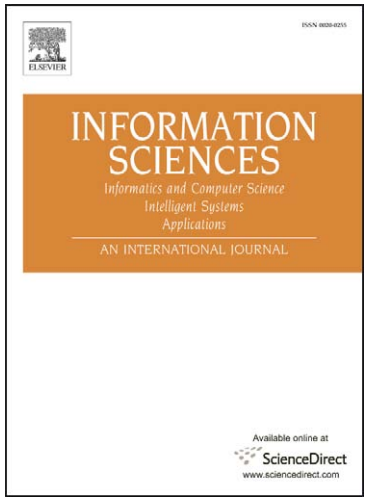
Information Sciences, Volume 179, Issue 14, Including Special Section - Linguistic Decision Making - Tools and Applications, 27 June 2009, Pages 2494-2511

Accepted Manuscript



Exploiting the Performance Gains of Modern Disk Drives by Enhancing Data Locality

Yuhui Deng

PII: S0020-0255(09)00073-5
DOI: 10.1016/j.ins.2009.02.002
Reference: INS 8252

To appear in: *Information Sciences*

Received Date: 19 September 2007
Revised Date: 22 January 2009
Accepted Date: 1 February 2009

Please cite this article as: Y. Deng, Exploiting the Performance Gains of Modern Disk Drives by Enhancing Data Locality, *Information Sciences* (2009), doi: 10.1016/j.ins.2009.02.002

Exploiting the Performance Gains of Modern Disk Drives by Enhancing Data Locality

Yuhui Deng,

Cambridge-Cranfield High Performance Computing Facilities,
Cranfield University Campus, Bedfordshire MK430AL, United Kingdom

Email: deng_derek@emc.com; yuhuid@hotmail.com

Due to the widening performance gap between RAM and disk drives, a large number of I/O optimization methods have been proposed and designed to alleviate the impact of this gap. One of the most effective approaches of improving disk access performance is enhancing data locality. This is because the method could increase the hit ratio of disk cache and reduce the seek time and rotational latency. Disk drives have experienced dramatic development since the first disk drive was announced in 1956. This paper investigates some important characteristics of modern disk drives. Based on the characteristics and the observation that data access on disk drives is highly skewed, the frequently accessed data blocks and the correlated data blocks are clustered into objects and moved to the outer zones of a modern disk drive. The idea attempts to enhance spatial locality, improve the efficiency of aggressive sequential prefetch, and take advantage of Zoned Bit Recording (ZBR). An experimental simulation is employed to investigate the performance gains generated by the enhanced data locality. The performance gains are analyzed by breaking down the disk access time into seek time, rotational latency, data transfer time, and hit ratio of the disk cache. Experimental results provide useful insights into the performance behaviours of a modern disk drive with enhanced data locality.

Key words: Disk drive; Data Locality; Data access pattern; Block correlation; Data migration; Performance

1. Introduction

The storage hierarchy in current computer architectures is designed to take advantage of data access locality to improve overall performance. Each level of the hierarchy has higher speed, lower latency, and smaller size than lower levels. For decades, the hierarchical arrangement has suffered from significant bandwidth and latency gaps among processor, RAM, and disk drive [27,32,38]. The performance gap between processor and RAM has been alleviated by fast cache memories. However, the performance gap of RAM to disk drive has been widened to 6 orders of magnitude in 2000 and will continue to widen by about 50% per year [38].

Since the first disk drive was announced in 1956, disk drives have grown by over six orders of magnitude in density and over four orders in performance [29]. Over the last decade, areal density, track density and linear density have achieved 100%, 50%, and 30% growth respectively. Revolutions Per Minute (RPM) has been increased from 3600 in 1981 to 15000 in 2000. Due to the significant growth in

both the linear density and RPM, Internal Data Rate (IDR) has been growing at an exponential rate of 40% each year over the past 15 years [11]. Unfortunately, the basic mechanical architecture in disk drive has not changed too much. Slowed by the mechanical delays, disk access time was improved only about 8% per year [12]. Therefore, the disk I/O subsystem is repeatedly identified as a major bottleneck to system performance in many computing systems. The widening gap will be more serious when disk drives reache its physical limits due to the super paramagnetic effect.

The increasing performance gap between RAM and disk drive has long been a primary obstacle to improve overall system performance. To alleviate the impact of this widening gap, a lot of research efforts have been invested to improve disk access time. We only mention some of them which are related to our work in this paper. Ruemmler and Wilkes [36] employed disk shuffling to move frequently accessed data into the centre of a disk drive and organize the data into an organ pipe to reduce mean seek distances. They constructed a repeatable simulation environment across a range of workloads and disk drive types for comparing different shuffling algorithms. Their research indicated that the benefits are small to moderate, but are likely to be much larger with file systems that do not do a good initial data placement. Akyürek and Salem [1] presented an adaptive technique to copy a small number of frequently referenced disk blocks from their original locations to a reserved space near the middle of the disk. Their experiments showed that seek times are reduced 30% to 85% (depending on workloads) by adaptively rearranging about 3% of the data on the disk drive. FS2 [16] dynamically places multiple copies of data in file system's free blocks according to the disk access patterns observed at runtime to reduce the head positioning latencies. Because one or more replicas can be accessed in addition to their original data block, choosing the nearest replica that provides the fastest access can significantly improve disk I/O performance.

The above methods can substantially reduce seek time or rotational latency. However, those approaches may not be effective on modern disk drives due to evolutional disk drive technologies. First of all, disk access time mainly consists of seek time, rotational latency and data transfer time. Due to the advance of Voice Coil Motors (VCM) electric drive and the increasing track density, long distance seeks of modern disk drives may not involve enormously more overhead than short ones. This results in a significant proportion decrease of seek time in disk access time. Secondly, due to geometric features, outer tracks on disk platters are much larger than the inner tracks. Modern disk drives employ a technique called Zoned Bit Recording (ZBR) to take advantage of its geometric features to increase disk capacity by varying the number of sectors per track with the distance from the spindle [30]. This characteristic results in much higher data transfer rate of outer zones than that of inner zones. Finally, the organ pipe is formed by placing the most frequently accessed cylinder in the middle of the disk drive, the next most frequently accessed cylinders on either side of the middle cylinder, and so on. This arrangement is provably optimal for independent disk accesses [36]. However, block correlations are common semantic patterns in storage systems, so the organ pipe arrangement could destroy the original block correlations [21]. The combination of these three reasons significantly counteracts the performance improvement of data reorganization.

This paper explores the performance gains of data reorganization based on a modern disk drive. Some new characteristics of modern disk drives, which are related to data reorganization, are reviewed. Based on the characteristics, blocks, which are correlated to the frequently accessed block, are clustered into objects, and the objects are moved to the outer zones of the disk drive. The aggressive sequential prefetch of the modern disk drive is enhanced and the block correlations remain due to the frequency based objects. Disk access time is broken into four basic components: seek time, rotational latency, data transfer time, and hit ratio of disk cache, each one is analyzed separately to determine its actual performance gains. Experimental results provide useful insights into the performance behaviour of block reorganization of a

modern disk drive.

The remainder of this paper is organized as follows. An overview of modern disk drives is introduced in Section 2. Section 3 describes some important features of workload. The motivations of this paper are depicted in section 4. Section 5 illustrates how to implement the block reorganization. The simulation environment and experimental validation are depicted in section 6. Section 7 concludes the paper with remarks on main contributions and indications.

2. Modern Disk Drive Overview

Disk access time T_{access} is mainly composed of seek time T_{seek}, rotational latency T_{rotate} and data transfer time $T_{transfer}$. The seek time measures the time for the disk head to move to a specified track. When the disk head arrives at the required track, the time spent on rotating the required sector to appear underneath the disk head is called rotational latency. The data transfer time is the amount of data divided by data transfer rate. T_{access} is expressed as follows:

$$T_{access} = T_{seek} + T_{rotate} + T_{transfer} \quad (1)$$

2.1. Seek Time

Disk head is driven by a VCM to move over the recording surface to seek a target sector. In order to reduce the heat dissipation of the VCM, the temperature of the coil in the VCM is controlled by selecting a fixed maximum current for the seek distances which exceed a threshold. This threshold is typically 35% of a full stroke [40]. For a long seek distance which exceeds the threshold, the current in the coil reaches the maximum value when the disk head reaches a nominal maximum velocity. At the end of an acceleration period, the current is removed from the coil, which incurs a coast period that maintains a nominal maximum velocity. Then, the fixed maximum current is applied to the coil in an opposite direction to decelerate the disk head. When the disk head reaches the target track, a procedure is triggered to verify the current position. The time cost for the disk head to settle at the end of a seek is called settling time. Therefore, a seek time is composed of an acceleration time t_{acc}, a coast period t_{coast}, a deceleration time t_{dec}, and a head settling time t_{settle}. The acceleration time and deceleration time are proportional to the square root of the seek distance. The cost period is linear in the seek distance. For a short seek (e.g. single cylinder seek), the disk arm accelerates and decelerates without reaching the nominal maximum velocity.

