TPL: A Data Layout Method for Reducing Rotational Latency of Modern Hard Disk Drive

Li Xiao, Tan Yu-An
Beijing Institute of Technology
Department of Computer Science and Technology
5 South Zhongguancun Street, Haidian District, Beijing 100081, P. R. China

Abstract

This paper proposes a data layout method for reducing the rotation delay of read access in Modern hard disk drive. TPL (Track Partition Layout) is a data layout that mainly bases on the detailed knowledge of the hard disk's physical geometry. By dividing one disk track into a set of partitions of sectors and duplicating data in these partitions, the rotational latency would be greatly reduced. And the method also contribute to minimizing the response time of disk request.

We have implemented this layout method in the widely used disk simulator – DiskSim by slightly revising the code in disk controller module, and have chosen four different types of disk to run the experiment. As the simulation shows, with a typical workload trace, the average rotational latency decreased to 70% when dividing a track into only two partitions. This number further drops to 42% when the number of partition reaches 32. Also, the average overall I/O system response time drops correspondingly from 83% to 65%.

1. Introduction

Modern disk drive contains a set of rotating platters which are stacked on a spindle and the read-write heads on sliders which are mounted onto arms. These heads are positioned over each surface of the platter for reading and writing data on the disk. Each platter is broken into tightly packed concentric tracks, which are further divided into sectors, the smallest unit a disk access request can address.

Each disk access basically involves two phases of mechanical activity: moving the heads into the track which contains the requested sector and waiting until that sector rotates under the read-write head, then the media transfer would begin. The latency caused in that two phases is often referred as seek latency and rotational latency.

Being the only mechanical component of the computer system, it is the latency and the low media transfer time caused by the fixed rotation speed that turns the hard disk drive into the bottle neck of the entire system performance. As the technology in advancing every year, the capacity of the disk is increasing dramatically, but the speed is not. For over 15 years, the drives have only spun up from 5400 rpm to 15000 rpm, while at the same time, the capacity has grown 1000 times, and the trend is not likely to stop.

Various methods have been proposed in the literature addressing the issue. These works range from a large number of disk scheduling algorithms [7] to carefully designed cache management strategy [2] and prefetching policy [3][4]. The aim is basically to minimize the I/O system response time or maximize the throughout, or to trade off in the scene of some specific application.

In the previous works, many efforts have been made to reduce the seek latency, this includes many of the early scheduling algorithm (e.g. SCAN, SSTF, etc.). But as the disk capacity increases every year, seek time now is not the only factor that dominate the disk request access time, rotational latency has become another main factor.

Some mechanism has already been proposed to deal with the rotational latency, for example, the SPTF scheduling algorithm. This algorithm rearranges the requests dispatched to the disk according to both seek latency and rotational latency. Other mechanism utilizes a disk's potential media bandwidth by filling rotational latency periods with useful media transfers [5]. But to achieve good result needs very detailed knowledge of the disk geometry, the algorithm used in the firmware and other physical parameters [8][6]. Besides, the actual latency is not reduced. This means these methods, while improving the throughout, cannot reduce the disk response time.

This paper proposes a method that can reduce the rotational latency in terms of redundancy. As we divide every track into several partitions, each partition in the same track will contain same set of logical block numbers (LBN), that is, the same data copies. So when disk head seeks to the

978-0-7695-3507-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CSIE.2009.430

track that contains the requests LBN, it could choose the nearest partition which contains the LBN. In this way, TPL greatly reduces the rotational latency of read access, especially in the workload of intensive random read access.

Although it seems that this method trade time with space, and makes only a fraction of disk's capacity usable, fortunately we will see that this method could take the advantages of the disk's free space and dynamically adjust the numbers of partition. All these can be done in block driver of the operating system and transparent to the user.

2. Track Partition Layout

When a disk access request is sent to the hard disk, it specifies the sector number where access begins and whether it is a read request or a write request. This sector number is called Logical Block Number (LBN). This is because the physical geometry of the disk is transparent to the BIOS and the Operating System. To the operating system, the entire disk is a linear address space, whereas inside the disk drive, it uses a tuple of cylinder, head, sector to designate the physical sector, this is referred as the Physical Block Number (LBN).

