User-Level SCSI Disk Feature Extraction

Zoran Dimitrijević[1] Raju Rangaswami[1] Edward Chang[1]
David Watson[2] Anurag Acharya[2]

Univ. of California, Santa Barbara[1] Google[2]

{zoran,raju}@cs.ucsb.edu echang@ece.ucsb.edu {davidw,acha}@google.com

Despite the increases in the disk capacity and decreases in mechanical delays in recent years, the performance gap between magnetic disks and CPU continues to increase. To effectively improve disk performance, operating systems and file systems must have detailed information (e.g., zoning, bad-sector positions, and cache size) about the disks that they use. In this paper, we present a tool that we have developed to extract such information. We first specify the disk features that we extract. We then explain various interrogative and empirical methods for extracting these disk features. Finally, we present extraction results for two testbeds. From our empirical study, we conclude that intelligent data placement and access methods can be devised, by exploiting low-level disk knowledge, to attain better disk performance.

1. INTRODUCTION

The performance gap between hard drives and CPU-memory subsystems is steadily increasing. In order to bridge this gap, several intelligent disk scheduling and storage management strategies have been proposed [3, 4, 5, 13]. Most strategies assume that detailed disk parameters, such as zoning, bad-sector locations, and disk latency, can be obtained from the disk manufacturers. However, most information provided can be coarse and static. For instance, disk vendors only give out maximum, minimum, and average data transfer rates and seek time. In addition, some dynamic information such as the locations of bad sectors cannot be known a priori. As a consequence, the effectiveness of most traditional disk management strategies can be compromised.

For optimal disk performance, it is necessary to tune disk accesses to the requirements of the application effectively. For instance, the disk management for video playback must guarantee all IOs meeting their real-time constraints. If the system does not have fine-grained information about disk features, the system must assume the worst-case IO time or predict the IO time statistically. These pessimistic and statistical estimates of disk drive performance lead to the sub-optimal performance of the entire system.

The processor-disk speed gap has made the implementation of complex disk scheduling algorithms [3, 4, 5, 13] more practical. (More CPU cycles can be spent to improve disk performance.) In addition, the almost uniform disk geometry makes it possible to compress the disk map (refer to Sections 3.3) and store it entirely in cheaper main memory. Because of above trends, the system can afford employing more complicated and effective disk scheduling policies to improve overall system performance. Implementing these scheduling policies, however, requires knowing disk features precisely and in real-time. It is thus necessary to develop a fast, convenient, and generic method for disk feature extraction.

In this paper we present Scsibench [2], which is a generic tool for the extraction of SCSI disk features. Scsibench runs in user space on Linux systems and uses the SCSI generic interface provided by Linux. It uses both interrogative (using interrogative SCSI commands) as well as empirical (experimental) methods for feature extraction. Our tool is open source and available for download [2]. (We are not aware of any similar tool that is open source.)

Using Scsibench, disk features including 1) rotational time, 2) seek curves, 3) other mechanical delays, 4) caching and prefetching techniques used by the disk, and 5) logical-to-physical block mappings can be obtained. In addition, the access delay (including seek time and rotational delay) between two disk accesses can be predicted with high accuracy. Scsibench also supports the execution of disk traces by providing an intuitive interface for executing primitive disk commands (e.g., write, seek, prefetch, etc.). The trace support also provides a very accurate timing mechanism. Using knowledge about disk features and the trace support provided by Scsibench, system or application programmers can know the precise distribution of times spent by the disk performing various operations. Bottlenecks can thus be identified, and disk scheduling can be adjusted accordingly to utilize the disk more efficiently. Scsibench can also be used as a tool to validate research that has so far been restricted to using simulated disk models for performance analysis and evaluation.

The rest of the paper is organized as follows: In Section 2, we present several disk features that are of interest. In Section 3, we explain various feature extraction methods that are part of the implementation of our Scsibench tool. Section 4 presents the results obtained using our tool on several different disks. In Section 5, we compare our extraction methods with existing approaches. We offer some concluding remarks in Section 6.

2. DISK FEATURES

Before we get into the details of disk features that are of interest when designing high performance systems, we provide a brief overview of the disk architecture. A more detailed explanation of these features can be found in [8]. The main components of a typical modern day disk drive are:

- One or more *disk platters* rotating in lockstep fashion on a shared *spindle*,
- A set of *read/write heads* residing on a shared arm moved by an *actuator*,
- *Disk logic*, including the disk controller,
- *Cache memory* with embedded replacement and scheduling algorithms, and
- The *IO Bus*.

The data is logically organized into disk *blocks* (the minimum unit of disk access). Typically, a block corresponds to one disk *sector*. The set of sectors on the same magnetic surface and that are at the same distance from the central spindle form a *track*. The set of tracks at the same distance from the spindle form a *cylinder*. Error detection and correction data, and other disk meta-data are stored in between regular sectors. Sectors can store the data for a logical block, can be reserved for future bad sector re-mappings (spare sectors), or can store disk meta-data. They can also be marked as bad if they are located on the damaged magnetic surface.

The *storage density* (amount of data that can be stored per square inch) is constant for magnetic surfaces (disk media) used in modern disks. However, since the outer tracks are longer, they can store more data than the inner ones. This is why modern disks do not have a constant number of sectors per track. Disks divide cylinders into multiple *disk zones*, each zone having a constant number of sectors per track.

The disk *rotational speed* is a constant value (with some random variations). Since the track size varies from zone to zone, each disk zone has different maximum *raw disk bandwidth* (data transfer rate from the track to the internal disk logic). The outer zones have significantly larger raw disk bandwidth than the inner ones.

When the disk switches from one track to the next, some time is spent in repositioning the disk head to the center of the next track. If two adjacent tracks are on the same cylinder, this time is referred to as *track switch time*. If the tracks are on different cylinders, then it is referred to as *cylinder switch time*. In order to optimize the disk for sequential access, disk sectors are numbered so that the starting sectors on two adjacent tracks are skewed. This skew compensates for the track or cylinder switch time. It is referred to as *track skew* and *cylinder skew* for track and cylinder switches respectively.

The *seek time* is the time that the disk arm needs in order to move from the current position to the destination cylinder. In the first stage of the seek operation, the arm accelerates at a constant rate. This is followed by a period of constant maximum velocity. In the next stage, the arm slows down with constant deceleration. The final stage of the seek is the settle time, which is needed to position disk head exactly at the center of the destination track. Since the disk seek mainly depends on the characteristics of the disk arm and its actuator, the seek time curve does not depend on the starting and destination cylinders. It depends only on the seek distance (in cylinders).

The magnetic surfaces of a disk are not perfect because it would be too expensive to build surfaces without defects. Hence, during low-level format, bad sectors are marked and skipped during logical block numbering. Also, some disk sectors are reserved as spare, to enable disk to re-map bad sectors that occur during the disk lifetime. Algorithms for spare sector allocations differ from disk to disk. In order to accuratly model the disk for intelligent data placement, scheduling, or even simple seek curve extraction, system needs detailed mapping between the physical sectors and the logical blocks. In addition to mapping, a system has to be able to inquire the disk about re-mapped blocks. Re-mapping occurs when disk detects a new bad sector.