According to the above discussion, for the seek distances which are shorter than the threshold, the disk heads will never reach the nominal maximal velocity. This indicates that in this scenario there is no cost period no matter how fast a VCM is. The average seek time is generally taken to be the average time needed to seek between two random blocks on the disks which is normally called average seek distance $D_{average}$. The $D_{average}$ for a large number of random seeks is equal to a seek across 1/3 of the data zone which is shorter than the threshold. Therefore, the coast time of average seeks is zero. Consequently, we have an acceleration phase followed immediately by a deceleration phase [18], which can be described as $D_{average} = (1/2) \times a_{acc} \times t_{acc}^2 + (1/2) \times a_{dec} \times t_{dec}^2$, where a_{acc} is the acceleration

which is equal to the deceleration a_{dec}, and the acceleration time t_{acc} is equal to the deceleration time t_{dec}.

We assume that $a_{acc} = a_{dec} = a$, then we have:

$$t_{acc} = t_{dec} = \sqrt{\frac{D_{average}}{a}} \tag{2}$$

The average seek time T_{seek} is computed with the following equation:

$$T_{seek} = 2 \times \sqrt{\frac{D_{average}}{a}} + t_{settle} \tag{3}$$

For random small requests, seek time is a major component of disk access time, because the settling time dominates the overall short seeks and the settling time has remained largely constant [1]. However, over the last decade, areal density has achieved 100% growth. This has resulted in 50% growth of track density measured in Tracks Per Inch (TPI), and 30% growth of linear density measured in Bits Per Inch (BPI) [11]. Due to the increased BPI, there are more sectors on a track, which means more sectors in a cylinder if the number of disk heads is not changed [31]. Within a certain range of data, a bigger cylinder may impact the seek time in two ways: The first one is increasing the probability of reducing the number of seeks. When dealing with a certain amount of data, having a bigger cylinder raises the probability that the next data request will be satisfied in the current cylinder, thus avoiding a seek completely. The second one is reducing seek distance. If the size of each cylinder is increased, then an equal amount of data will occupy fewer cylinders compared with before. As a result, the seek distance is decreased. Both impacts result in shorter seek time in terms of equation (3).

Lumb et al [24] investigated the impact of seek time, rotational latency, and data transfer time that add up to 100% of the disk head utilization for five modern disk drives which were sold on the market from 1996 to 1999. Their investigation indicated that the faster seek of the Cheetah 18LP (average seek time 5.2ms), relative to the Cheetah 9LP and Cheetah 4LP which have average seek time of 5.4ms and 7.7ms respectively, resulted in lower seek components. The results also showed that as the request size of random workload increased, larger request size yielded larger media transfer component, reduced the seek and rotational latency components by amortizing larger transfer over each positioning step.

2.2. Rotational latency

Rotational latency depends on the RPM and the number of sectors that must pass underneath the disk head. Traditionally, when the disk head arrives at a target track, it must wait for the disk platters to rotate until it reaches the first sector of the request before it begins to transfer data. The amount of time it takes for the required sector to appear underneath the disk head is called rotational latency. If the disk head settles above a sector which is one of the required sectors but not the first one, it will incur almost one revolution to reach the first sector.

Zero-latency access, which is a new feature of modern disk drives, can start transferring data when the disk head is positioned above any of the sectors in a request. If multiple contiguous sectors are required to be read, the disk head can read the sectors from the media into its buffer in any order with zero-latency access support. The sectors in the buffer are assembled in ascending Logical Block Number (LBN) order and sent to the host. If exactly one track is required, the disk head can begin reading data as soon as the

seek is completed. It involves no rotational latency because all sectors on the track are needed. The same concept applies to writes with a reverse procedure which moves the data from host memory to the disk cache before it can be written onto the media [39]. Therefore, the rotational latency decreases with the growth of the useful blocks in a track.

2.3. Data Transfer Time and Zoned Bit Recording

Data transfer time is the amount of data divided by data transfer rate. This consists of two parts. The first part is an external data rate adopted to measure the transfer rate between memory and disk cache. The second part is employed to measure the transfer rate between disk cache and disk storage media, this part is called IDR. Due to the mechanical components in disk drives, the IDR is much lower than the external data rate. Generally, the IDR is employed to measure the data transfer rate of disk drives because it is raw transfer rate. The IDR depends on the combination of BPI and RPM. The BPI indicates how many bits can be stored on a track, which in turn determines the number of sectors on a track. The data transfer time can be calculated with following equation:

$$T_{transfer} = \frac{N_{request}}{N_{track}} \times \frac{60}{RPM} \tag{4}$$

where N_{track} denotes the number of sectors on a track, and $N_{request}$ is the data length of a request measured in sectors.

Due to the geometric features, outer tracks on disk platters are much larger than inner tracks. Modern disk drives employ a technique called ZBR, sometimes called Zoned Constant Angular Velocity (ZCAV), to take advantage of the geometric features to maximize disk capacity by varying the number of sectors per track with the distance from the spindle [30]. This technique groups tracks into zones based on their distance from the spindle, and assigns each zone a different number of sectors per track. Outer zones are longer and contain more sectors than the shorter inner zones. The ratio of the sectors of the outmost zone to that of the innermost zone ranges from 1.43 to 1.58, according to the disk characteristics illustrated in [24]. In terms of equation (4), for the same amount of data, the ZBR results in a much smaller data transfer time of the outer zones than that of the inner zones.

2.4. Disk Cache

Almost all modern disk drives employ a small amount of on-board cache (RAM) to speed up access to data on the disk drives [17]. Today's SDRAM has access time ranging from 7 to 10 nanoseconds. We assume that 512Byte data (one sector) needs to be accessed in a SDRAM. The SDRAM has 64 bit chip configuration and 10 nanoseconds access time. The data access overhead is about 6.4×10^{-4} milliseconds. This overhead is only 0.032% of the latest Hitachi Ultrastar 15K which has an average access time of 2 milliseconds and an RPM of 15000. Because accessing data from cache is much faster than accessing from disk media, disk cache can significantly improve performance by avoiding slow mechanical latencies if the data access is satisfied from the disk cache (cache hit). Disk cache today can hold more data due to the increasing cache size (Ultrastar 15K has 16MB disk cache), resulting in a higher hit ratio. As the hit ratio grows, the benefits of reducing the seek time, rotational latency, and data transfer time decrease.

Hsu and Smith [12] reported that disk cache in the megabyte range is sufficient. For a very large disk

cache, its hit ratio continues to slightly improve as the cache size is increased beyond 4% of the storage used. This indicates that if the disk cache size grows beyond a certain threshold, the increased cache only achieves a limited contribution to the hit ratio, which is not cost-effective.

Disk cache works on the premise that the data in the cache will be reused often by temporarily holding data, thus reducing the number of physical access to the disk media [7]. To achieve this goal, caches exploit the principles of spatial and temporal locality of reference. The spatial locality implies that if a block is referenced, then nearby blocks will also soon be accessed. The temporal locality implies that a referenced block will tend to be referenced again in the near future. As the data locality improves, the hit ratio of disk cache grows. Of all the I/O optimizations that increase the efficiency of I/Os, reducing the number of physical disk I/Os by increasing the hit ratio of disk cache is the most effective method to improve disk I/O performance.

3. Data Access Locality

Data locality is a measure of how well data can be selected, retrieved, compactly stored, and reused for subsequent accesses. In general, there are two basic types of data locality: temporal, and spatial. The temporal locality denotes that a data is accessed at one point in time will be accessed in the near future. The temporal locality relies on the access patterns of different applications and can therefore change dynamically. The spatial locality defines that the probability of accessing a data is higher if a data near it was just accessed. Unlike the temporal locality, the spatial locality is inherent in the data managed by a storage system. It is relatively more stable and does not depend on applications, but rather on data organizations.

3.1. Data Access Pattern and Locality

Data locality is a property of both the access patterns of applications and data organization. Even though reshaping access patterns can be employed to improve temporal locality [3], it is difficult to modify the access patterns of those existing applications. Some methods have been proposed in the pattern recognition community to preserve the locality when the original data is projected into a lower dimensional feature space [22].

