Proceedings of the ACM Sigmetrics Conference, May 1995.

On-Line Extraction of SCSI Disk Drive Parameters

Bruce L. Worthington[†], Gregory R. Ganger[†], Yale N. Patt[†], John Wilkes[‡]

[†] Dept. of EECS, