Placement of Data in Multi-Zone Disk Drives*

Shahram Ghandeharizadeh, Douglas J. Ierardi, Dongho Kim, Roger Zimmermann

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

February 22, 1996

Abstract

This paper describes a placement technique for the layout of files across the surface of a multi-zone magnetic disk drive to maximize its bandwidth. It argues for a file system that monitors the frequency of access to the files and modifies the placement of files in order to respond to an evolving pattern of file accesses. Such a file system is particularly useful for repositories whose primary functionality is to disseminate information, such as the World-Wide-Web along with thousands of ftp sites that render documents, graphic pictures, images, audio clips, and full-motion video available on the Internet. Employing software techniques to enhance the disk performance is important because its mechanical nature has limited its hardware performance improvements at 7 to 10 percent a year.

1 Introduction

While microprocessor and networking technologies have been advancing at a rapid pace during the past decade, disk performance improvements have been negligible, resulting in the I/O bottleneck phenomenon. The speedup for microprocessor technology is typically quoted at a rate of 40 to 60 percent annually. While disk storage densities are improving at a high rate (from 60 to 80 percent annually), performance improvements have been rated at only 7 to 10 percent annually [RW94]. A technique employed by modern magnetic disk drives to improve their storage density is *zoning*. Zoning introduces a disk with regions that provide different transfer rates. Depending on the characteristics of a disk, the fastest zone might provide a transfer rate 60 to 80 percent higher than that of the slowest zone. In this paper, we study software techniques that improve the performance of

*This research was supported in part by the National Science Foundation under grants IRI-9203389, IRI-9258362 (NYI award), and CDA-9216321, and a Hewlett-Packard unrestricted cash/equipment gift.

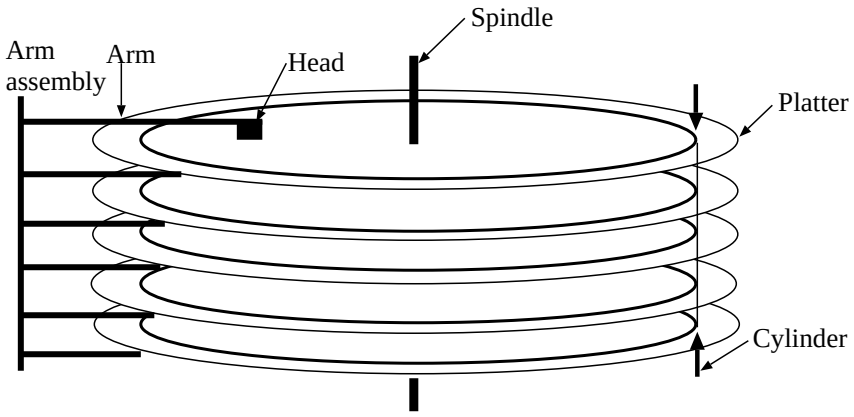

Figure 1: A disk drive

multi-zone disks by controlling the placement of data across the zones. To the best of our knowledge, no related work has appeared in the literature due to the recent introduction of multi-zone disk drives. We show that one can enhance the performance of a multi-zone disk by assigning the frequently accessed files to the zone with the highest transfer rate. We use the term *heat* [CABK88] to denote the frequency of access to a file.

The rest of this paper is organized as follows. Section 2 provides an overview of multi-zone magnetic disks and transfer rates from two commercial disk drives. Assuming that the size of a file system equals the storage capacity of a magnetic disk drive[1], Section 3 describes an optimal layout strategy that maximizes the performance of the disk drive. Section 4 describes an on-line reorganization technique the enables the file system system to respond to an evolving pattern of disk accesses. In addition, this section describes the design of a heat tracking module that estimates the frequency of access to each file. Section 5 quantifies the tradeoffs associated with the proposed techniques using a trace-driven simulation study. The obtained results demonstrate significant improvements using the proposed designs. Section 6 eliminates the assumption that the size of a file system equals the storage capacity of a disk and describes two alternative designs for data layout. Our conclusion and future research directions are contained in Section 7.