Many common access patterns approximate a Zipf-like distribution indicating that a few blocks are frequently accessed, and others much less often [43]. Staelin and Garcia-Molina [41] observed that there was a very high locality of reference on very large file systems (on the order of one tera byte). Some files in the file system have a much higher skew of access than others. The skew of disk I/O access is often referred to as 80/20 rule of thumb, or in more extreme cases, 90/10 Rule. The 80/20 rule indicates that twenty percent of storage resources receive eighty percent of I/O accesses, while the other eighty percent of resources serve the remainder twenty percent I/O accesses. Furthermore, the percentages are applied recursively. For example, twenty percent of the twenty percent storage resources serve eighty percent of the eighty percent I/O accesses [8, 41].

The skew access patterns bring opportunities for block reorganization or migration. Generally, the twenty percent of the storage resources which receive eighty percent of I/O accesses are distributed across the whole disk. If the twenty percent data blocks residing in the storage resources are packed together, the spatial locality would be enhanced and the frequently accessed blocks could remain in the disk cache longer.

3.2. Block Correlations and Aggressive Prefetch

Block correlations commonly exist in storage systems. Two or more blocks are correlated if they are linked together semantically. A lot of algorithms can be used to extract the correlations [13, 20, 21]. For example, C-Miner [21] employs a data mining technique called frequent sequence mining to discover block correlations in storage systems. These methods are very useful to exploit complex block correlations or extreme long-range dependence.

Riska and Riedel [34] reported that seek distances in some traces exhibit extreme long-range dependence. However, simple block correlations (e.g. spatial locality) are common patterns in storage systems, because most of the correlated blocks are usually accessed very close to each other though it may not be true for large files [35]. Fortunately, many studies indicate that the files in a file server are small files. Baker [2] reported that 80% of file accesses in file servers are less than 10KB. Nine years later, Roselli et al. [35] found that small files still comprised a large number of file accesses even though the number of accesses to large files had increased since the study in [2]. Riska and Riedel [34] measured the disk drive workloads in systems representing enterprise, desktop, and consumer electronics environments. They found that the common request size is 4KB across all traces, except the video streaming and game console which issue 128KB requests. Tanenbaum et al.[42] investigated the file size distribution on UNIX systems in 1984 and 2005, respectively. They reported that the files which were smaller than 8KB was 84.97% in 1984, and 69.96% in 2005. Also that 99.18% of files in 1984 and 90.84% of files in 2005 are smaller than 64K, respectively.

Block correlations generally depend on how the file system above organizes the disk blocks. A lot of effort has been invested to optimally organize disk blocks, thus improving the access performance of small files. Fast File System (FFS) [28] determines the location of the last allocated block of its file and attempts to allocate the next contiguous disk block when a new block is allocated. When blocks of a file are clustered, multiple block transfers can be used to read/write the file, therefore, reducing the number of disk I/O and disk access latency. Co-located Fast File System (C-FFS) [10] adjacently clusters the data blocks of multiple small files especially the small files in the same directory and moves to and from the disk as a unit. C-FFS attempts to allocate a new block of a small file into an existing unit associated with the same directory. Reiserfs [33] uses balanced trees to store data and metadata. For extremely small files, the entire file's data can be stored physically near the file's metadata, so that both can be retrieved together with little or no disk seeking time. Log-structured File System (LFS) [23] delays, remaps, and clusters all data blocks into large, contiguous regions called segments on disks. LFS only writes large chunks to the disk, which exploits disk bandwidth for small files, metadata, and large files. Deng et al. [6] suggested clustering the small files as the size of the product of one cylinder size and disk number in a network attached system.

The above works illustrate that most of modern file systems tend to place correlated disk blocks close to each other and cluster blocks to reduce the number of disk I/Os and disk access latency. Such behaviours again confirm that simple locality is an inherent characteristic of disk drive workloads [37]. Based on the observation that a block is usually semantically correlated to its neighbour blocks especially for small files (For example, if a file's blocks are allocated in a disk drive consecutively, these blocks are correlated to each other. Therefore, in some workloads, these blocks are likely accessed one after another.), most of the modern disk drivers adopt aggressive sequential prefetch which takes advantage of the spatial locality to improve disk I/O performance.

4. Motivations

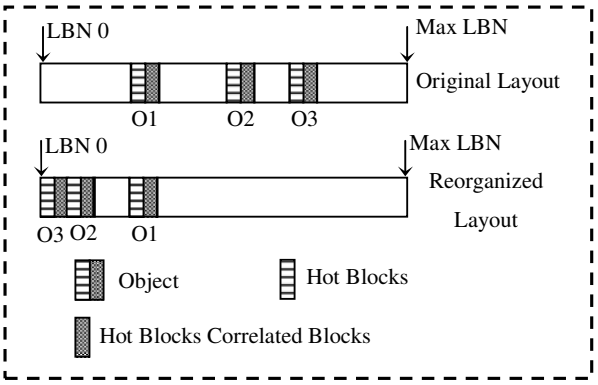


Fig. 1 Disk layout comparison with and without block reorganization

Due to the highly skewed access patterns, if the frequently accessed blocks are distributed across the whole disk drive, long seek distances from each other may be involved. Most modern disk drives employ aggressive sequential prefetch to exploit spatial locality and improve I/O access performance, but the effect of the prefetch could be significantly reduced in the above scenario, because a large number of blocks with low access frequency could be prefetched to disk cache, thus reducing the hit ratio of the disk cache. Moving the frequently accessed data blocks to a small area to enhance the data locality can significantly improve the effectiveness of the aggressive prefetch. Hsu et al.[14] recommended placing the reorganized data blocks roughly at a 24-33% radial distance offset from the outer edge. The reason is that most of the disk reads are either eliminated due to the more effective sequential prefetch, or can be satisfied from the reorganized area. They believed that the remaining disk reads tend to be uniformly distributed. Therefore, placing the reorganized data blocks at the centre of the disk can reduce seek time significantly [1, 36]. However, due to the ZBR technique, the data transfer rate of the outmost zone is much higher than that of the innermost zone (ranging from 43% to 58% in [24]). Please note that the disk drives in [24] are about ten years old. Because of the increasing magnetic recording density, the ratio at present is much higher than that. As discussed in section 3.1, most of the data accesses are highly skewed. For a skew of 90/10, 90% of the data accesses go to the hot areas. In the 90% data accesses, if we assume that 30% of the data accesses are absorbed by disk cache, the remaining 60% references have to access disk drive. If the hot data blocks are placed in the out zones, the 60% references can take full advantage of the ZBR. Even though the remaining 10% infrequent data accesses could distribute uniformly (It is arguable). On the contrary, if the hot data blocks are placed at the centre of the disk, 60% of data accesses will lose the benefits of ZBR, even though the placement can leverage the 10% infrequent data accesses to reduce the seek time to a certain degree. Therefore, we believe that the benefits achieved by putting the hot data blocks in the outer zones are bigger than placing the data blocks in the centre of a disk drive.

Our method is based on the observation that only a small portion of blocks are accessed frequently. We divided a drive disk into a fast band and a slow band. The fast band is used to store the frequently accessed data on outer cylinders to take advantage of the ZBR. The size of fast band is determined as 10% of the disk drive capacity in terms of the 90/10 ruler of thumb. Because most of the frequently accessed blocks are packed into the fast band, the spatial locality is enhanced. Due to the observation that a block is usually correlated to its neighbor blocks as discussed in section 3.2, a migration unit called object which packs the frequently accessed blocks and the correlated neighbour blocks together is adopted to maintain

the block correlations. We clustered multiple adjacent blocks into one object and moved objects rather than cylinders, files or blocks to enhance the aggressive prefetch of modern disk drives.

For decades past, the most common storage interfaces (SCSI and IDE/ATA), which expose storage capacity as a linear array of fixed-sized blocks to file systems, mainly consist of simple read and write commands. Data access for read and write is specified by a LBN and a data block length. Disk firmware is responsible for translating LBN to physical location (C/H/S). Fig. 1 depicts the two disk layouts with and without block reorganization. In the upper figure, three objects with different access frequency are distributed across the disk drive which is organized in terms of LBN. Each object consists of frequently accessed blocks (hot blocks in Fig. 1) and the correlated neighbour blocks. The bottom figure shows the disk layout after object 2 and object 3 are migrated to the outer zones of the disk drive. Object 1 remains in the original location even if it has the highest access frequency, because its location is in the fast band. The object 3 has higher access frequency than that of object 2, and therefore the object 3 is moved to the outermost zone first, followed by the object 2. The upper figure illustrates that due to the long inter-object distance between object 1 and the other objects, long seek distance would be incurred if the data accesses to the hot blocks contained in the objects are interleaved, and the aggressive prefetch in modern disk drives could prefetch a large number of useless blocks which will not be accessed for a long time to the disk cache, thus decreasing the hit ratio of the disk cache. The bottom figure depicts a new disk layout after block migration. The seek distances among object 1, object 2 and object 3 are reduced significantly.