The *disk cache* is divided into a number of *cache segments*. Each cache segment can either be allocated to a single sequential access stream or can be further split into blocks for independent allocation. The cache parameters of interest are segment size, number of cache segments, segment replacement policy, and prefetching and write buffer organization. Prefetching is used to improve performance of sequential reads. Write buffer is used to delay actual writing to disk media, and enable disk to re-schedule writes.

In this paper, we present Scsibench, a tool that automatically extracts all the disk features mentioned above. In the next section, we present the methods used to extract these features.

3. EXTRACTION METHODS

In this section, we present methods for extracting various disk features. We use a combination of the interrogative and empirical methods. Interrogative methods use interrogative SCSI commands to inquire the disk about its features. Empirical methods measure completion times for different disk access patterns, and extract disk features based on the timing measurements.

In some of our extraction methods we assume the ability to force access to the disk media for read or write requests. Most modern disks allow turning off the write buffer. In the case of SCSI disks, this can be done by turning off the disk buffers, or by setting the "force media access bit" in a SCSI command.

3.1 Rotational Time

Since modern disks have a constant rotational speed, if the interrogative SCSI command for obtaining rotational period (T_{rot}) is implemented, it will return the correct value. In the absence of the interrogative command, we can also use the following empirical method to obtain T_{rot}: First, we ensure that read (or write) commands access disk media. Next, we perform n successive disk accesses to the same block, and measure access completion times. The absolute completion time for each disk access is:

$$T_i = T_{end\_reading} + T_{transfer} + T_{OS\_delay_i} \qquad (1)$$

$T_{end\_reading}$ is the absolute time immediately after the disk reads the block from the disk media. $T_{transfer}$ is the transfer time needed to transfer data over the IO bus. $T_{OS\_delay_i}$ is the time between the moment when the OS receives data over the IO bus, and the moment when the data is transfered to the user level Scsibench process. Since we need to wait for one disk rotation for each successive disk block access, we have:

$$T_{i+1} - T_i = T_{rot} + (T_{OS\_delay_{i+1}} - T_{OS\_delay\_i}) \qquad (2)$$

$$T_{n+1} - T_1 = n \times T_{rot} + (T_{OS\_delay_{n+1}} - T_{OS\_delay_1}) \quad (3)$$

The rotational period for modern disks is longer than OS delays and other IO overheads (not including seek and rotational times). Because of this, each disk access will increase the number of ellapsed disk rotations by one. Thus, we can measure the rotational period as:

$$T_{rot\_measured} = T_{rot} + \frac{\Delta T_{OS\_delay_{n+1,1}}}{n} = \frac{T_{n+1} - T_1}{n} \quad (4)$$

For large n, the error term ($\frac{\Delta T_{OS\_delay_{n+1,1}}}{n}$) is negligible.

3.2 OS Delay Variations

In order to estimate variations in operating system delay for IO requests, we use the following method. First, we turn off all disk caching and disk buffering. Then, we read the same block on disk in successive disk rotations, as in the empirical method for extracting T_{rot}. We measure completion times for each read request.

Variations in the rotational period T_{rot} are negligible. Because of this, variations in $T_i - T_{i-1}$ from Equation 2 can be directly attributed to the variations in $\Delta T_{OS\_delay_{i,i-1}}$, where:

$$\Delta T_{OS\_delay_{i,i-1}} = T_{OS\_delay_i} - T_{OS\_delay_{i-1}} \quad (5)$$

Thus, by measuring variations in $T_{i+1} - T_i$ from Equation 2, we can estimate variations in operating system delay.

3.3 Mapping from Logical to Physical Block Address

Modern SCSI disks implement SCSI commands for address translation (Send/Receive Diagnostic Command [10]) which can be used to extract disk mapping. However, in the case of older SCSI disks, or for disks where address translation commands are not supported (e.g. ATA disks), empirical methods are necessary.

3.3.1 Interrogative Mapping

Using the interrogative method, a single address translation typically requires significantly less than one millisecond. But, since the number of logical blocks is large, extracting mapping using translation commands for each logical block is inefficient. Fortunately, modern disks are optimized for sequential access. Also, most disks use the skipping method to skip bad sectors (instead of re-mapping them) during the low-level format. Because of this, logical blocks on a track are placed mostly sequentially. Thus, we can extract highly accurate mapping information by translating just one address per track, except when we detect anomalies (tracks with bad blocks).

Since the number of re-mapped blocks is small compared to total number of blocks, we propose using two data structures for storing mapping information obtained using interrogative methods. In the first data structure, we store the mapping as it was available immediately after the low-level format. Since there are no re-mapped blocks, we simply store the starting logical block number and the track size (in blocks) for each track. The second data structure is used to store information about all re-mapped blocks. This way, we only need to update the second structure periodically, using SCSI interrogative commands.

Procedure: *Interrogative Mapping*

- **Variables:**

  cyl_num : Total number of cylinders
  $track\_per\_cyl$: Number of tracks per cylinder
  i : Cylinder number
  j : Track number
  $cyl\_info[i]$: data structure for cylinder i info
  $cyl\_info[i].logstart[j]$: Starting LBA for track j on cylinder i
  $cyl\_info[i].logsize[j]$: Track j's size in blocks

- **Execution:**

  for $i = 0$ to cyl_num do
  for $j = 0$ to $track\_per\_cyl$ do
  $cyl\_info[i].logstart[j]$ = phys_to_log(i,j,0)
  if not valid($cyl\_info[i].logstart[j]$) then
  $cyl\_info[i].logstart[j]$ = phys_to_log(i,j,1)
  if not valid($cyl\_info[i].logstart[j]$) then
  mark track *bad*.
  sort by $cyl\_info[i].logstart[j]$
  calculate $cyl\_info[i].logsize[j]$

Figure 1: Interrogative method for mapping extraction.

Figure 1 presents a simplified algorithm for the interrogative extraction used in Scsibench. Using the SCSI command for physical to logical address translation, we extract the LBA for the first sector (sector zero) of each track. In the case when sector zero is marked as bad, we continue performing address translation for subsequent sectors until we obtain a valid logical block number. When we come across a cylinder in which the number of sectors that do not have a valid LBA (bad blocks), is above a fixed threshold, we mark that cylinder as logically bad. We do not use these cylinders in our seek curve extraction method. Tracks which have a substantial number of blocks without a valid LBA are usually the ones containing mostly spare sectors.

3.3.2 Empirical Mapping

Empirical methods for accurate mapping extraction are needed for disks that do not support address translation. However, we do assume that we can force the disk media access, as explained at the beginning of this Section. In what follows, we present extraction method used in Scsibench.

In the first step, we measure the time delay between reading a pair of blocks from the disk. We repeat this measurement for a number of block pairs, keeping the position of the first block in the pair always fixed. In successive experiments, we linearly increase the position of the second block in a pair. A similar method was also used in the study of [11], but since they could not force access to disk media, it was difficult to extract exact track and cylinder boundaries which are needed for accurate mapping. Our tool extracts accurate positions of track and cylinder boundaries.