2 Multi-zone magnetic disks

A magnetic disk drive is a mechanical device, operated by its controlling electronics. The mechanical parts of the device consist of a stack of platters that rotate in unison on a central spindle (see [RW94] for details). Presently, a single disk contains one, two, or as many as sixteen platters (see Figure 1).

[1] This assumption is relaxed in Section 6.

Zone #	Size (MB)	Transfer Rate
0	324.0	3.40
1	112.0	3.17
2	76.0	3.04
3	77.0	2.92
4	71.0	2.78
5	145.0	2.54
6	109.0	2.27
7	89.0	2.02

(a) HP C2247 (Fast & Wide) disk

Zone #	Size (MB)	Transfer Rate (MBps)
0	139.7	4.17
1	67.0	4.08
2	45.7	4.02
3	45.3	3.97
4	43.9	3.88
5	42.6	3.83
6	41.3	3.74
7	56.4	3.60
8	39.2	3.60
9	37.5	3.50
10	51.5	3.36
11	35.0	3.31
12	33.8	3.22
13	46.5	3.13
14	31.8	3.07
15	30.7	2.99
16	41.8	2.85
17	28.0	2.76
18	27.6	2.76
19	37.1	2.62
20	25.0	2.53
21	33.5	2.43
22	25.4	2.33

(b) Seagate ST31200W (Fast & Wide) disk

Table 1: Two commercial disks and their zoning information

Each platter surface has an associated disk head responsible for reading and writing data. Each platter is set up to store data in a series of tracks. A single stack of tracks at a common distance from the spindle is termed a cylinder. To access the data stored in a track, the disk head must be positioned over it. The operation to reposition the head from the current track to the desired track is termed *seek*. Next, the disk must wait for the desired data to rotate under the head. This time is termed *rotational latency*. In a final step, the disk reads the referenced data. This time is termed *transfer time*. Its value depends on the size of the referenced data and the transfer rate of the disk.

To meet the demands for higher storage capacity, magnetic disk drive manufacturers have introduced disks with *zones*. A zone is a contiguous collection of disk cylinders whose tracks have the same storage capacity. Tracks are longer towards the outer portions of a disk platter as compared to the inner portions, hence, more data may be recorded in the outer tracks. While zoning increases the storage capacity of the disk, it produces a disk that does not have a single transfer rate. The multiple transfer rates are due to (1) the variable storage capacity of the tracks and (2) a fixed number of revolutions per second for the platters. Table 1 presents the storage capacity and transfer rate of zones for two commercial disk drives. Both are SCSI-2 fast and wide disks with a 1 gigabyte storage capacity.

3 Optimal zone layout

Assume that the disk consists of a sequence of blocks, partitioned into zones $Z_0, Z_1, \ldots, Z_k$, where the ith zone Z_i has a transfer rate of $\text{tfr}(Z_i)$ MB/s. The zones are ordered from fastest to slowest, so that $\text{tfr}(Z_i) > \text{tfr}(Z_{i+1})$ for $i = 0, 1, \ldots, k-1$.

To improve the average service time sustained by a disk–resident file, we propose to keep the "hottest" files — the most popular files, with the highest frequency of access — in the fastest zones. Specifically, we propose the following Optimal Zone (OZ) Layout for a given set of files, under the assumptions that (1) we have an accurate estimate for the probability of access of each file f_x, denoted $\text{heat}(f_x)$, and (2) the size of the set of files is no more than the capacity of the disk. First, order the files by their heats, from maximum to minimum. Then lay out the sequence of files contiguously, starting with zone Z_0 and proceeding progressively towards the innermost zone. Objects may span multiple zones, and no blocks are left unoccupied.

The OZ layout assigns the hottest files to the fastest zones. As one might expect, this layout in fact minimizes the expected (or average) service time for a file from the file system. As discussed in the previous section, the actual service time of any file includes time for seeking, rotational latency and transfer time. Since every access requires at least an initial seek and latency, we shall exclude this amount from our calculations and focus on the transfer time and additional latencies that arises from the fragmentation of files. However, in the proposed layout all files are stored contiguously, so none of these additional costs are observed.