Based on the above block arrangement, if the fast band receives most of the data accesses, we expect the following benefits:

(1) The disk head lingers over the fast band most of the time and the seek time is significantly reduced.

(2) Zero-latency access of modern disk drives could achieve more benefit because the probability that more useful blocks are accessed within one revolution is increased. Therefore, the rotational latency could be reduced.

(3) The data transfer time is decreased because most of the frequently accessed blocks are clustered in the fast band to utilize the ZBR.

(4) The spatial locality is enhanced because most of the frequently accessed blocks are packed in a relatively narrow area.

(5) Due to the enhanced spatial locality, the blocks of fast band could reside in the disk cache much longer, thus increasing the hit ratio.

(6)The number of physical disk I/Os is decreased due to the increased hit ratio.

(7)The block correlations are maintained because the migration unit packs the frequently accessed blocks and the correlated neighbour blocks together.

5. Implementation

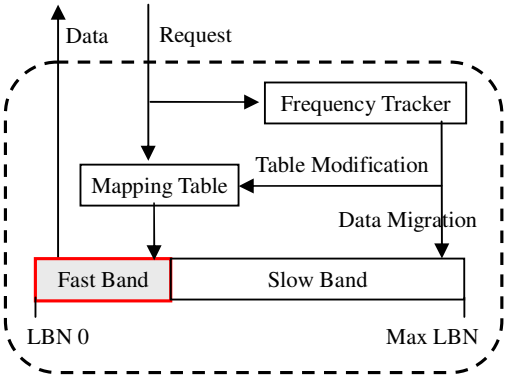


Fig. 2 A schematic of the block reorganization

Our Implementation is composed of a frequency tracker, a mapping table, and a data migration mechanism. The three major components are illustrated in Fig. 2. The frequency tracker monitors the stream of I/O requests. Periodically, it produces or updates a mapping table which contains a list of frequently accessed objects ordered by frequency. A newly incoming request is compared against the mapping table. The request will be redirected to a new location if the request is hit in the table. Otherwise, it goes to the original location. The data migration is triggered periodically in terms of the migration mechanism. The frequently accessed blocks and the correlated blocks will be moved to the fast band to enhance the spatial locality and take advantage of the ZBR.

5.1. Frequency Tracker

A data structure consisting of original object location, migrated object location, object access frequency and unoccupied block number in the object is employed to track the object access frequency and construct the mapping table. The data structure takes 16 bytes per object. For a 9.1GB disk drive which has 17938986 blocks (the disk we will use in the simulation) with an object size of 32 blocks, it takes (17938986/32)*16=8969493Bytes=8.55MB storage capacity to track the whole disk drive. By analogy, a 100 GB modern disk needs about 90 MB storage space to track all objects. Although the storage overhead is much smaller than the disk itself, the data structures should be maintained in memory so that they can be accessed with little overhead. Therefore, the storage capacity occupied by the data structures is important and should be kept as small as possible. A more space-efficient alternative is to maintain a small group of data structures for objects that have been recently accessed frequently. According to the skewed access pattern (10% storage resources receive 90% I/O accesses), the maximum number of objects which should be tracked is 10% of the whole objects. It is very easy to calculate that a 100GB disk drive needs about 9MB memory capacity to track the frequency of the most frequently accessed objects. This is acceptable for a modern computer system.

We used two fixed length Least Recently Used (LRU) lists including a hot list and a recent list to identify the most frequently and recently accessed objects. When the system receives a request, the corresponding object will be recorded on the recent list. If the object on the recent list is accessed again in a short period, the object will be promoted to the hot list. If the promoted object is already on the hot list, the object will be moved to the head of the hot list. If the hot list is full, the last object on the hot list will be degraded to the recent list. If the recent list is full, the last object on the recent list will be discarded, and the fields of object location including the original location and the migrated location will be recorded on a

mapping table. The objects on the hot list are the most frequently accessed objects over a period. However, the sequence of the objects is not sorted in terms of the frequency. The most recently accessed object is always on the head of the hot list, the next recently accessed object will follow the previous object. Therefore, compared with the frequency counting method, our method can further enhance the spatial locality which indicates that the probability of accessing a data unit is higher if a data unit near it was just accessed. The method is very effective in our experiment.

When the data migration mechanism is triggered, a process runing in background examines the hot list. If the objects on the hot list are stored in the slow band, the objects will be moved to the fast band. If the objects already exist in the fast band, the object locations will be kept unchanged. When the fast band runs out of its 90% capacity, the objects which are not on the hot list will be migrated to the slow band. The fields of original object location and migrated object location in the data structure will be employed to record the objects location and construct the mapping table.

5.2. Object Mapping Table

Data access for read and write is specified by a LBN (Logical Block Number) and a data block length. Disk firmware is responsible for translating LBN to physical location (i.e. cylinder, head, and sector). This high-level interface has enabled great portability, interoperability, and flexibility for storage devices and their vendors [9]. However, the narrow storage interface between file systems and storage hides details from both sides. Though both sides have made considerable advancement independently, the interface has limited opportunities for whole system performance improvement due to lacking effective cooperation.

A mapping table is employed to augment the interface in our implementation, because the mapping table contains some hints of disk access patterns coming from the above file system. The mapping table contains a list of frequently accessed objects, their original locations, and the locations after data moving. Newly incoming requests are compared against the list and redirected to the new location if the requested object resides there.

5.3. Data Migration Mechanism

Data migration could have significant impact on the overall I/O performance. The key point of a data migration is to complete the data moving process in the shortest possible time with minimal performance impact on the foreground applications, while guaranteeing QoS. It involves when and what blocks should be migrated to where.

Because workloads tend to be bursty, there are enough idle periods for the system to analyze and reorganize the data blocks (e.g. perform the data migration daily) [1, 12, 14, 15]. Hsu and Smith [14] reported that there is a lot of time during which the storage system is relatively idle. They also proved that infrequent (daily to weekly) block reorganization is sufficient to realize most of the benefit. Their experiments confirm that the data migration takes only a small fraction of the idle time available between reorganizations. On the other hand, some intelligent algorithms which can eliminate or alleviate the impact of data migration have been proposed and developed. Aqueduct [25] uses a control-theoretical approach to statistically guarantee a bound on the amount of impact on foreground work during a data migration, while still accomplishing the data migration in a time period as short as possible. Lumb et al [26] proposed a free block scheduling to replace a disk drive's rotational latency with useful background media transfers, potentially allowing the background disk I/O to occur with no impact on foreground service times. The two methods are supposed to further alleviate or eliminate the performance penalty of data migration. Because

this paper is exploring the performance gains of block reorganization, not the data migration methodology, the reader is referred to [25, 26] for a comprehensive understanding of data migration methodologies.

6. Experimental Validation

Table 1 Disk characteristics of Quantum Atlas 10K

Size	9100 MB
Disk cache size	2MB
Cylinders	10024
Number of Zones	24
Sectors per track of outermost zone	334
Sectors per track of innermost zone	229
Rotation speed (RPM)	10025
Single cylinder seek time	1.24500
Full strobe seek time	10.82800
Head switch time	0.17600

A real implementation of the comprehensive and complicated system would be difficult and take an extremely long time. Trace driven simulation is a principal approach to evaluate the effectiveness of our proposed design, because it is much easier to change parameters and configurations in comparison with a real implementation. The trace driven simulation is a form of event driven simulation in which the events are taken from a real system that operates under conditions similar to the ones being simulated. By using a simulator and reference traces, we can evaluate the system in different environments and under a variety of workloads.

DiskSim [4] is an efficient, accurate, highly configurable, and trace-driven disk system simulator. To explore the performance gains of block reorganization based on modern disks, we enhanced DiskSim to measure the proposed method. Several experimentally validated disk models are distributed with Disksim. The experimental results reported in this paper were generated by using the validated Quantum Atlas 10K disk model. The disk drive is divided into 24 Zones in terms of the number of sectors per track, head skew, cylinder skew and number of spare sectors. The detailed disk characteristics are summarized in Table 1.

Two important metrics normally employed to measure I/O performance are throughput and average response time. Throughput is the maximum number of I/Os that can be satisfied in a given period by the system. Measuring the throughput with trace driven simulation is difficult because the workloads recorded in the trace are constant. Average response time includes both the time needed to serve the I/O request and the time spent on waiting or queuing for service. In this paper, we measure the average response time and break down the disk access time into four major components including seek time, rotational latency, data transfer time, and the hit ration of disk cache, and analyze each component separately to determine its actual performance gains.

