# Frequency-based Block Reordering

Jazmin Ortiz    Emily Dorsey    Elizabeth Krenkel    Geoff Kuenning

Harvey Mudd College

Undergraduate Senior jortiz@hmc.edu

Undergraduate Sophmore edorsey@hmc.edu

Undergraduate Junior ekrenkel@hmc.edu

Faculty Advisor geoff@cs.hmc.edu

## 1. Introduction

Processor speed is increasing at a faster rate than disk I/O speed, making I/O a bottleneck for overall performance. Because of this, an increase in I/O speed often leads to an improvement in the overall speed of a system. One of the main sources of delay for disk I/O is seek time. The goal of our research is to find a way to rearrange the blocks on disk to minimize the overall seek distance. By minimizing the distance which the disk head needs to travel, we believe that we can decrease overall response time.

Previous approaches to this problem include the sequence mining, smart disks, and disks aware of the context of specific I/O requests [4] [6] [7]. We chose to work on improving disk layout based on stored knowledge of the previous I/O stream and to not work with the higher-level file system.

## 2. Previous Works

Some previous attempts to minimize seek distance include finding semantic relations between blocks from the file system, identifying correlated blocks on the I/O stream level, and placing highly correlated blocks close together. Sivathanu et al. dynamically arranged blocks using smart disks, or disks aware of the context of specific I/O requests [6]. Though these studies produced promising results, the difficulties in implementation lead us to work solely on the disk level.

SLPMiner, a frequent pattern finding algorithm, proved useful for finding block correlations as well [5]. Li et al. used an algorithm in the same family as SLPMiner to reorganize and identify sequences of blocks [4]. However, their primary focus was on the identification of sequences for prefetching purposes, and they did not find any significant improvement by adding block re-organizing to their prefetching algorithm.

Another common approach for identifying correlated blocks of memory is to use a directed graph. This was used notably and effectively by Bhadkamkar et al., and decreased disk seek to between 50% and 6% of the baseline [1].

Though effective, the overhead from mining the relationship data between blocks of memory is very costly. The goal of our research is to minimize seek distance as much as possible while also minimizing the overhead cost in computation and memory. One notable previous approach with low overhead is the organ pipe arrangement, which places data so that the most frequently accessed block is in the center of the disk, with the second-most accessed block directly to the right, and the third-most to the left, and so on. Though this would be effective if block accesses were uncorrelated, in reality blocks with similar frequencies of access are often accessed together, and trying to read the entirety of a file in from beginning to end often requires going over the central ‘pipe’ many times. Organ-pipe arrangements have been shown to occasionally decrease performance, and so have largely been discarded [2] [3].

## 3. Design

Our goal is to minimize the distance the read-write head must travel to retrieve data on disk. To do that, we began by making several simplifying assumptions. First, we modeled the disk as a one-dimensional array. Second, we looked only at the absolute value of the distance between blocks, and did not consider the effect of the direction of spin of the disk. Third, when evaluating the effectiveness of the initial layout of the disk we did not consider the action of the disk-head scheduler and considered only the distance that would be traveled by the read-write head if it serviced IO requests as they come to disk. To reorder blocks on disk based on the order that they are serviced after the disk head scheduler has changed their sequence would presumably produce an order which is close to the order in which blocks are already laid out on the disk.

To evaluate the goodness of our reorganization we used trace files provided by the SNIA IOTTA website. Considering the disk as a one-dimensional array of memory addresses, we evaluated the total initial seek distance by summing the total distance between successive IO calls. We then rearranged blocks on our “disk” by swapping out their positions in our one-dimensional array. We evaluated the effectiveness of our final arrangement by again summing the total distance and comparing that to the initial distance traveled.

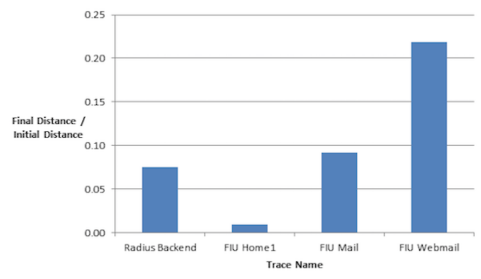

**Figure 1.** We saw significant improvement in seek distance in our simulation with foreknowledge of what blocks would be accessed. Smaller numbers indicate more improvement.