To complete a disk request, the controller in the disk drive needs to translate the the LBN to the PBN. Then according to the cylinder, head, and sector number in PBN, the disk head designated is moved to the right cylinder, and waits the sector requested to rotate under the head. Figure 1 shows a greatly simplified LBN layout that illustrates the mapping between LBN and PBN. It should be pointed out that the actual mapping would be far more complicated, since the defect sectors and spare sectors will cause remappings in the regular linear pattern.

The numbers in figure 1 indicates the LBN of the sector. When the disk is doing a sequential reading, after reading a whole track, the head seeks to the adjacent track, the skew ensures that when the head reaches the adjacent track, the sector of the following LBN would be right under the head. For example, in figure 1, when the head finishes reading block 24 in the outer track, it then seeks to the middle track. at the same time, the disk continues to rotate clockwise, when the head reaches the middle track, the skew makes block 25 right under the head.

However, in the case of random reading, the rotational latency is inevitable. Take figure 1 for example again, when the head finishes reading block 24, if the next request is block 48, when the head reaches the middle track, block 48 has already passed over the head, so the head will have to wait the disk to rotate a whole cycle that block 48 could pass by again. Even if the disk rotates at 10000 rpm, the latency would be close to 6 ms. The expectation rotational latency becomes 3 ms with a uniform distribution of random access.

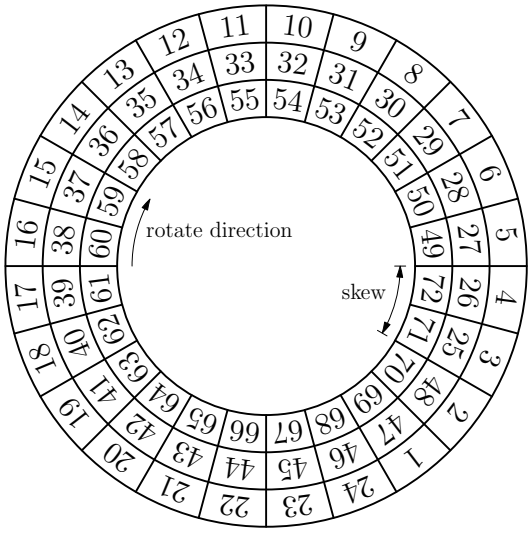

Figure 1. Normal LBN layout

To reduce this rotational latency, our method is to rearrange and duplicate the LBNs on the disk. Given that a disk track contains a number of N sectors, for simplicity, we assume there is no defect and spare sectors in the track. that is to say, the track also contains N sequential LBNs, say from b_1 to b_N. Upon that, we divide the track into m segments of sequential sectors, each segment has $\lfloor N/m \rfloor$ sectors. All the segments will be mapped to the same set of LBNs, b_1 to $b_{\lfloor N/m \rfloor}$, LBN $b_{\lfloor N/m \rfloor+1}$ to b_N will be left to the next adjacent track. Sectors mapped to the same LBN also hold the same block of data. This layout is called the Track Partition Layout (TPL), and the segment of sectors is called partition. Figure 2 describes this layout when dividing a track into two partitions.

From figure 2 we can see that every track contains two copies of the same data blocks, which distributes uniformly around the track, shown in white and grey blocks in the figure. In this case, the rotational latency could be greatly reduced. For example, when disk head finishes reading the LBN 12 in the outer track, say the one in grey block, assuming the next request is on LBN 24, the head needs not to wait the disk to rotate a full cycle. The reason is that although the LBN 24 in white block has already passed over, after only a half cycle, the LBN 24 in grey block will be right under the head.

Suppose each sector in the track has the same probability of read access, that is $p_{sec} = \frac{1}{N}$, the average distance from current disk head position to the LBN specified by the next disk request will be $\frac{N}{2}$. Thus if we configure TPL with m partitions, the expectation of the distance will be $\frac{N}{2m}$, this is the case of random access pattern which best takes the advantage of TPL.