6.1. Evaluating the impact of object size

In order to explore the impact of the object size, we employed a real trace ST1 which extracts 50,000 requests from the cello99 trace [5] and modifies some fields in terms of the format requirements of the Disksim. The access type (read or write), request address, request size, and request arrival time are kept

unchanged to maintain the block correlations and data access pattern. The average request size of ST1 is 7.2 KB. 65% requests of ST1 are read accesses. The skew of ST1 is 74/10. It means that 74% I/Os in ST1 concentrate on 10% disk space.

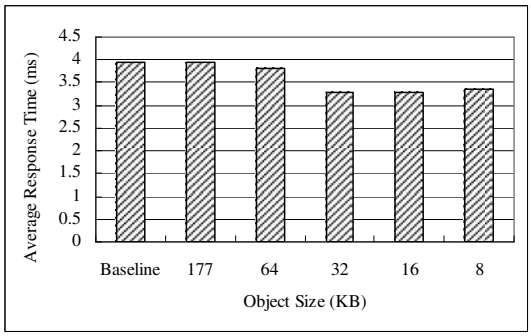


Fig. 3 Average response time with trace ST1

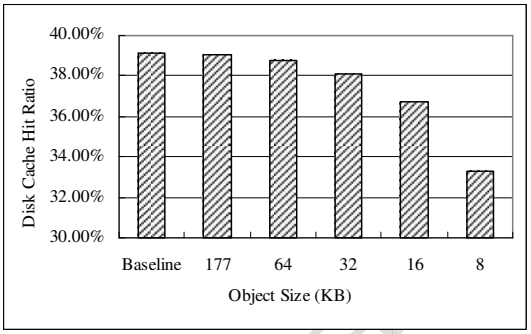


Fig. 4 Hit ratio of disk cache with trace ST1

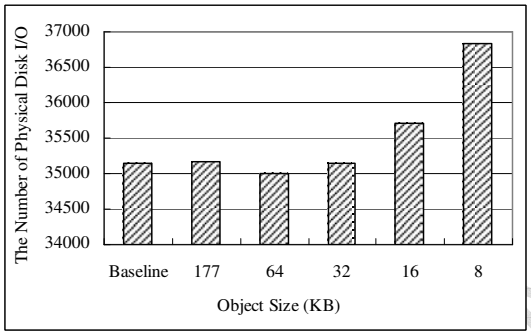


Fig. 5 Number of physical disk I/Os with trace ST1

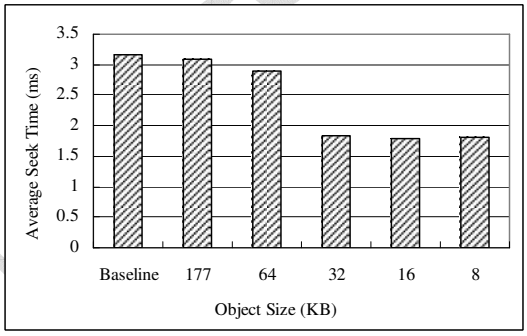


Fig. 6 Average seek time with trace ST1

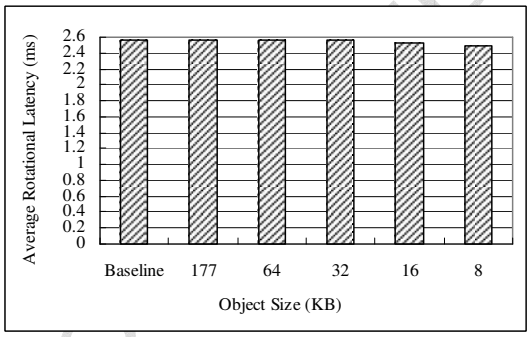


Fig. 7 Average rotational latency with trace ST1

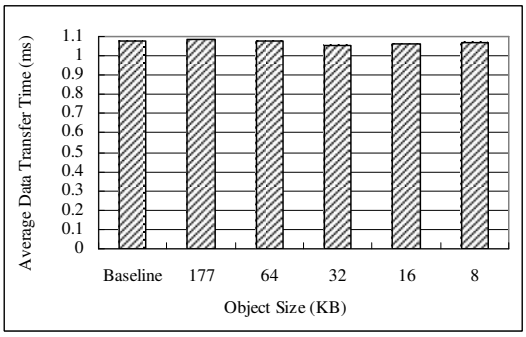


Fig. 8 Average data transfer time with trace ST1

We used ST1 to do the first round of measurements. A set of object sizes ranging from 177KB to 8KB were employed to investigate the optimal block correlations size by measuring the average response time incurred by the block reorganization. 177KB is the size of one segment of the disk cache extracted from the Quantum Atlas 10K disk model. The leftmost bar in each figure from Fig. 3 to Fig. 8 denotes the performance of the baseline system which does not employ any block reorganization.

Fig. 3 shows the average response time of the system which uses different object size to migrate blocks against the baseline system. It illustrates that the performance improvement ranges from 5% to 17.2% due to the block reorganization. According to the performance improvement, the optimal object

size is 32KB. We believe that the object size maintains the block correlations of the baseline system very well because it achieves the highest performance improvement.

Fig. 4 illustrates the hit ratio of disk cache. Before the test, we expected to see a significant increase of the hit ratio because of two reasons. The first reason is that the spatial locality is enhanced by gathering frequently accessed blocks. The second reason is that the prefetch could get more useful data which could stay in the cache much longer due to the high access frequency. However, as the object size decreases, the hit ratio also starts to decrease gradually and decrease acutely when the size reaches 16KB. The number of physical disk I/Os shown in Fig. 5 varies slightly when the object size is changed from 177KB to 32KB, whereas it has a sharp increase when the object size reaches 16KB. We believe that the object size which is smaller than 32KB could break the block correlations, accordingly incur more physical disk I/Os and decrease the hit ratio. Another reason is that the disk cache of the employed disk model in our simulation is only 2MB. The small disk cache can not take full advantage of the enhanced data locality. The results in Fig. 5 are consistent with that in Fig. 4.

Fig. 6 depicts that the average seek time is significantly reduced when the object size is changed to 32KB. The seek time is reduced 42.5% when the object size reaches 16KB. This validates our expectation that the disk head lingers over the fast band most of the time, and the seek time is reduced due to the reduced seek distance. A basic requirement is a relatively small object size which can cluster the frequently accessed blocks compactly.

Fig. 7 shows that the rotational latency achieves a slight performance improvement (less than 2.8%) when the object size varies. Fig. 8 illustrates that the data transfer time starts to decrease when the object size is changed to 64KB. The data transfer time achieves the highest 2.5% performance improvement when the object size is 32KB. These two measurements are not expected. According to equation (4), larger request size yields more performance gains at the data transfer time. Because the average request size of ST1 is 7.2 KB, we decided to do the second round of test with another trace by varying the request size. The object size in the second round of test is determined as 32KB because it maintains the basic block correlations.

6.2 Evaluating the impact of request size

Synthetic traces have the advantage of isolating specific behaviours which are not clearly expressed in the real world traces. The trace characteristics can be varied as much as possible in order to cover a wide range of different workloads. Therefore, we generated a synthetic trace named ST2 to investigate the impact of the request size. ST2 consists of 25,000 requests which also were extracted from cello99 [5]. In contrast to the ST1, we changed the request size of ST2 to explore the impact on the system performance. As discussed in section 3.2, the biggest request size in the disk drive workloads is 128KB. Therefore, we investigated the request size of 8KB, 16KB, 32KB, 64KB, and 128KB, respectively. We believe bigger request sizes can obtain more benefit from the ZBR [19]. The skew of ST2 is 87/10 which implies that 87% I/Os are accumulated in 10% disk space.

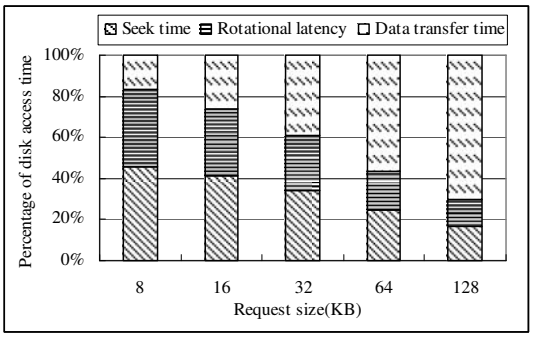


Fig. 9 Breakdown of disk access time with trace ST2

Table 2 Constitutions of disk access time with different request size

	8KB	16KB	32KB	64KB	128KB
Seek time	45.59%	40.97%	34.28%	24.77%	16.91%
Rotational latency	37.95%	33.16%	26.84%	18.72%	12.49%
Data transfer time	16.46%	25.87%	38.88%	56.51%	70.60%

Fig. 9 lists a breakdown of disk access time produced by ST2 with different request sizes. The test results are based on the 10025 RPM Quantum Atlas 10K disk. It shows that with the increase of the average request size, the portion of seek time and rotational latency are decreased, whereas the portion of data transfer time is increased. This is reasonable because larger request size involves more data transfer time. Table 2 shows when the average request size is changed, the percentages of each component including seek time, rotational latency, and data transfer time which contributes to the disk access time.