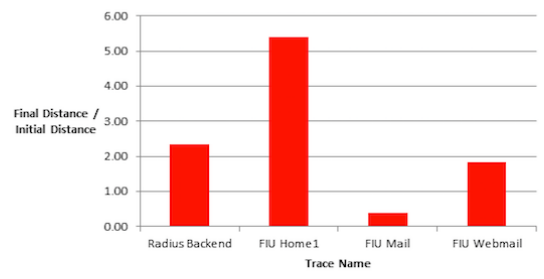

**Figure 2.** We had mostly negative results when our simulation ran without foreknowledge of what blocks would be accessed.

## 4. Preliminary Results

We intend to eventually look at more involved sequence-mining techniques and clustering algorithms, but we began by looking at a very basic approach: ordering blocks on the disk in the order of their frequency of access. Initially we looked at the frequencies produced from the entire trace, evaluated on the entire trace. This showed a surprising amount of improvement in terms of seek distance and was also inexpensive in terms of memory and computation compared to previous correlation identification approaches. When reordering blocks based on frequency, we were able to decrease the total seek distance traveled to an average of 10% of the initial seek distance, with the smallest improvement we saw reducing the seek distance to 22%. Figure 1 shows the improvement in seek distance after reordering relative to the initial seek distance.

We also evaluated the same traces in the more realistic scenario where we did not have foreknowledge of which blocks would be accessed. We used the first half of the traces to find the block frequencies and evaluated based on the second half of the trace. The results of this test can be seen in Figure 2. Our results from this were not nearly as good as our case with foreknowledge. We were only able to decrease seek distance in two of the traces that we tried it on. The other two became worse.

However, we believe that this problem could be attributable to two factors which we intend to address in future research. First, it is possible that the 'learning' period for our program was simply not long enough. This is supported by the fact that the trace which shows the most improvement, the FIU mail trace, is also longer than the others by two orders of magnitude. Second, we think that it is probable that the data from the trace files may be extremely periodic in nature. If that is the case, there is little reason that the first half of the trace should arrange data in the correct order for testing on the second half of the case. In previous attempts to dynamically reorganize data on disk, the periodic behavior of computer systems has been compensated for by programming the computer to collect and store data in different simulations at different times of day, and to rearrange itself based on those separate simulations again depending on the day of the week and the time of day [1].

## 5. Future Work

Our next steps will be to test our simulation without foreknowledge on longer traces and to implement a timing-based reordering of blocks. If that is successful, we would like to then try applying our block rearrangement to a real file system to test the actual performance impact.

## References

[1] Medha Bhadkamkar, Jorge Guerra, Luis Useche, Sam Burnett, Jason Liptak, Raju Rangaswami, and Vagelis Hristidis. BORG: Block-reorganization for self-optimizing storage systems. In *Proceedings of the Seventh USENIX Conference on File and Storage Technologies*, pages 183–196, 2009.

[2] Scott D. Carson and Paul F. Reynolds. Adaptive disk reorganization. Technical Report UMIACS-TR-89-4 and CS-TR-2178, Institute for Advanced Computer Studies, Department of Computer Science, University of Maryland, and Department of Computer Science, University of Virginia, January 1989.

[3] Windsor W. Hsu, Alan Jay Smith, and Honesty C. Young. Automatic improvement of locality in storage systems. Technical Report UCB/CSD-03-1264, University of California, Berkeley, July 2003.

[4] Zhenmin Li, Zhifeng Chen, Sudarshan M. Srinivasan, and Yuanyuan Zhou. C-Miner: Mining block correlations in storage systems. In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, 2004.

[5] Masakazu Seno and George Karypis. SLPMiner: An algorithm for finding frequent sequential patterns using length-decreasing support constraint. In *Proceedings of the Second IEEE Conference on Data Mining*, pages 418–425, 2002.

[6] Gopalan Sivathanu, Swaminathan Sundararaman, and Erez Zadok. Type safe disks. In *Proceedings of the Seventh USENIX Symposium on Operating Systems Design and Implementation*, 2006.

[7] Neeraja J. Yadwadkar, Chiranjib Bhattacharyya, K. Gopinath, Thirumale Niranjan, and Sai Susarla. Discovery of application workloads from network file traces. In *USENIX Conference on File and Storage Technologies*, February 2010.