$T(i)$ defined in Equation 6 is the completion time measured at the moment when the user process receives data for logical block address i. The variables on the right side of Equation 6 are defined in Section 3.1.

$$T(i) = T_{end\_reading} + T_{transfer} + T_{OS\_delay} \quad (6)$$

Access time ($T_a(x,0) = T_{end\_reading}(i) - T_{end\_reading}(0)$) is the time needed to access block x after accessing block 0. It includes both seek time and rotational delay, but does not include transfer time and OS delay.

$$\Delta T(x,0) = T(x) - T(0)$$

$$\Delta T(x-1,0) = T_a(x,0) + (T_{OS\_delay_{x-1}} - T_{OS\_delay_0})$$

$$\Delta T(x,0) = T_a(x,0) + (T_{OS\_delay_x} - T_{OS\_delay_{0'}}) \quad (7)$$

$$\Delta = \Delta T(x,0) - \Delta T(x-1,0)$$

$$\Delta = T_a(x,0) - T_a(x-1,0) + (\Delta T_{OS\_delay_{x,0'}} - \Delta T_{OS\_delay_{x-1,0}}) \quad (8)$$

Equation 8 presents the first derivative of $\Delta T(x,0)$ defined in Equation 7.

When the OS delay variations are small, and both blocks $x-1$ and x are on the same track, Δ is small, and proportional to $T_{rot}/track_{size}$. When the access can be performed without any additional rotational delay after seek, or with delay of a whole T_{rot}, Δ is proportional to $-T_{rot} + T_{rot}/track_{size}$. The sign of Δ depends on T_{OS_delay}.

If x and $x-1$ are on different tracks, or cyinders, Δ is proportional to skew time T_{skew}, where T_{skew} is the track or cylinder skew time. Since the cylinder skew is usually larger than the track skew, we can use the positions of the skew times in Δ in order to find out accurate track and cylinder boundaries.

$$norm(\Delta) = (\Delta T(x,0) - \Delta T(x-1,0)) \bmod T_{rot} \quad (9)$$

Equation 9 defines the normalized first derivative from Equation 8. This way we eliminate the $-T_{rot}$ factor, and can hence easily extract accurate disk mapping automatically.

Since $norm(\Delta)$ is useful for automatic extraction only if the OS delay is small comparing to the skew times. Since the OS delay is random variable, we perform several measurements whenever $|\Delta|$ is greater than a specific threshold ($0.02 * T_{rot}$), and stop when the difference between two consecutive Δs is less than a specific threshold (5%).

Remark

Interrogative mapping is faster than empirical mapping. Also, it can give us more detailed mapping, including information about bad and spare sectors. However, many disks do not support interrogative commands. In such cases, there is no alternative to empirical mapping.

3.3.3 Disk Zones

Since the outer tracks of the disk are longer, they can store more data than the inner ones. To utilize this property, disk manufactures divide the cylinders into multiple zones. Each zone has a constant number of sectors per track. This number differs from zone to zone.

Using the extracted disk mapping, Scsibench implements methods for the extraction of zoning information, including precise zone boundaries, the track and cylinder skew factors for each zone, the track size in logical blocks, and the sequential disk throughput.

The algorithm used to extract zoning information scans the cylinders from the logical beginning to the logical end based on the disk mapping table. Due to the presence of bad and spare sectors, some tracks in a zone have smaller number of blocks than the others. Since we store only the track size (in logical blocks) for each track, we may detect a new zone incorrectly. In order to minimize the number of false positives, we use the following heuristics. First, we ignore cylinders with large number of spare sectors. Second, during the cylinder scan, we detect a new zone only if the maximum track size in the current cylinder is different from the track size of the current zone by more than two blocks. Third, we detect a new zone only when the size of the new zone (in cylinders) is above specific threshold.

3.4 Seek Curves

Seek time is the time that the disk head requires to move from the current to the destination cylinder. We implement two methods for seek curve extraction. The first method uses the SCSI seek command to seek to a destination cylinder. The second one measures the minimum time delay between reading a single block on the source cylinder and reading a single block on the destination cylinder. In order to find this minimal time, we can measure the time between reading a fixed block on the source cylinder, and all blocks on one track in the destination cylinder. The seek time is the minimum of the measured times. Since LBAs grow lineary on a track, we also implement a binary method in order to find the minimum access time (which is equal to seek time).

Using the extracted mapping, we can calculate the distance in cylinders between any two logical disk blocks. $T_{seek}(y,x)$ returns seek time (in μs) between logical blocks x and y. This function is symmetrical, i.e., $T_{seek}(y,x) = T_{seek}(x,y)$. This seek time also includes the disk head settling time.

3.5 Predicting Rotational Delay

In order to optimize disk scheaduling algorithms one may use both seek and rotational delay characteristics of a disk [4, 13, 5, 6]. Since the logical blocks are placed sequentially on a track, we can predict rotational distance between two LBAs using the following:

- mapping information extracted in Section 3.3, and
- skew factors for the beginning of each track, relative to a choosen rotational reference point.

In this study we choose the disk block with LBA zero as the reference point. The c_i is track's cylinder number, t_j is track's position in a cylinder, $track_{size}(c_i, t_j)$ is track's size in logical blocks, $LBA_{start}(c_i, t_j)$ is tracks's starting logical block number, and T(LBA) is the time after access to a specific LBA. The skew factor of a track is defined as:

$$s_{c_i,t_j} = [T(LBA_{start}(c_i,t_j)) - T(LBA_0)] \bmod T_{rot} \quad (10)$$

If the number of spare and bad sectors is small, we can accurately predict the rotational distance between two LBAs (x

and y) using the following equations:

$$X = T_{rot} \times \frac{[x - LBA_{start}(c_x, t_x)] \bmod track_{size}(x)}{track_{size}(x)} + s_{c_x,t_x} \quad (11)$$

$$Y = T_{rot} \times \frac{[y - LBA_{start}(c_y, t_y)] \bmod track_{size}(y)}{track_{size}(y)} + s_{c_y,t_y} \quad (12)$$

$$T_{rot\_del}(y, x) = (Y - X) \bmod T_{rot} \quad (13)$$

Using the seek time $T_{seek}(y, x)$ defined in Section 3.4 and the rotational delay prediction from Equation 13 we can predict the access time to a disk block y after access to a block x, $T_a(y, x)$ as:

$$T_a(y, x) = T_{rot\_del}(y, x) + T_{rot} \times \left\lceil \frac{T_{seek}(y, x) - T_{rot\_del}(y, x)}{T_{rot}} \right\rceil \quad (14)$$

3.6 Read Cache Parameters

In this section, we present methods to find out the cache segment size, the number of segments, and the segment replacement policy.

3.6.1 Cache Segment Size

The method for extracting the cache segment size consists of three stages. First we read few sequential disk blocks from a specific disk location. Then we wait for a long enough period of time (several disk rotations) to allow the disk to fill up the cache segment with prefetched data. In the last stage, we read consecutive disk blocks and measure the completion time.

If the block is in the cache, the completion time only includes the transfer time from the cache to the OS through the SCSI bus and a random T_{OS_delay}. If the block is not in the cache, the completion time also includes seek time and rotational delay, as well as data transfer time.

Since the seek time and rotational delay are the dominant factors in the completion time, we can detect the size of cache segment by detecting the moment when the completion time includes mechanical seek and rotational delays.

3.6.2 Number of Cache Segments

In order to extract the number of disk cache segments, we need to be able to clear the cache. We do this by performing a large number of random and sequential reads for different logical blocks, which effectively clear the cache by polluting it.