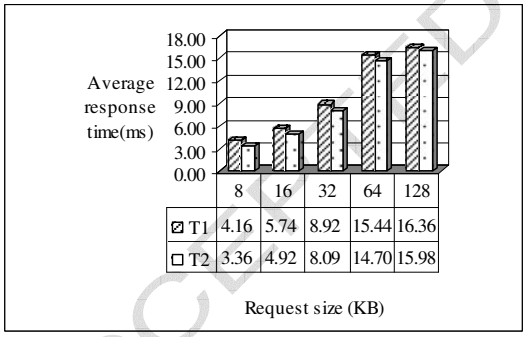


Fig. 10 Average response time with trace ST2

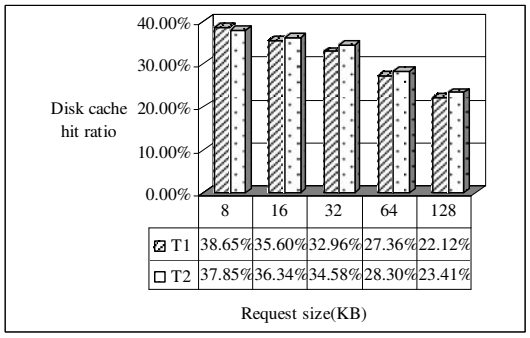
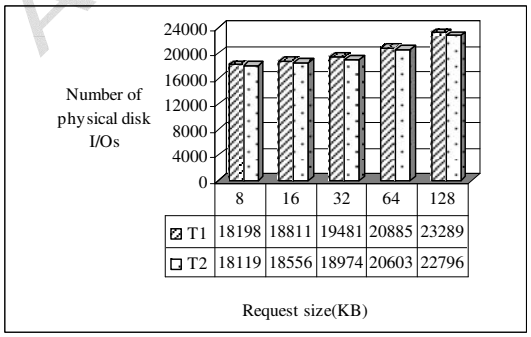


Fig. 11 Hit ratio of the disk cache with trace ST2



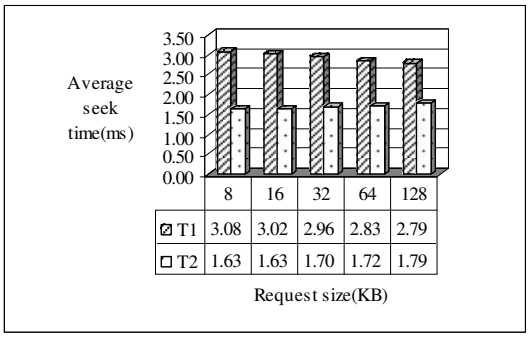
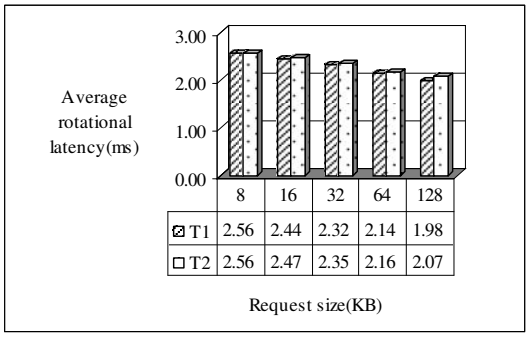


Fig. 12 Number of physical disk I/Os with trace ST2

Fig. 13 Average seek time with trace ST2



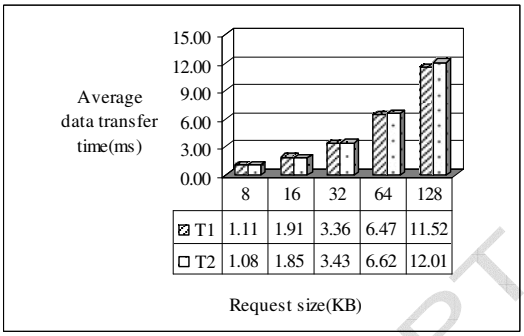


Fig. 14 Average rotational latency with trace ST2

Fig. 15 Average data transfer time with trace ST2

In Fig. 10, Fig. 11, Fig. 12, Fig. 13, Fig. 14, and Fig. 15, T1 denotes the baseline system which does not involve any data migration, T2 indicates the system which employs a 32KB object size to move data blocks. Fig. 10 illustrates that the average response time grows with the increase of the average request size. It is reasonable because bigger request size produces more data transfer time which is a portion of the average response time. When the request size is changed from 8KB to 16KB, 32KB, 64KB, and 128KB, the average response times are improved 19%, 14%, 9%, 5%, and 2.3%, respectively. The measurement indicates that bigger request size achieves less performance improvement. The hit ratios of disk cache in Fig. 11 shows a slight increase when the disk drive conducts data reorganization. Fig. 11 also demonstrates that the hit ratios are decreased with the growth of request size. The number of physical disk I/Os depicted in Fig. 12 follows the same trend as the hit ratio. This is reasonable because higher hit ratios imply less physical disk I/Os. The average seek time shown in Fig. 13 achieves significant reduction ranging from 40% to 47% when the request size is changed from 128KB to 8KB. Fig. 13 also indicates that bigger request size achieves less seek time reduction, which is consistent with Fig. 10. However, the average rotational latency depicted in Fig. 14 and the average data transfer time illustrated in Fig. 15 are not expected. The average rotational latency does not obtain any performance improvement. The highest performance improvement of the average data transfer time is only 3.1%. When the request size is increased to 64KB, a slight performance penalty of data transfer time is incurred due to the data reorganization. Fig. 14 indicates that the average rotational latency of both T1 and T2 are reduced with the growth of the request size. As discussed in section 2.2, due to zero-latency access, the rotational latency decreases with the growth of useful blocks in a track. Therefore, bigger request size can take better advantage of the zero-latency access, thus improving performance. Fig. 15 confirms the observations in Fig. 9 and Table 2.

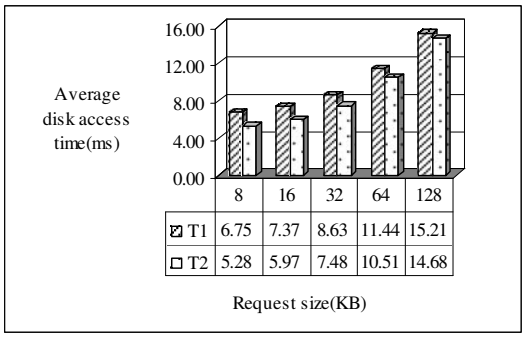


Fig. 16 Average disk access time with trace ST2

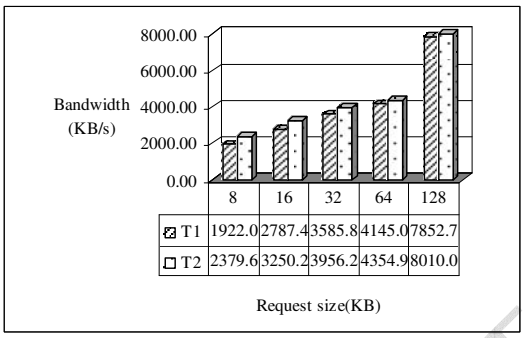


Fig. 17 Bandwidth with trace ST2

The above measurements show that the significant reduction of seek time is not dramatically reflected in the average response time. A very important reason is that the high hit ratio of disk cache reduces the impact of seek time on the average response time. Therefore, we evaluated the disk access time illustrated in Fig. 16 which excludes the impact of disk cache. The performance improvements of different request size are 22%, 19%, 13%, 8%, and 3.5%, respectively. It confirms the illustration in Fig. 10, and shows the same implication that bigger request size achieves less performance improvement. The tests indicate that another reason is because the average seek time is only a portion of the average response time. That's why the impact of the reduced seek time on the average response time is alleviated.