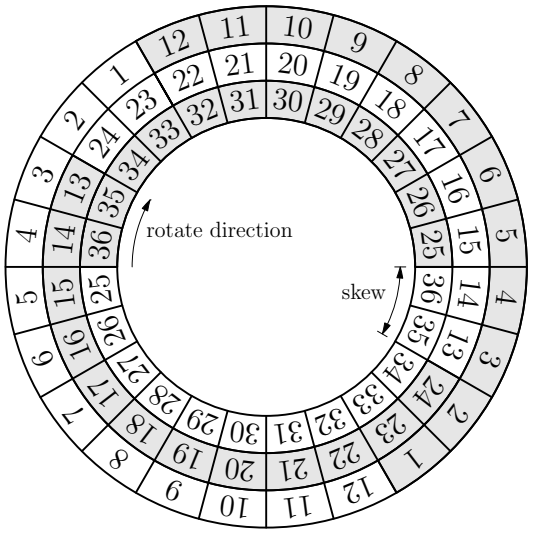

Figure 2. Track Partition Layout, when the number of partitions is 2.

Certain disadvantages exist in TPL. First, if all the tracks in the disk uses the fixed m partitions, then the number of available LBNs drops to $\frac{N}{m}$, where N donates the original LBN address size. This means only a part of the disk's capacity can be used as data storage, the rest of the capacity is used as data redundancy. Second, long sequential access will span over more tracks, since in TPL the track shrinks to $\frac{1}{m}$ of its original size. Skews still connects the consecutive LBNs in adjacent tranks, but more tracks introduce more seek latency. Third, write request takes even more time in synchronizing data in the duplications. We will talk about these weak points later.

3. Simulation

To test our method, we use Disksim 4.0 to do the simulation.DiskSim is an efficient, accurate and highly-configurable disk system simulator. It includes modules that simulate disks, intermediate controllers, buses, device drivers, request schedulers, disk block caches, and disk array data organizations. In particular, the disk drive module simulates modern disk drives in great detail and has been carefully validated against several production disks [1]. Among these disks, we choose 3 different types of disk to use in our simulation. And we have used DiskSim to extract and validate the physical parameters from another disk that has a large capacity, this disk is also used in the simulation. These disks are listed in table 1.

With a little revision in the disk controller module of DiskSim, the disk controller can use the TPL approach to translate the LBN to the PBN. That is, when a disk request is sent to the disk controller, if it is a read request, the controller then computes the distance from the current disk head position to all the sectors that maps to the request's LBN. In each partitions in that track, there is a sector mapping to the request's LBN. Then the controller finds the sector which is nearest to the disk head in one of the partitions. Finally the request's LBN is translated into that sector's PBN. In this way the disk head waits the shortest time before the sector with designate LBN to come.

It should be pointed out that in the simulation, we have made necessary simplification to the TPL and the algorithm used to translate from the LBN to the PBN. As we described in previous section, TPL makes the disk LBN address space shrink, but in the simulation we mainly care about the rotational latency and the response time, so we use the original LBN address space. This makes it possible to use disk's original trace workload which includes significant local and sequential access. Based on the same reason, disk access ignores partitions after the media transfer starts, so that the access will span no more tracks than it does when there is no TPL. Finally, the write request is implemented as normal. This means write request's LBN is translated to its original PBN and no synchronization is involved.

4. Results

The three hard disks tested in the simulation, atlas10k, HP_c3323a and ibm18es, use the trace workload, while Cheetah10k7 uses the synthetic workload. There are 10000 requests generated in the synthetic workload. Probability of read access is set to 0.66. Result in figure 3, 4, 5 shows the average rotational latency when partition number goes from 1 up to 32. Partition number 1 indicates the result without TPL.

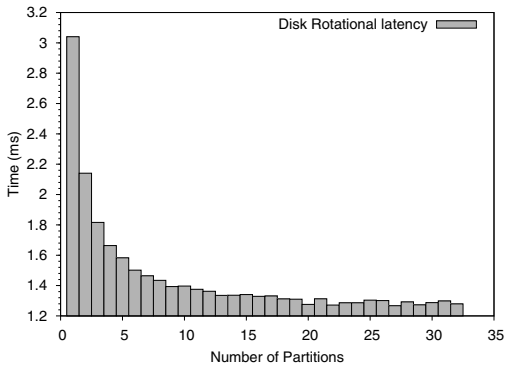

Figure 3. Experimental result of atlas10k

From the figure we can see that all 3 disks show the same behavior, irrelevant to their capacity and rotation speed. The

	atlas10k	ibm18es	hp_c3323a	Cheetah10k7
Number of data surfaces	6	5	7	20
Number of cylinders	10042	11474	2982	46875
Block count	17938986	17916240	2056008	585937499
Single cylinder seek time	1.24500	1.086000	0.200000	0.548400
Full strobe seek time	10.82800	12.742000	2.300000	7.108200
Add. write settling delay	0.48700	0.126000	0.070000	0.0
Head switch time	0.17600	0.062000	0.650000	0.018000
Rotation speed (in rpms)	10025	7200	5396	10022
Percent error in rpms	0.0	0.0	0.0	0.026097