Our extraction method lineary increases the number of sequential streams accessing the disk. In each iteration we do the following: For each stream we read a few blocks (from locations which are not used in cache pollution). Then we wait sufficient amount of time so that disk can fill the cache segment with prefetched data. After this step, we assume that the disk cache has allocated one cache segment for each stream.

We perform read requests for all streams, and measure completion times. If the completion times are smaller than a specific threshold, we assume that all blocks were in the cache, and that the number of cache segments is greater than or equal to the number of streams in this iteration. When we detect that one read request requires an amount of time that is greater than the threshold, we need to repeat the experiment to confirm that the excessive time is not caused by high random OS delay. When we confirm that the completion time is not caused by OS delay, we deduce that the number of cache segments is equal to the number of streams in the previous iteration.

If we differ access pattern, and consider which stream is not serviced from the cache, we can deduce the policy for the cache segment replacement.

3.7 Write Buffer Parameters

In this Section, we concentrate on extracting the disk write buffer size. If the disk has sufficient space in the write buffer, then the write command will be completed as soon as data is transferred to the disk's write buffer. The disk writes this data to the disk media in an optimal manner at some later time. We perform the following method:

Procedure: *Write Buffer Size*

- **Variables:**

  max_size : Maximal estimated write buffer size
  $start$: Starting LBA for write request
  far : LBA for a block with large $T_{seek}(start, far)$
  i : Write request size iteration
  T_1, T_2, T_{prev}: Time registers

- **Execution:**

  $T_{prev} = 0$
  for $i = 1$ to max_size do
      scsi_seek(far)
      wait($20 \times T_{rot}$)
      T_1 = get_time()
      scsi_write($start$, $size$)
      T_2 = get_time()
      if $T_2 - T_1 - T_{prev} > \frac{T_{rot}}{10}$ then
          return $i - 1$
      $T_{prev} = T_2 - T_1$

Figure 2: Empirical method for extracting write buffer size.

Figure 2 presents our method for the empirical buffer size extraction. Between iterations, we allow the disk to purge write buffers to the disk media. Before issuing the write request, we also seek to a cylinder far away from the write request's destination.

If the write request size is smaller than the write buffer, we expect that the write completion time will increase lineary, proportional to the throughput of the SCSI IO bus. When the write request size is greater than the write buffer, the completion time will incur seek and rotational delays. We can detect this using the simple heuristics.

4. EXPERIMENTAL RESULTS

Before we present our experimental results, we provide an overview of the Scsibench system design. The main components of Scsibench are the following:

	Disk	T_{rot}(in μs)	RPM	Interrog.
1.	ST39102LW	5972.56	10045.94	10045
2.	ST318437LW	8305.83	7223.84	7200

Table 1: Rotational time for two testbed disks.

- A user level SCSI command library (SCSI library) which is built on top of the Linux SCSI generic interface,
- Implementation of algorithms (described in Section 3) for extracting various disk features, and
- Execution of SCSI disk access traces.

Scsibench runs as a user level process on Linux systems, and uses the SCSI generic interface to talk to SCSI disk. Using the command line arguments, user can specify which features to extract, or execute trace file.

In this section, we present experimental results for two testbeds. The first testbed is a dual Intel Pentium II 800MHz machine with 1GB of main memory and a 9GB Seagate ST39102LW 10000 RPM SCSI disk (12 disk heads). The second configuration is an Intel Pentium III 800MHz machine with 128MB of main memory and an 18GB Seagate ST318437LW 7200 RPM SCSI disk (2 disk heads). The first configuration is a typical SMP server system with a fast SCSI disk and a large number of tracks per cylinder. The second configuration is a typical workstation, with large, but not very fast hard disk (smaller rotation rate) and with a smaller main memory. We present experiements for each disk feature described in Section 3.

4.1 Rotational Time

Using Equation 4 to calculate the time required for a single rotation of the disk, we obtained rotational times for both testbed configurations. These are presented in Table 1.

4.2 Variations in OS Delay

B 0	; Turn off all disk buffering
R 0 1	; Read one sector starting from LBN 0
T 2	; Store current time to register T_2
	; repeat the following:
R 0 1	; Read one sector starting from LBN 0
T 3	; Store current time to register T_3
- 0 3 2	; Print $T_3 - T_2$
- 2 3 0	; $T_2 = T_3 - 0$
...	

Figure 3: Sample trace file to find T_{OS_delay} variations.

Based on Equation 2, we note that the variation in operating system delay for disk accesses is proportional to the variation in request completion times for the same request. Using the trace[1] execution described in Figure 3, we measured variations in request completion times (and thus, the distribution of operating system delay variations) for the two testbed configurations. These are presented in Figures 4 and 5.

Figure 4 shows results for our first testbed configuration. We can see that the variations in OS delay are of the order of $10\mu s$.

[1]Please refer to Appendix A for a detailed explanation of Scsibench trace commands

Figure 5 shows results for our second testbed configuration. Here the OS delay variations are of the order of $40\mu s$, with a greater percentage of large OS delays as compared to the first testbed. This is because our second testbed has much lesser memory and a single CPU, while the first testbed is a dual processor system with huge main memory size.

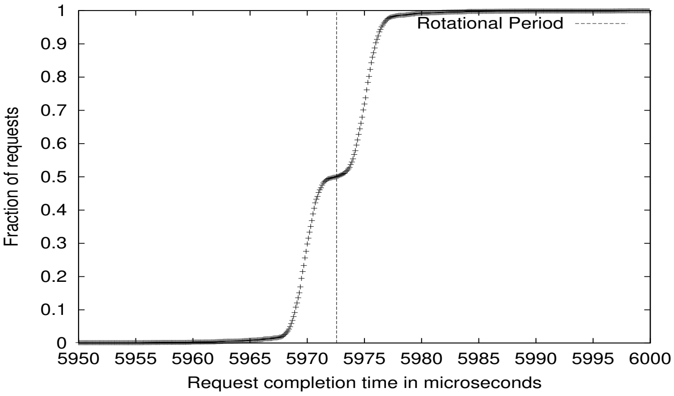

Figure 4: Distribution of request completion times for Seagate ST39102LW.

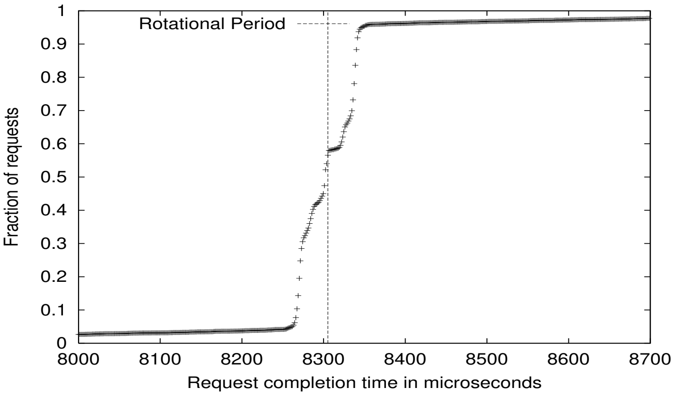

Figure 5: Distribution of request completion times for Seagate ST318437.