According to Fig. 15, the data transfer time does not achieve too much benefit from the ZBR. In order to investigate the reason, we measured the bandwidth because a different request size has different but straightforward impact on the bandwidth. Fig. 17 depicts that when the average request size is changed to 8KB, 16KB, 32KB, 64KB, and 128KB, the bandwidth achieves 24%, 17%, 10%, 5%, and 2% improvement, respectively. It does illustrate that our method (T2 in Fig. 17) can take advantage of the ZBR to improve performance. However, it does not confirm the discussion in section 2.3 that bigger request size should gain more benefits from the ZBR. The reason is that if the request size is bigger than the object size, the request will be split into several sub-requests. Therefore, the block correlations could be destroyed. For a fixed object size, the bigger the request size is, the higher the probability that the requests could be split. We employed 32KB as the object size in the measurements of this section. That's why the performance improvement decreases with the growth of the request size. We tracked the requests in the simulation, and observed many split requests even when the request size is 8KB. It gives us an indication that though the object can maintain the basic correlation among data blocks, the object size actually depends on the workload. For example, 32KB is the optimal object size for ST1 which has an average request size of 7.2 KB. That's why in the test, 8KB request size achieves the maximal performance improvement. A dynamic algorithm used to determine the optimal object size for different workloads could alleviate this problem.

6.3 Evaluating the impact of data access pattern

As discussed in section 6.1 and 6.2, the traces ST1 and ST2 are highly skewed. We reorganized the frequently accessed data blocks into a relatively small area, thus enhancing the data locality which significantly reduces the average seek time. However, the significant reduction of seek time is not reflected in the average response time. We believe that a very important reason is the high hit ratio of disk cache. Therefore, we constructed the third synthetic trace ST3 and the fourth synthetic trace ST4 which have different skews by modifying the trace ST1. The skew of ST3 is 44/10 which means 44% I/Os go to 10%

storage capacity. The request addresses in ST4 are distributed across the disk drive randomly. We believe that the trace ST4 can simulate an extreme scenario of the online transaction processing, and the decreased skew can reduce the hit ratio of disk cache.

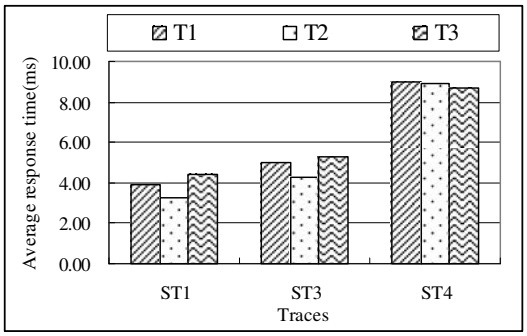


Fig. 18 Average response time

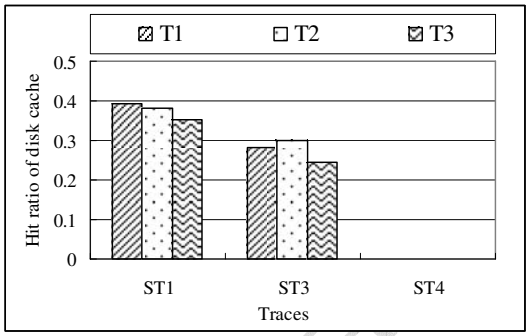


Fig. 19 Hit ratio of disk cache

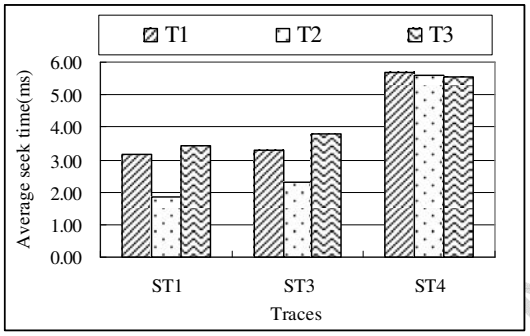


Fig. 20 Average seek time

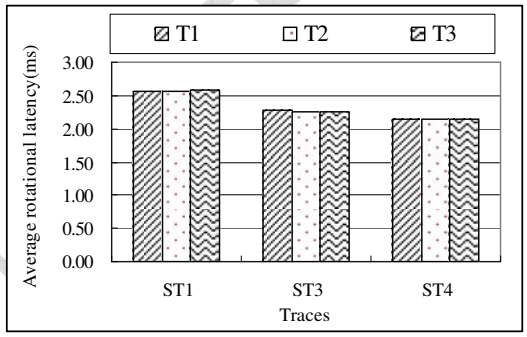


Fig. 21 Average rotational latency

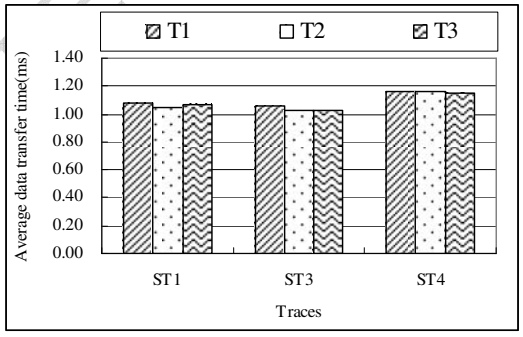


Fig. 22 Average data transfer time

In order to compare our method with the existing work, we implemented an organ pipe data layout in the simulator. In this section, T1 denotes the baseline system which does not involve any data migration, T2 indicates the system which employs a 32KB object size to move the frequently accessed data blocks to the fast band, and T3 implies that the moved data blocks are organized in an organ pipe style in the centre of disk drive.

Fig. 18 illustrates the average response time of the three systems measured by the three different traces. It shows that when the traces ST1 and ST3 are employed to evaluate the systems, our method achieves the

best performance, whereas T3 degrades the performance. The reason is because the sequentially accessed data is split on either side of the organ pipe arrangement, and the disk arm has to seek back and forth across the disk resulting in decreased performance [36]. It is interesting to observe that the system T3 obtains the best performance improvement (3.8%) when ST4 is adopted to evaluate the systems. This is because the requests in ST4 are independent. It means that the requests which are not reorganized are distributed randomly across the disk drive. Placing the hot area in the middle of disk drive can reduce the seek time when the disk arm has to be moved to locate the remaining data blocks.

Fig. 19 demonstrates the hit ratio of disk cache. As expected, the hit ratio is reduced when the skew is decreased. When we used ST4 to compare the hit ratios, the actual values of T1, T2, and T3 are 0.004%, 0.04%, and 0.003%, which is too low to be observed in Fig. 19. The low hit ratio is reasonable, because the requests in ST4 do not contain any data locality including temporal locality and spatial locality. We also can observe that when ST1 and ST3 are adopted to measure the systems, the hit ratio is decreased when the data blocks are organized in an organ pipe style.

Fig. 20 depicts the average seek time. The experimental results show that our method does reduce the average seek time significantly by using the traces ST1 and ST3, whereas the T3 incurs a growth of the average seek time. This is because T3 destroys the block correlations. As expected, Fig. 20 also shows that compared with our method, T3 obtains a better performance improvement when ST4 is adopted to measure the system. This is because the requests in ST4 are independent. However, as discussed in section 3, most of the real data accesses show some locality. It means our method is much more effective than the organ pipe in a real scenario. Fig. 21 and Fig. 22 do not show significant performance variation when different traces are used to measure T1, T2, and T3.

7. Discussion and Conclusion

Since the first disk drive was announced in 1956, disk drives have experienced dramatic development, but still lagged far behind the performance improvement of processor and RAM. A lot of I/O optimization methods have been devised to alleviate the gap between RAM and disk driver. One of the most effective approaches of improving disk access performance is increasing the hit ratio of disk cache and thus reducing the number of physical disk I/Os.

We reviewed some important characteristics of modern disk drives and disk access patterns in this paper. Attempting to take advantage of the characteristics, we clustered the frequently accessed blocks and the correlated blocks into objects, and moved the objects to the outer zones of a modern disk drive to enhance its data locality. Synthetic trace driven simulation was adopted to break down disk access time into seek time, rotational latency, data transfer time, and hit ration of disk cache, and investigate the performance gains of each of them due to the enhanced locality. Experimental results give the following indications:

(1) Data blocks are correlated with each other. When performing data reorganization, an optimal object size which compactly cluster the frequently accessed data blocks with the correlated data blocks can achieve the best performance. Otherwise, the destroyed correlations can incur performance penalty.

(2) By reorganizing the frequently accessed data blocks into a small area, average seek time achieves significant reduction due to the enhanced data locality. However, the reduction is not reflected dramatically in the response time because the high hit ratio of disk cache alleviates the impact. Even so, the reduction of seek time contributes most of the performance improvement, and the other components constitute a small portion of the improvement.