Table 1. Parameters extracted from the disks used in simulation

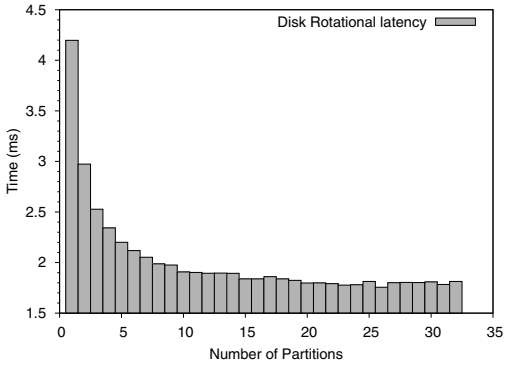

Figure 4. Experimental result of ibm18es

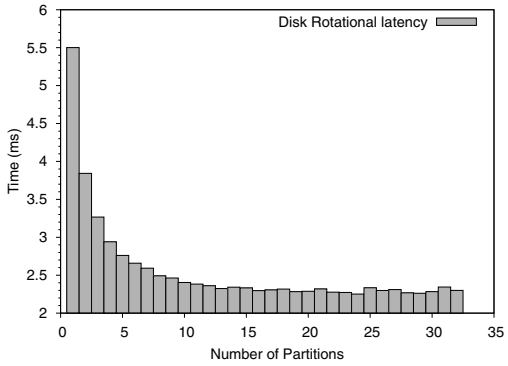

Figure 5. Experimental result of hp_c3323a

latency drops rapidly, especially when the partition number is small. So even with only two partitions, the latency decrease about 30%. When the number of partitions becomes large, the decreasing slows down, this is due to the sequential and local access in the trace workload. These accesses span over tracks so that it leaves no room for rotational latency. The write request in the trace also makes TPL less effective.

Result in figure 6 shows the Overall I/O System Response time when partition number goes from 1 up to 32. As we can imagine, the low rotational latency contributes to response time, this figure verifies that the average system Response time decreases just like the response latency. However, because of the same reason for the rotational latency, decreasing of the System Response time slows as the number of partitions becomes large. In addition, since the response time consists of not only rotational time, but also the seek time, media access and data transfer time, so time indicated in this figure is larger than in previous figures.

Finally, in figure 7, we can see TPL's effect in the ideal situation. In the simulation, disk Cheetah10k7 uses totally random workload, in which Local request distances is normally distributed, and sizes of the request is exponentially distributed. As shown in the figure, the average response time even declines to 33%. If there is no write request in the workload, this number will be even low.

5. Conclusions

In this paper, we describe a new data layout method, the Track partition layout. TPL places data duplications of a set of sectors within a track, intending to reduce the rotational latency of read request. We have tested 4 disks of different types in simulation. The experimental results show that TPL have significant effect on typical workload. The rotational latency has greatly reduced even with a small number of partitions. In the situation of random workload, the effect was even remarkable. This result suggests that, TPL is perfectly suitable in the kind of application in which random read access dominates the disk access and the response time is critical.

Finally, we'd like to point out some possible improvement and further work concerning the disadvantages we have described in the previous section.