Without loss of generality, assume that each file occupies only one zone. (If not, we can apply the following argument by considering each fragment of the file separately.) The actual transfer time for a file f_x in zone Z_i is $\text{size}(f_x)/\text{tfr}(Z_i)$; so the expected transfer time for a disk–resident file is

$$\sum_i \sum_{f_x \in Z_i} \text{heat}(f_x)\text{size}(f_x)/\text{tfr}(Z_i) \tag{1}$$

where the sum is computed over all zones Z_i and files contained therein.

Lemma 3.1

The layout proposed by OZ is optimal — that is, it minimizes the expected transfer rate, and thus the expected service time, of each request — when the given heat values provide an accurate estimate for the probability of access to any file.

Proof We claim that the quantity in (1) is minimized when the heat of each file in Z_i is greater or equal to the heat of each file in Z_{i+1}, for all i. This is proved by contradiction. Assume, for a

contradicion, that we in fact have an optimal file layout in which some file f_1 in zone Z_i that has a lower heat than file f_2 in zone Z_{i+1}. Then we could increase the average transfer rate by swapping blocks of f_1 and f_2, proving that this layout could not be optimal. Since an optimal layout must exist, the argument shows that it can only be achieved when the files are laid out so that the fastest zones have the hottest files. □

Section 5.2 describes a trace–driven simulation study that quantifies the performance improvements that can be achieved in this best–case scenario.

4 Dynamic on-line reorganization

It is not always the case that the static layout described in the previous section will be possible because it requires that: (1) all heat values to be known at the time that the layout is realized on the disk, (2) files are not added to or deleted from the file system, and (3) the heats remain constant. In many situations, one would expect both the content of the system, and/or the popularity of its files, to evolve over time. In this case, we envision a file system which (1) maintains statistical records of the popularity of its files, and (2) uses these computed heat values to reorganize data on the disk, migrating the hotter files to faster zones, in order to approximate the layout obtained using OZ.

4.1 Accumulating heat statistics

The file system must learn about the heat of files by gathering statistics from the issued requests. Our design employs the past pattern of access to estimate the identity of files referenced frequently in the future.

Its learning process is as follows. The file system maintains a queue of timestamps for every file, as well as an estimated heat value. All the queues are initially empty and the heat values are uniformly set[2] to $\frac{1}{n}$, where n is the total number of files. Upon the arrival of a request referencing file f_x, the current time is recorded in the queue of file f_x. Whenever the timestamp queue of file f_x becomes full, the heat value of that file is updated according to

$$\text{heat}_{new}(f_x) = (1-c) \times \frac{1}{\frac{1}{K} \times \sum_{i=1}^{K-1} t_{i+1} - t_i} + c \times \text{heat}_{old}(f_x) \qquad (2)$$

[2]This is the case in our simulation, and would be true in any situation where there is no reliable information available on the expected heat of files. However, if such information is available, it could be used in initializing the system.

where K is the length of the timestamp queue, c is a constant between 0 and 1, and t_x is one individual timestamp. After the update is completed, the queue of this file is flushed and new timestamps can be recorded.

This approach is similar to the concept of the *backward k-distance* used by the authors of [OOW93] in the LRU-k algorithm. The two schemes differ in three ways. First, the heat estimates are not based on the interval between the first and the last timestamp in the queue but are averages over all intervals. Second, the heat of a file f_x is only updated when the timestamp queue of f_x is full, therefore reducing overhead. And third, the previous heat value $\text{heat}_{old}(f_x)$ is taken into account when $\text{heat}_{new}(f_x)$ is calculated. The above measures balance the need for smoothing out short term fluctuations in the access pattern and guaranteeing responsiveness to longer term trends. The constants k and c could thus be used to tune the module's behavior to a particular application.

4.2 Migrating files among zones

In Section 3, the optimal layout for files was attained by ensuring that each was laid out contiguously along the surface of the disk, thus avoiding file fragmentation. However, since the file system may now rearrange files on the disk, there is the likelihood that files will become fragmented, causing the system to incur additional seeks and latencies. So while the system migrates hotter files to faster zones to improve their transfer time, it may also degrade the expected service time of files by fragmenting them.