(3) Zero-latency access does not obtain too much benefit from the enhanced spatial locality.
(4) ZBR does contribute to the performance improvement, though it is relatively small in comparison with the contribution of seek time.
(5) The enhanced spatial locality does not give too many opportunities to disk cache to improve the hit ratio and reduce the number of physical disk I/Os. This could be caused by the disk model employed in the simulation. Quantum Atlas 10k is the latest model we can find in the Disksim. The model does support some new characteristics such as zero-latency, ZBR, etc. However, a disk drive with 9.1GB storage capacity and 2MB disk cache is old. It can not demonstrate other characteristics such as high TPI and DPI, and the small disk cache can not obtain too much performance gains from the enhanced spatial locality. We believe a disk model with higher storage capacity and bigger disk cache could illustrate more interesting experimental results. For example, because long distance seeks of modern disk drive may not involve enormously more overhead than short ones, the performance improvement of seek time could decrease, while the bandwidth could further increase.
(6) The data access pattern has an impact on our method. However, as discussed in section 3.1, the skewed data access is a normal behaviour of the workloads in real world. Therefore, we believe that real applications can benefit from our method.

Acknowledgements
We would like to thank the anonymous reviewers for helping us refine this paper. Their constructive comments and suggestions are very helpful. Thanks also to my friend Michael Sandling and Andrew Clemens who polished the language of this article. In addition, I am grateful to Prof. Witold Pedrycz for giving me the opportunity to clarify my thoughts.

References

[1]. S. Akyürek and K. Salem, Adaptive block rearrangement, ACM Transactions on Computer Systems 12(2) (1995) 89-121.
[2]. M. Baker, J. Hartman, M. Kupfer, K. Shirriff, J. Ousterhout, Measurements of a distributed file system, in: Proceedings of ACM Symposium on Operating Systems Principles, 1991, pp. 198–212.
[3]. A. J. C. Bik, Reshaping access patterns for improving data locality, in: Proceedings of the 6th Workshop on Compilers for Parallel Computers, 1996, pp. 229-310.
[4]. J. S. Bucy and G. R. Ganger, The DiskSim simulation environment version 3.0 reference manual, Technical Report CMU-CS-03-102, January 2003.
[5]. Cello 1999 traces, Storage Systems Program HP Laboratories, URL: http://tesla.hpl.hp.com/public_software/.
[6]. Y. Deng, F. Wang, N. Helian, D. Feng, K. Zhou, Optimal clustering size of small file access in network attached storage device, Parallel Processing Letters, 16(4) (2006) 501-512.
[7]. Y. Deng, F. Wang, N. Helian, EED: energy efficient disk drive architecture, Information Sciences, 178(22) (2008) 4403-4417.
[8]. G. R. Ganger, B. L. Worthington, R. Y. Hou, Y. N. Patt, Disk subsystem load balancing: disk striping vs. conventional data placement, in: Proceedings of the Hawaii International Conference on System Sciences, January 1993, pp. 40-49.

[9]. G. Ganger, Blurring the line between oses and storage devices, Technical report, Carnegie Mellon University, December 2001.

[10]. G. R. Ganger, M. F. Kaashoek, Embedded inodes and explicit grouping: exploiting disk bandwidth for small files, in: Proceedings of Annual USENIX Technical Conference (Anaheim, CA), January 1997, pp. 1-17.

[11]. Hitachi Global Storage Technologies – HDD Technology Overview Charts, http://www.hitachigst.com/hdd/technolo/overview/storagetechchart.html.

[12]. W. W. Hsu and A. J. Smith, The performance impact of I/O optimizations and disk improvements, IBM Journal of Research and Development 48(2) (2004) 255–289.

[13]. C. Hsu, C. Chen and Y. Su, Hierarchical clustering of mixed data based on distance hierarchy, Information Sciences 177(20) (2007) 4474-4492.

[14]. W. W. Hsu, A. J. Smith, H. C. Young, The automatic improvement of locality in storage systems, ACM Transactions on Computer Systems 23(4) (2005) 424–473.

[15]. W. W. Hsu and A. J. Smith, Characteristics of I/O traffic in personal computer and server workloads, IBM Systems Journal 42(2) (2003) 347–372.

[16]. H. Huang, W. Hung, and K. G. Shin, FS2: dynamic data replication in free disk space for improving disk performance and energy consumption, in: Proceedings of the 20th ACM symposium on Operating systems principles, 2005, pp.263-276.

[17]. R. Karedla, J. S. Love, B. G. Wherry, Caching strategies to improve disk system performance, Computer 27(3) (1994) 38 - 46.

[18]. Y. Kim, S. Gurumurthi, A. Sivasubramaniam, Understanding the performance-temperature interactions in disk I/O of server workloads, in: Proceedings of the 12th International Symposium on High-Performance Computer Architecture, 2006, pp.176-186.

[19]. S. H. Kim, H. Zhu, R. Zimmermann, Zoned-RAID, ACM transactions on storage 3(1) (2007) 1-17.

[20]. A. J. T. Lee and C. Wang, An efficient algorithm for mining frequent inter-transaction patterns, Information Sciences 177(17) (2007) 3453-3476.

[21]. Z. Li, Z. Chen, Y. Zhou, Mining block correlations to improve storage performance, ACM Transactions on Storage 1(2) (2005) 213-245.

[22]. J. Li, J. Pan and S. Chu, Kernel class-wise locality preserving projection, Information Sciences 178(7) (2008) 1825-1835.

[23]. Log-structured File System, http://log-file-system.area51.ipupdater.com/.

[24]. C. R. Lumb, J. Schindler, G. R. Ganger, D. F. Nagle, Towards higher disk head utilization: extracting free bandwidth from busy disk drives, in: Proceedings of the Fourth Symposium on Operating Systems Design and Implementation(OSDI),2000, pp. 87–102

[25]. C. Lu, G. A. Alvarez, J. Wilkes, Aqueduct: online data migration with performance guarantees, in: Proceedings of the 1st USENIX Conference on File and Storage Technologies, 2002, pp.219-230.

[26]. C. R. Lumb, J. Schindler, and G. R. Ganger, Freeblock scheduling outside of disk firmware, in: Proceedings of the 1st USENIX Conference on File and Storage Technologies, 2002, pp.275-288.

[27]. N. R. Mahapatra and B. Venkatrao, The processor-memory bottleneck: problems and solutions, ACM Crossroads 5(3) (1999).

[28]. M. McKusick, W. Joy, and S. Leffler, A fast file system for UNIX, ACM Trans. on Computer Systems 2(3) (1984) 181–197.

[29]. M. Mesnier, G. R. Ganger, E. Riedel, Object-based storage, IEEE Communications Magazine 41(8) (2003) 84 – 90.

[30]. R. V. Meter, Observing the effects of multi-zone disks, in: Proceedings of the USENIX Annual Technical Conference, January 1997, pp.19-30.

[31]. S. W. Ng, Advances in disk technology: performance issues, Computer 31(5) (1998) 75-81.

[32]. E. Pugh, Storage hierarchies: Gaps, cliffs, and trends, IEEE Transactions on Magnetics 7(4) (1971) 810-814.

[33]. Reiserfs, http://www.namesys.com/.

[34]. A. Riska and E. Riedel, Disk drive level workload characterization, in: Proceedings of the USENIX Annual Technical Conference , Boston, 2006, pp. 97-103.

[35]. D. Roselli, J. R. Lorch and T. E. Anderson, A comparison of file system workloads, in: Proceedings of the USENIX Annual Technical Conference (Berkeley, CA), 2000, pp.41–54.

[36]. C. Ruemmler and J. Wilkes, Disk shuffling, Technical report HPL-91-156, Hewlett-Packard Company, Palo Alto, CA, October 1991.

[37]. C. Ruemmler, and J. Wilkes, Unix disk access patterns, in: Proceedings of the Winter 1993 USENIX Technical Conference , 1993, pp. 313-323.

[38]. S. W. Schlosser, J. L. Griffin, D. F. Nagle, G. R. Ganger, Designing computer systems with MEMS-based storage, in: Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2000, pp.1–12.

[39]. 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 Conf. on File and Storage Technologies (FAST02), 2002, pp.259-274.

[40]. Seek distance dependent variable max VCM seek current to control thermal rise in VCM's. http://www.patentstorm.us/patents/6724564-description.html.

[41]. C. Staelin and H. Garcia-Molina, Clustering active disk data to improve disk performance, Tech. Rep. CS-TR-283-90, Dept. of Computer Science, Princeton University,1990.

[42]. A. S. Tanenbaum, J. N. Herder, H. Bos, File size distribution on UNIX systems: then and now, ACM SIGOPS Operating Systems Review, 40(1) (2006)100-104.

[43]. T. M. Wong and J. Wilkes, My Cache or yours? making storage more exclusive, in: Proceedings of USENIX Annual Technical Conference (USENIX 2002), 2002, pp. 161–175.