4.3 Mapping from Logical to Physical Block Address

4.3.1 Interrogative Mapping

Sample results from the mapping extraction for our testbed 2 configuration are presented in Figure 6. In each line we print information for one cylinder. After C we print the cylinder number, the starting LBA, and the cylinder size (in logical blocks). After T we print track information, first the track number, then the starting LBA, and the track size (in logical blocks). We print this information for all tracks.

We store the starting LBA and the size of each disk track. If the number of bad sectors in a track is greater than the number of spare sectors allocated per each track, then, the track size is smaller than the size of a regular track in that zone (cylinders $718 - 721$ in Fugure 6).

4.3.2 Empirical Mapping

We present results for the disk used in testbed 1 in Figures 7-10. This disk has 12 tracks per cylinder.

In Figure 8 we present the access time $\Delta T(x, 0)$ (defined in Equation 7) between disk blocks 0 and x. The rotational pe-

```
C 0 0 1500 T 0 0 750 1 750 750
C 1 1500 1500 T 0 2250 750 1 1500 750
C 2 3000 1500 T 0 3000 750 1 3750 750
C 3 4500 1500 T 0 5250 750 1 4500 750
C 4 6000 1500 T 0 6000 750 1 6750 750
C 5 7500 1500 T 0 8250 750 1 7500 750
C 6 9000 1500 T 0 9000 750 1 9750 750
...
C 718 1077000 1499 T 0 1077000 750 1 1077750 749
C 719 1078499 1499 T 0 1079248 749 1 1078499 750
C 720 1079998 1499 T 0 1079998 750 1 1080748 749
C 721 1081497 1499 T 0 1082246 749 1 1081497 750
...
```

Figure 6: Sample LBA-to-PBA mapping for Seagate ST318437LW.

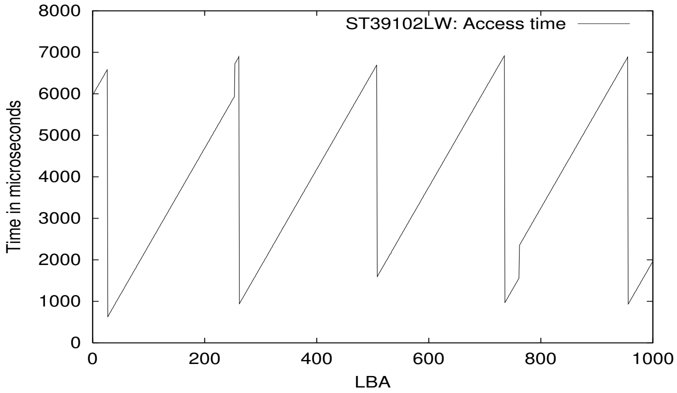

Figure 7: Sample of $\Delta T(x,0)$ needed to read logical block x after reading same starting LBA_0 for ST39102LW.

riod of the disk (T_{rot}) is approximately $6ms$. We use Figure 7, where we enlarge a small section of Figure 8, to explain our results in detail.

We can see that for small values of x, the access time $\Delta T(x,0)$ is larger than T_{rot}. When T_{OS_delay} (defined in Section 3.1) is larger than the rotational distance between blocks 0 and x, the second read request (to the logical block x) has to be serviced during the next disk rotation.

When $\Delta T(x,0)$ is greater than T_{OS_delay}, an additional disk rotation is not needed. $\Delta T(x,0)$ increases linearly for all blocks on the same track. When blocks $x-1$ and x are located on different tracks, $\Delta T(x,0)$ increases by the track (or cylinder) skew time (after which $\Delta T(x,0)$ continues to increase linearly). In our example this happens at the logical block number 254.

When the access time to the block $x-1$ requires a rotational delay of nearly T_{rot}, and the access to x does not require any rotational delay after seek, $\Delta T(x,0)$ decreases by T_{rot}. This happens at the block number 262 for the first time. At the next track boundary (508), one skew time increase and a T_{rot} decrease are overlapped. Figure 8 shows $\Delta T(x,0)$ curve for the distances up to 5000 logical blocks.

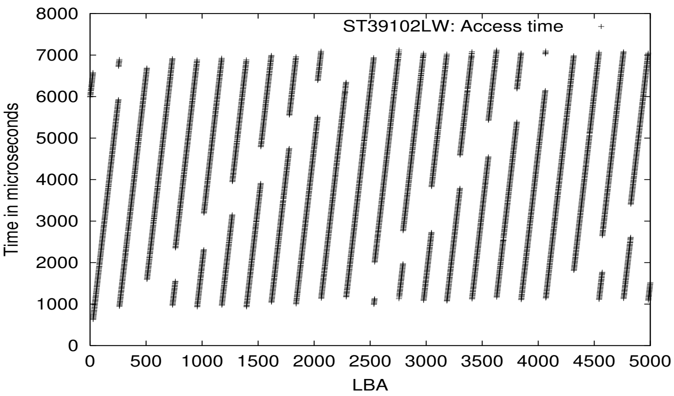

Figure 8: Time $\Delta T(x,0)$ needed to read logical block x after reading same starting LBA_0 for ST39102LW.

Figure 9 shows the first derivative of $\Delta T(x,0)$ curve (Δ) defined in Equation 8. We can see that, since T_{OS_delay} is a random variable, $\Delta T(x,0)$ can also suffer from an increase of one T_{rot}. In this experiment, it occurs at $x = 2758$. This happens when the difference between T_{OS_delay} in successive measurements is substantial, so that $\Delta T(x,0)$ incurs one disk rotation more than $\Delta T(x-1,0)$. The positive T_{rot} increase in $\Delta T(x,0)$ is allways followed by one negative T_{rot} decrease in the next few accesses.

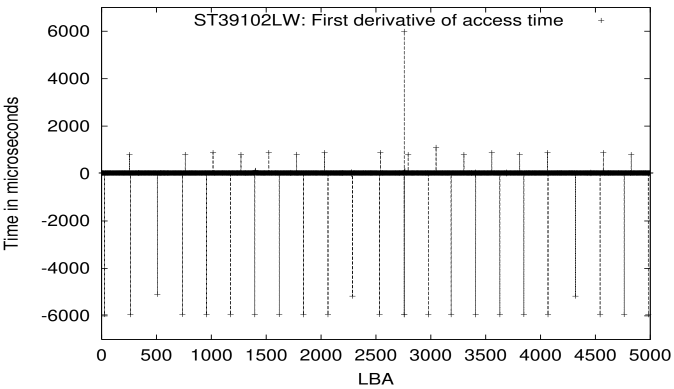

Figure 9: First derivative (Δ) of access time($\Delta T(x,0)$) for ST39102LW.

In order to extract empirical mapping automatically, we use several heuristics to find the normalized value for Δ ($norm(\Delta)$) defined in Equation 9. Figure 10 presents $norm(\Delta)$ for our testbed 1, where we capture only the track and cylinder skew times. The positions of the track and cylinder skew times on the x-axis are exact positions of the track and cylinder boundaries (every 254 blocks for Figure 10). We note that the difference between the track and cylinder skew times is not large (approximately $880\mu s$ and $1100\mu s$ respectively). The skew times to switch from an odd to an even track, and from an even to an odd track are also slightly different ($880\mu s$ and $800\mu s$ respectively).