To deal with these issues, we propose the following policies for reorganization:

1. On each access to a file f_x, if it is discovered that f_x's heat has changed, then it can be moved or *promoted* to a hotter zone. Any valid data residing in its target location will be swapped into f_x's current location. So in effect, we attempt to improve the performance of the system only by swapping files "pairwise", although the target of the swap may involve one or more files, or fragments of files. Moreover, the swap is initiated to improve the transfer time of a hotter file, never to intentionally degrade the transfer time of a colder one.

2. To deal with fragmentation that can arise, each file that is promoted to a faster zone must remain contiguous. In other words, if the hot file that is targeted for migration is currently contiguously laid out, then it remains continguous after its transfer to the new location. If the file was initially fragmented, then it is reassembled in a single contiguous sequence of blocks once moved. Note that the swap may result in the fragmentation of the colder files currently residing in the faster zone; yet the policy tries to insure the integrity of the more popular files

on the disk.

Given these policies, the system must still choose potential target locations for the migrating file, and for each potential target, must determine whether the proposed swap is worthwhile. These problems are resolved in the following way.

Initially, upon a reference to file f_x whose heat has changed, we determine the zone that it should occupy with OZ (say zone $Z_{optimal}$). Assume f_x resides in zone Z_x. If Z_x and $Z_{optimal}$ refer to the same zone then no migration is necessary. Otherwise, f_x is a candidate for migration from Z_x to $Z_{optimal}$. The system searches $Z_{optimal}$ for the coldest contiguous sequence of blocks that can hold f_x. To calculate the heat of a sequence of blocks, we note that each block of the disk that holds valid data inherits a heat from the file that occupies it, which is just the frequency at which that block is accessed from the device. The *total heat* of a sequence of blocks is then just the sum of the heat of all blocks in the sequence. So if the difference in total heat between f_x and the current target exceeds a preset threshold, the file is swapped into this location. This heat difference is proportional to the amount by which the total service time is expected to decrease if the swap is made, excluding the time for any additional seeks that have been introduced.[3] If the difference does not exceed the threshold, $Z_{optimal}$ is reset to $Z_{optimal+1}$ (the next slower zone) and this process is repeated. This process terminates when $Z_{optimal} = Z_x$.

A more detailed description of the algorithm is presented in Figure 2. The procedure is conservative, to the extent that it attempts to maintain the contiguity of the more popular files, and never tries to move a file beyond its placement in the optimal static schedule. (This is an attempt to promote files, while avoiding competition with other, presumably hotter, files that should reside in the faster outermost zones.) The procedure also relies on the configurable threshold parameter (denoted *THRESHOLD* in Figure 2) that is used to tune the sensitivity of the reorganization procedure: a swap is done only if the gain in expected service time exceeds this threshold. For smaller values of this parameter, files are migrated frequently in response to small variations in heat; as the parameter's value is increased, files will migrate only if their heat changes by a large amount, or if the files are themselves large. In this way, the parameter helps to quantify the tradeoff between the one-time cost of moving a file, and the expected reduction in service time that will be observed during future accesses.

[3] A more accurate measure of the change in service time can be computed by estimating the service time for both the current and proposed layouts of the files involved in the swap, weighting these by the files' heats, and determining the effect upon the overall average service time. This approach is discussed further in Section 5.

```
move(f)
{
  if the file already resides in the fastest zone
    then return;

  /* At this point all the heats should be current. */
  current_location = blocks currently occupied by f;
  current_zone = slowest zone in which any fragment of f resides;
  target_zone = best static placement of f for current heats;

  while (current_zone > target_zone) {
    target_location = a sequence of size(f) blocks of minimum total
      heat in target_zone;
    if (total_heat(current_location) - total_heat(target_location) > THRESHOLD) {
      swap the contents of current_location and target_location;
      break;
    }
    else
      --current_zone;
  }
  return;
}
```

Figure 2: Procedure for attempting to promote a file during on-line reorganization.

