao, A. Ailamaki, and G. R. Ganger. Atropos: A disk array volume manager for orchestrated use of disks. *In Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, March 2004.

[12] D. I. Shin, Y. J. Yu, and H. Y. Yeom. Shedding light in the black-box : Structural modeling of modern disk drives. *15th Annual Meeting of the IEEE International Symposium, MASCOTS*, 2007.

[13] F. J. Velez and L. M. Correia. Mobile broadband services: classification, characterization, anddeployment scenarios. *Communications Magazine, IEEE*, 40(4):142–150, 2002.

[14] Y. Won, H. Chang, J. Ryu, Y. Kim, and J. Shim. Intelligent storage: Cross-layer optimization for soft real-time workload. *ACM Transactions on Storage (TOS)*, 2(3):255–282, 2006.

[15] Y. Won, J. Kim, and W. Jung. Energy-aware disk scheduling for soft real-time i/o requests. *Multimedia Systems*, 13(5):409–428, 2008.

[16] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang. Modeling hard-disk power consumption. *In Proceedings of the 4th USENIX Conference on File and Storage Technology (FAST 03)*, pages 217–230, 2003.