Figure 11 presents the empirical mapping extraction results for testbed 2. The disk used in this configuration has two tracks per cylinder. Results from Section 4.2 show that variations in operation system delay, and hence the noise in measured $norm(\Delta)$, is much higher than for the first testbed configuration. But, since T_{OS_delay} is random variable, and we can repeat experiment in order to find accurate $norm(\Delta)$, we can limit the noice level, and extract accurate $norm(\Delta)$.

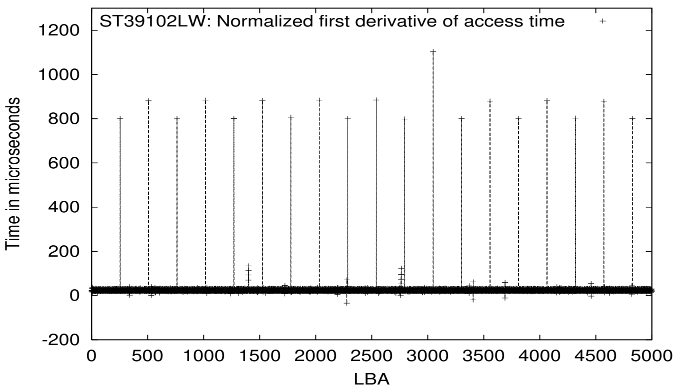

Figure 10: Normalized first derivative ($norm(\Delta)$) for ST39102LW.

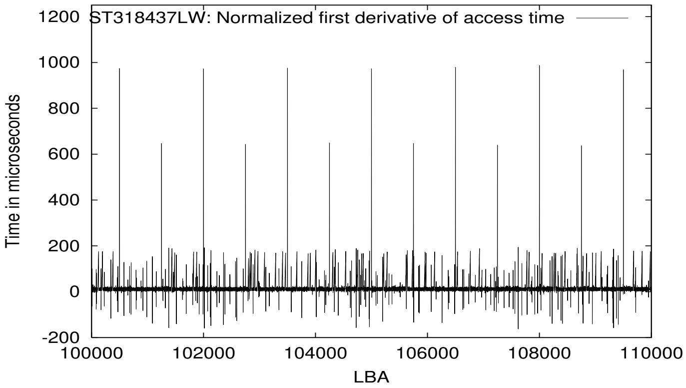

Figure 11: Normalized first derivative $norm(\Delta)$ for ST318437LW.

4.3.3 Disk Zoning

Zone	*Cylinders*	t_{size}	$\mathcal{TR}$	$\mathcal{TR}_{max}$	$\gamma(1)$	$\mathcal{H}$
1	0-847	254	18.85	21.77	1108	884
2	848-1644	245	18.02	21.01	1108	885
3	1645- 2393	238	17.51	20.40	1098	876
4	2394- 3097	227	16.70	19.45	1114	891
5	3098- 3758	217	15.99	18.60	1115	890
6	3759- 4380	209	15.43	27.91	1105	885
7	4381- 4965	201	14.84	17.23	1100	876
8	4966- 5515	189	13.98	16.20	1123	901
9	5516- 6031	181	13.39	15.52	1125	903
10	6032- 6517	174	12.89	14.92	1107	885
11	6518- 6961	167	12.38	14.31	1118	899

Table 2: Disk zone features for ST39102LW.

Table 2 and 3 present the zoning information extracted for the Seagate ST39102LW and Seagate ST318437LW disks respectively. t_{size} dentotes track size in logical blocks. $\mathcal{TR}$ is the transfer rate measured for long sequential reads that spans over multiple cylinders. $\mathcal{TR}_{max}$ is the calculated theoretical maximum transfer rates for read requests which suffer from zero seek and rotational delays. $\mathcal{H}$ and $\gamma(1)$ are the track and cylinder switch times in microseconds.

Figure 12 provides the disk bandwith for large sequential accesses depending on the starting LBA for two different SCSI

Zone	*Cylinders*	t_{size}	$\mathcal{TR}$	$\mathcal{TR}_{max}$	$\gamma(1)$	$\mathcal{H}$
1	0- 4553	750	31.07	46.24	977	652
2	4554- 6582	687	28.94	42.35	985	654
3	6583- 8247	678	28.48	41.80	987	648
4	8248-11554	666	27.92	41.06	1134	651
5	11555-14597	625	27.13	38.53	981	646
6	14598-17370	600	26.00	36.62	983	652
7	17371-19908	583	25.44	35.94	987	657
8	19909-22226	550	24.49	33.91	982	648
9	22227-26338	500	22.74	30.82	982	650
10	26339-28170	458	21.04	28.23	1159	654
11	28171-29850	437	20.24	26.94	990	663

Table 3: Disk zone features for ST318437LW.

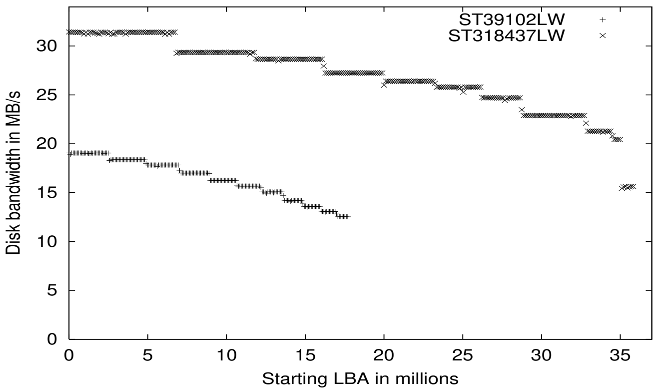

Figure 12: Disk bandwith depending on data location for two SCSI disks.

disks. For modern disks, the difference between the maximum and minimum sequential disk bandwidth is usually a factor of two.

4.4 Seek Curves

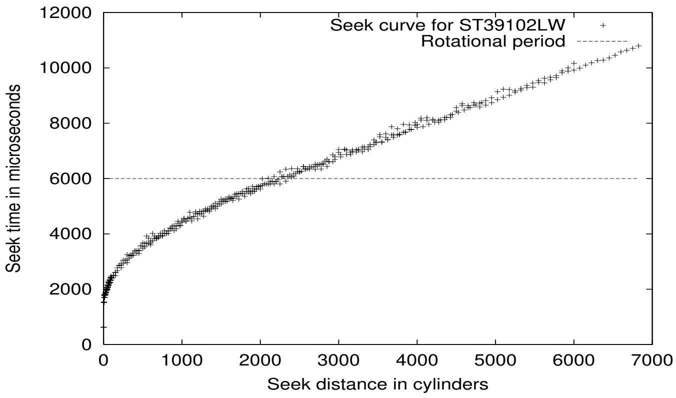

Figure 13: Complete seek curve for ST39102LW.

In figure 13 we present the seek curve for our first testbed (ST39102LW). We can see that the difference between the rotational period and the maximum seek time is less than a factor of two. We also note that the seek time depends mainly on the seek distance in cylinders, and not on starting or destination cylinder positions, since the variations in the seek curve are small. Figure 14 presents the seek curve for the second testbed.