5 Trace driven evaluation

We employed a trace driven simulation study to investigate the tradeoffs associated with the proposed techniques. The trace is a sequence of file accesses to a ftp site at the University of California-Berkeley (s2k-ftp.CS.Berkeley.EDU). It was accumulated over a period of 17 months (from 11/14/93 to 4/13/95). The file system consists of 59,072 files and is 4.22 gigabytes in size. The trace contains neither the creation nor deletion date of a file, only the references to retrieve a file. It consists of 460,000 requests. A few files are hot and referenced more frequently than the others. On the average, a request retrieves 294 kilobytes of data.

5.1 Simulation model

For the purposes of this evaluation, we represented a disk drive as an array of blocks. The size of each block corresponded to a physical disk block and is 512 bytes long. A zone occupied a contiguous number of blocks. The simulator maintained the number of blocks that constituted a file and the location of each block. As we did not have the zone characteristics of a 4 Gigabyte disk drive, we employed the zone characteristics of the Seagate ST31200W disk for this evaluation. The storage capacity of this disk is 25% of the file system size. We increased the size of each zone by a factor of

four to store the file system.

We employed the analytical models of [GSZ95] to represent the seek operation and latency of a magnetic disk drive. These models estimate the characteristics of the Seagate ST31200W disk and are similar to those of [RW94, WGPW95]. The minimum and maximum seek times are 3 and 21.2 milliseconds, respectively. The average rotational latency time is 5.5 milliseconds. When a file transfer was initiated, the simulator reads its physically contiguous block in one transfer (eliminating seek and rotational latency for two or more physically adjacent blocks). A seek is incurred if a file is fragmented across the surface of the disk. The simulator computes the number of seeks incurred when reading a file. We assume all the references are directed to the file system and do not observe a memory hit.

5.2 Optimal zone layout

We analyzed the performance of the system with the OZ layout. For comparison purposes, we analyzed the worst case scenario that lays the files on the surface of the disk in the reverse order, assigning the frequently accessed files to the innermost zones. The average service time of the system is 83.5 milliseconds and 134 milliseconds with optimal and worst case assignment, respectively. The OZ layout provides a 38 percent improvement relative to the worst case.

The average service time of the disk drive based on a random assignment of files to the disk is estimated to be 100.4 msec. This number is computed based on the probability of a request referencing data stored in a zone (for zone Z_i, this probability is $\frac{size\ of\ Z_i}{disk\ capacity}$), the time required to retrieve the average number of bytes per request from zone Z_i ($\frac{average\ request\ size}{tfr(Z_i)}$), and the average seek and rotational delays. This average estimate was verified by running the simulator several times assuming a random assignment of files. OZ provides a 17 percent improvement as compared to this average service time.

5.3 Heat tracking

In order to quantify the impact and effectiveness of the configuration parameters K and c (Equation 2) of the heat tracking module we analyzed its performance by computing the root mean square (r.m.s.) deviation [Wal84] of the estimates computed by the on-line heat–tracking module from the

overall values computed over the duration of the run. That is, we consider the quality factor Q:

$$Q = \sqrt{\frac{\sum_{i=1}^{N}(E(f_i) - R(f_i))^2}{N}}$$

where N is the total number of files on the disk, $E(f_i)$ is the estimated heat value for file f_i computed using the tracking module, and $R(f_i)$ denotes the real heat value calculated as the fraction $\frac{number\ of\ accesses\ to\ f_i}{total\ number\ of\ accesses}$. Of course, this is not necessarily the only measure to use here. We are, in effect, tracking a moving target — the changing popularity of files. Nevertheless, this value can provide a good estimate of how much change there is among the heats of individual files throughout the run, and how responsive or unresponsive the heat–tracking module is to local fluctuations.

The deviation Q was computed after every 20,000 requests during the simulation of the trace. We investigated the following c values: 0.1, 0.25, 0.5, and 0.75. For each of the c values, the heat statistics queue length K was set to 5, 15, 25, and 50 for different experiments. The results indicate that for this particular trace, the best (i.e., smallest) overall deviation can be achieved with either $K = 50$ and $c = 0.5$ or $K = 25$ and $c = 0.75$, although in general the r.m.s. deviations are relatively consistent. A small c value that favors the more recent history of the frequency of access is more accurate at system startup time, however, its margin of error increases as the number of issued requests increase. Short heat queues (i.e., either $K = 5$ or $K = 15$) lead to more responsive heat tracking, however, they also increase the fluctuation in the estimated heat values, resulting in a higher margin of error.