Since TPL is basically a redundant layout method which trades space for time, part of the disk's capacity becomes unavailable. This can be solved by making use of the free

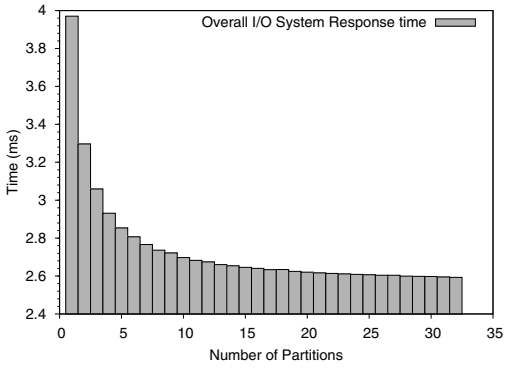

Figure 6. Experimental result of atlas10k

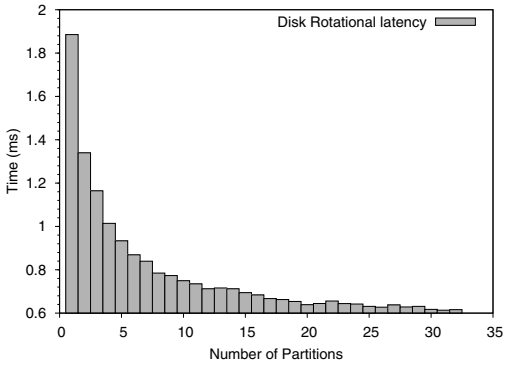

Figure 7. Experimental result of Cheetah10k7

space in the disk. Each track in an empty disk could be divided into a large number of partitions, for example, 32 partitions. Along with the data writing to the disk, we can dynamically adjust the number of the partitions in each track independently. That is, when $1/32$ of the disk's capacity is filled up, we decrease the number of partitions in some tracks, and this allows for new data to be stored to these tracks, and when adequate data is deleted in a track, we increase the number of partitions in that track and duplicate data in partitions. We can go even further to impose lower and upper bound to the partition number for each track, so we can have precise control over the speed and space we need.

Another disadvantage is the write access. We have pointed out that the write request needs to synchronize every duplication in the track, so it will spend more time. But we should also notice that the time spent in a track is always the time that disk head takes to rotate a full cycle, and this is irrelevant to how many partitions in the track. On the other hand, the synchronization can be incorporated into a carefully designed write-back cache algorithm, that is, the disk delays the synchronization to the disk idle period, or to a time that the disk head is passing by. In this way the duplications are incrementally updated, so part of the duplications may not be available at a given time. Although doing so will eliminate the effect that TPL offers, the synchronization overheads will be greatly reduced.

Further work includes implementation of TPL and evaluation of the possible improvement. We believe that it will be essential to implement TPL in a operating system, like Linux, to see the outcome in real system. This may need writing a kernel driver in the block device layer and maintaining a large table in memory that records partition settings of each track. Also, analyzing the effect and cost of all these improvements will be necessary. It could help us find out to what extent the TPL' applicability will be.

References

[1] J. S. Bucy and G. R. Ganger. The DiskSim simulation environment version 3.0 reference manual. Technical Report CMU-CS-03-102, Carnegie Mellon University, School of Computer Science, Jan. 2003.

[2] X. Ding, S. Jiang, and F. Chen. A buffer cache management scheme exploiting both temporal and spatial localities. *TOS*, 3(2), 2007.

[3] X. Ding, S. Jiang, F. Chen, K. Davis, and X. Zhang. Diskseen: Exploiting disk layout and access history to enhance I/O prefetch. In *USENIX Annual Technical Conference*, pages 261–274. USENIX, 2007.

[4] C. Li, K. Shen, and A. E. Papathanasiou. Competitive prefetching for concurrent sequential I/O. In P. Ferreira, T. R. Gross, and L. Veiga, editors, *EuroSys*, pages 189–202. ACM, 2007.

[5] C. R. Lumb, J. Schindler, G. R. Ganger, D. Nagle, and E. Riedel. Towards higher disk head utilization: Extracting "free" bandwidth from busy disk drives. In *OSDI*, pages 87–102, 2000.

[6] Raju Rangaswami Zoran Dimitrijevic, David Watson and A. Acharya. Diskbench: User-level disk feature extraction tool. Technical Report ucsb_cs:TR-2004-18, University of California, Santa Barbara, Computer Science, June 6 2004.

[7] B. L. Worthington, G. R. Ganger, and Y. N. Patt. Scheduling for modern disk drives and non-random workloads. Technical report, Aug. 25 1994.

[8] B. L. Worthington, G. R. Ganger, Y. N. Patt, and J. Wilkes. On-line extraction of SCSI disk drive parameters. Technical Report HPL-97-02, Hewlett Packard Laboratories, Jan. 15 1997.