4.5 Rotational Delay Prediction

From Equation 13, we note that, in order to predict rotational delay accurately, we need to extract the skew factor (defined

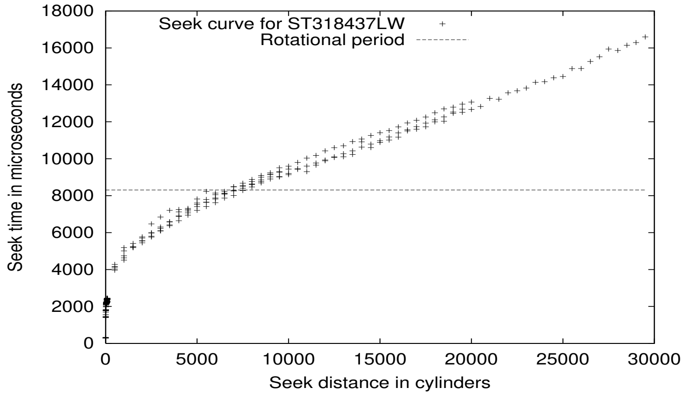

Figure 14: Complete seek curve for ST318437LW.

Cyl	$Track$	$Skew(\mu s)$	LBA_{rot0}
0	0	0	0
0	1	755.15	475
0	2	1595.02	694
0	3	2352.35	915
0	4	3191.09	1134
0	5	3949.03	1356
0	6	4788.61	1574
0	7	5546.36	1796
0	8	414.31	2268
0	9	1170.86	2490
0	10	2009.76	2708
0	11	2767.37	2930
1	0	3829.42	3139
...	...	...	...

Table 4: Rotational delay modeling for ST39102LW. Disk blocks with LBA_{rot0} are on the same disk radius. The track size for this zone is 254.

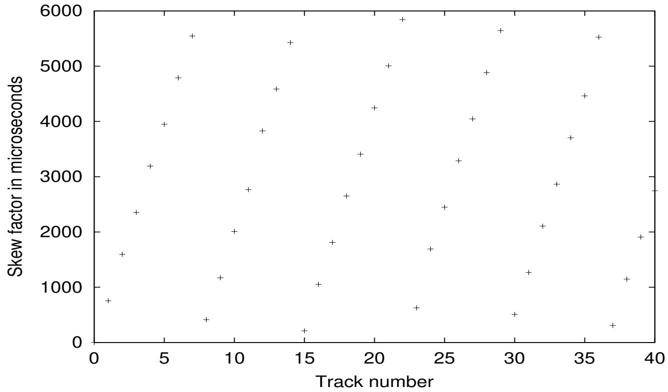

Figure 15: Skew factor s from Table 4 for ST39102LW.

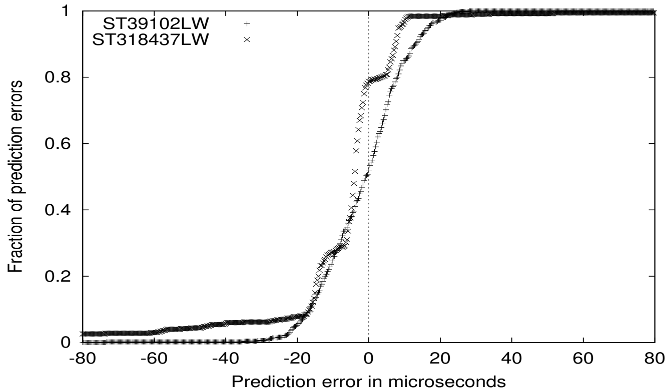

Figure 16: Rotational delay prediction accuracy for ST39102LW and ST318437LW.

in Equation 10) for each track. Sample results for extracted skew factors are presented in Table 4, wherein we also present the LBAs for blocks residing on the same disk radius.

In Figure 15, we plot the skew times against track numbers. We note that there is a distinct trend in skew times, which enables us to compress this information effectively and to reduce its space requirement. In Figure 15, we also note that there is a slight deviation of the common trend for tracks 12, 24 and 36. This is due to cylinder skew which occurs when the next track falls on an adjacent cylinder instead of the same cylinder. The experimental disk had precisely 12 surfaces. Hence, there is a trend deviation on tracks that are a multiple of 12.

Based on Equation 13, and the compressed information about skew times above, we were able to predict the rotational delay between two disk accesses. In Figure 16, we present the error distribution in our rotational prediction for a large number of random request pairs. We note that for the SMP-like testbed (testbed 1), which has a very predictable distribution of OS delay variations (Figure 4), our prediction is accurate within 25 μs for 99% of the requests. Even for the workstation-like testbed (testbed 2), which has less predictable OS delay variations (Figure 5), our prediction is accurate within 80 μs for 99% of the requests. We thus conclude that with detailed disk parameters, systems can implement very accurate mechanisms for predicting rotational delay.

4.6 Read Buffer

Figures 17 and 18 present results for the extraction of the prefetch segment size, using the method explained in Section 3.6.1. Both disks stop prefetching when they fill up the first prefetching segment. The extracted sizes for testbeds 1 and 2 are 561 and 204 blocks respectively. We also note that both disks continue prefetching into the next available prefetch segment, when they detect a long sequential access.

Using the extracted prefetch segment size, we can find out the number of prefetch segments in the disk read cache, as explained in Section 3.6.2. Using this method, we detect three cache segments for the testbed 1 disk, and 16 segments for the testbed 2 disk.

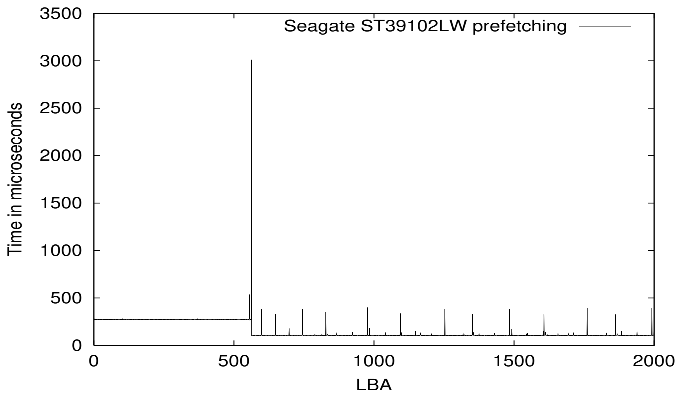

Figure 17: Read request completion times for ST39102LW.

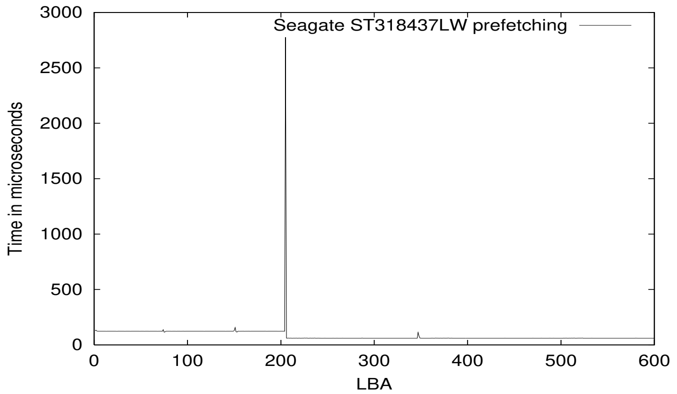

Figure 18: Read request completion times for ST318437LW.