5.4 On-line reorganization

We analyzed the on-line reorganization technique by assuming the worst-case scenario for the initial placement of files on the disk. Next the simulator was invoked to learn the heat of files (K=50, c=0.5 in Equation 2) and was allowed to migrate the frequently accessed files from the slower zones to the faster ones using the algorithm developed earlier. We tallied the average service time of the disk for each 20,000 requests. Finally, we computed the percentage improvement in service time with reorganization as compared to the worst-case layout with no reorganization. Figure 3 shows the obtained results. In this figure, every tick on the x-axis represents 20,000 requests (approximately 15 days of operation). The y-axis represents the percent improvement. For comparison purposes, this figure also shows the percent improvement[4] in service time with OZ relative to the worst case

[4]Assuming that S(layout) is a function that denotes the average service time of the system with a layout, say OZ, as compared to worst is computed at: $100 \times \frac{S(worst) - S(OZ)}{S(worst)}$. This number is always less than 100%.

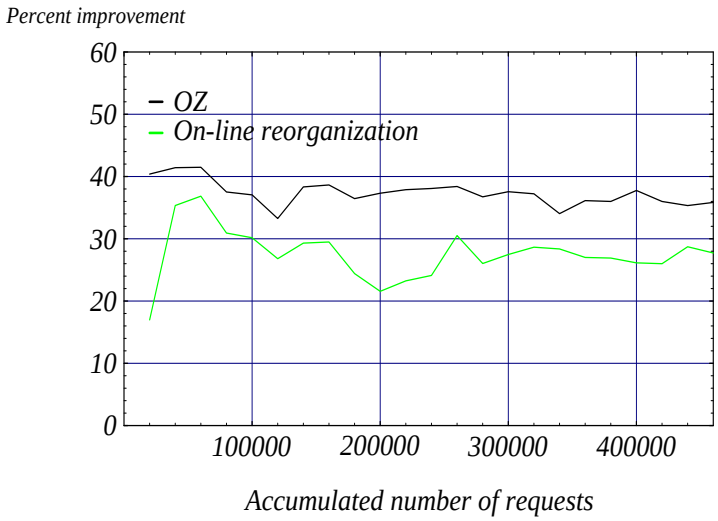

Figure 3: Percent improvement with on-line reorganization and optimal

layout with no reorganization.

During the first 20,000 requests, the reorganization module improves the average service time by 16 percent. This improvement increases to 30 percent during the next 20,000 requests because the heat tracking module provides a better estimate of file heats. The reorganization module approximates the percent improvement provided by optimal. It cannot provide a better improvement because it is approximating the heat of files while OZ has organized the files based on "perfect" heat statistics (that is, heat statistics corresponding to global averages across the entire trace). The amount of fragmentation attributed to the reorganization process is negligible. At the end of simulation, the reorganization process resulted in four hundred non-contiguous chunks due to fragmentation (as compared to zero at system startup).

6 Discussion

The simulation results of the previous section assumed that the size of the file system was about equal to the capacity of the disk, and thus that the files were packed into zones, leaving no free space[5]. In the case where the file system is smaller than the disk capacity, the system may choose between two alternatives when assigning files to the zones; it can either (1) try to fill completely each of the fastest zones, to take advantage of their higher transfer rate, or (2) distribute the files more evenly

[5]The proposed online reorganization procedure, however, made no assumptions in this regard.

across the disk, leaving free space in each of the zones. Filling up each of the fastest zones completely has the advantage of utilizing the whole space of these zones, and therefore maximizes the average observed transfer of the disk for its resident files. This can potentially give the greatest possible performance enhancement. However, this approach might suffer from the following limitations: greater difficulties during on-line compaction, file creation and update, which are more likely to lead to the fragmentation of files.

On the other hand, if we do not fill up each zone completely, there is the likelihood that online compaction, file creation and file update will be more efficient and effective, and that fragmentation can be minimized during these operations because of the available free space within zones. (By comparison, the Unix Fast File System [MJLF84] groups the cylinders of a hard disk drive as cylinder groups. It then tries to localize related files within a cylinder group, while also spreading unrelated files across cylinder groups in order to improve performance by minimizing seek time. To achieve this, it usually reserves 10 percent of the space as free space to do compactions.) In addition, it is likely that the initial stage of object promotions can be made cheaper, and that the possibility of multiple moves (both ping–ponging and domino effects) can be reduced. However, this strategy does not fully utilize the capacity of the fastest zones. Therefore it cannot expect to achieve the best possible performance.

It is likely that these tradeoffs can best be managed in an application dependent way. For example, the amount of free space reserved in each zone could be tuned to the projected stability of file system (*i.e.* how often files are created, deleted and updated, and how much variation in the heat of files is expected).

In the trace driven simulation, we also did not deal with the issue of file creation (since file creation times were not explicitly represented in the trace). Nevertheless, the creation of new files can be handled within the framework developed above. Specifically, if we are given a prospective heat for the newly created file, we can determine from this what its appropriate zone should be. If we have free space for the file in this zone, it is inserted there; otherwise, the reorganization algorithm is applied to this file. If this also fails, the file can then be placed into the zone with the most free space, which is most likely the inner-most or slowest zone. Updates can be handled in a similar manner.

7 Conclusion and Future Directions

This study describes an optimal placement technique for the assignment of files to the zones of a disk drive. This technique strives to assign the frequently accessed files to the fastest zones in order to maximize the disk bandwidth. In order to respond to evolving access patterns, we outlined an on-line reorganization process that monitors the frequency of access to the files and migrates the frequently accessed ones to the faster zones upon reference. Our trace driven simulation study demonstrates that: 1) the optimal layout improves the average service time of the system, 2) effective heat tracking modules can be constructed, and 3) the on-line reorganization process can approximate the percent improvement provided by the optimal organization. As discussed in Section 6, the reorganization process can be fine–tuned, based on the characteristics of a target application.

As part of our future research activity, we intend to analyze alternative multi-zone disk drives and traces obtained from other ftp sites. In addition, we intend to analyze the concept of striping using multi-zone disk drives. When a file is striped across multiple disks, its retrieval time is determined by the fragment assigned to the slowest zone of the participating disks. Is it important to control the assignment of different fragments of a file to the zones of available disks? Does this impact the size of a fragment? What is the impact of this on the parity block and its computation? Some of these issues are addressed in the context of heterogeneous disks [CRS95] and we intend to extend these designs to multi-disk environments consisting of multi-zone disk drives. We intend to investigate these design decisions along with the implementation details of the proposed ideas using the Unix operating system.

References

[CABK88] G. Copeland, W. Alexander, E. Boughter, and T. Keller. Data Placement in Bubba. In *Proceedings of ACM SIGMOD*, pages 100–110, 1988.

[CRS95] L. T. Chen, D. Rotem, and S. Seshadri. Declustering Databases on Heterogeneous Disk Systems. In *Proceedings of Very Large Databases*, 1995.

[GSZ95] S. Ghandeharizadeh, J. Stone, and R. Zimmermann. Techniques to Quantify SCSI-2 Disk Subsystem for Multimedia. USC Technical Report 95-610, University of Southern California, 1995.

[MJLF84] M. Mckusick, W. Joy, S. Leffler, and R. Fabry. A Fast File System for UNIX. *ACM Transactions on Computer Systems*, August 1984.

[OOW93] E. J. O'Neil, P. E. O'Neil, and G. Weikum. The LRU-K Page Replacement Algorithm for Database Disk Buffering. In *Proceedings of ACM SIGMOD*, pages 413–417, 1993.

[RW94] C. Ruemmler and J. Wilkes. An Introduction to Disk Drive Modeling. *IEEE Computer*, March 1994.

[Wal84] P.R. Wallace. *Mathematical Analysis of Physical Problems.* Dover, 1984.

[WGPW95] B. Worthington, G. R. Ganger, Y. N. Patt, and J. Wilkes. On-Line Extraction of SCSI Disk Parameters. In *Proceedings of ACM SIGMETRICS*, pages 146–156, 1995.