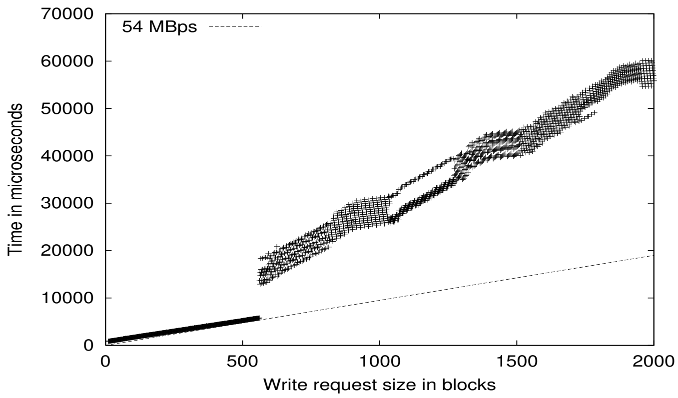

Figure 19: Write request completion times for ST39102LW.

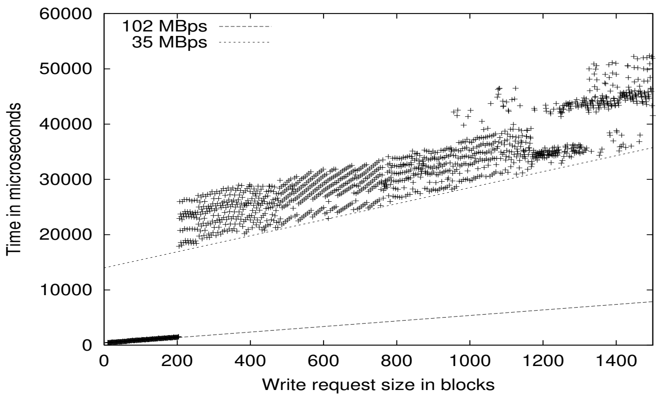

Figure 20: Write request completion times for ST318437LW

4.7 Write Buffer

Figures 19 and 20 present results for write buffer size extraction method explained in Section 3.7.

The write buffer size for testbed 1 and testbed 2 were measured to be 561 and 204 blocks respectively. If we compare these results to the results for prefetch segment size extraction presented in Section 4.6, we can see that both disks use one cache segment as a write buffer.

Using these measurements we also extract write throughputs, both to the disk write buffer (slope of the curve for write request sizes that fit into write buffer), and to the disk platter (when the request size is greater than the write buffer size). In the future we plan to extract the number of cache segments that can be used as write buffers. We assume that we can use a method similar to the method we used for detecting the number of read cache segments in Section 3.6.2.

5. RELATED WORK

Several previous studies have focused on the problem of disk feature extraction. The studies of [12, 9] extract disk features in order to model disk drives accurately. Both studies rely on interrogative SCSI commands to extract the LBA-to-PBA mapping from the disk. The study of [11] presents several methods for empirical feature extraction, including an approximate mapping extraction method. In [1], the authors propose methods for obtaining detailed temporal characteristics of disk drives. This study also includes a method for predicting rotational delay. The studies of [4, 13, 3, 5] propose scheduling algorithms which assume the ability to predict the rotational delay between successive requests to the disk.

Our study includes both interrogative and empirical methods for the accurate extraction of disk mapping information. The empirical methods can be used for disks that do not support interrogative commands in order to extract disk mapping. Using empirical extraction, we can obtain accurate disk mapping information, including precise positions of track and cylinder boundaries. Scsibench uses an approach for predicting rotational latency, similar to the study of [1]. However, we concentrate on predicting rotational distance between two LBAs, as explained in Section 3.5 and Section 4.5. Such prediction capability can be used for scheduling requests in real systems, which are known to arrive in a bursty fashion [7]. Scsibench can extract all the disk features that are required in order to be able to predict rotational delay with high accuracy.

Scsibench is open source and available for download [2]. At this time, we are not aware of any other disk feature extraction tool which is currently available for download, or as open source, and runs on a widely available operating system without any kernel modifications.

6. CONCLUSIONS

In this paper, we presented Scsibench, a user-level tool for SCSI disk feature extraction. Scsibench uses both interrogative and empirical methods to extract disk features. The empirical methods extract accurate track and cylinder boundaries, track and cylinder skew times, the number of tracks per cylinder, and the track sizes (in logical blocks).

Scsibench can also extract disk features necessary for predicting the rotational delay between any two blocks on the disk. This prediction was found to be highly accurate upon validation.

We believe that this work can be used by system and application programmers to improve disk performance. Using knowledge about disk features provided by Scsibench, system or application programmers can fine-tune disk accesses to match application requirements.

7. REFERENCES

[1] M. Aboutabl, A. Agrawala, and J.-D. Decotignie. Temporally determinate disk access: An experimental approach. *Univ. of Maryland Technical Report CS-TR-3752*, 1997.

[2] Z. Dimitrijevic, D. Watson, and A. Acharya. Scsibench. *http://www.cs.ucsb.edu/~zoran/scsibench/*, March 2000.

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

[4] D. M. Jacobson and J. Wilkes. Disk scheduling algorithms based on rotational position. *HPL Technical Report*, February 1991.

[5] C. R. Lumb, J. Schindler, G. R. Ganger, and D. F. Nagle. Towards higher disk head utilization: Extracting free bandwith from busy disk drives. *Proceedings of the OSDI*, 2000.

[6] M. McKusick, W. Joy, S. Leffler, and R. Fabry. A fast file system for unix*. *ACM Transactions on Computer Systems 2*, 3:181–197, August 1984.

[7] C. Ruemmler and J. Wilkes. UNIX disk access patterns. *Usenix Conference*, pages 405–420, Winter 1993.

[8] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *Computer*, 2:17–28, 1994.

[9] J. Schindler and G. R. Ganger. Automated disk drive characterization. *CMU Technical Report CMU-CS-00-176*, December 1999.

[10] Seagate. Scsi interface, product manual 2. *http://www.seagate.com/support/disc/manuals/scsi/38479j.pdf*, April 1999.

[11] N. Talaga, R. H. Arpaci-Dusseau, and D. Patterson. Microbenchmark-based extraction of local and global disk characteristics. *UC Berkeley Technical Report*, 1999.

[12] B. Worthington, G. Ganger, and Y. Patt. Online extraction of scsi disk drive parameters. *Proc. of ACM Sigmetrics Conference*, pages 146–156, 1995.

[13] B. L. Worthington, G. R. Ganger, and Y. N. Patt. Scheduling algorithms for modern disk drives. *Proceedings of the ACM Sigmetrics*, pages 241–251, May 1994.

APPENDIX

A. SCSIBENCH TRACE COMMANDS

Scsibench currently implements the following trace commands:

- *write command:* [R|W] $<start>$ $<size>$, reads or writes $<size>$ disk blocks starting from LBN $<start>$;
- *seek command:* S $<dest>$, seeks to track where block $<dest>$ is located;
- *handle disk buffers:* B $<cache\_flags>$, turns on or off disk read and write buffers and prefetching.
- *wait idle:* I $<delay>$, makes Scsibench to be idle for $<delay>$ in microseconds;
- *sample time:* T $<r_0>$, Samples current time to Scsibench time register with number $<r_0>$;
- *handle time registers:* [+|−] $<r_0>$ $<r_1>$ $<r_2>$, adds or subtracts $<r_1>$ and $<r_2>$, and stores result to $<r_0>$;
- *print time register:* P $<r_0>$, prints value stored in time register in microseconds.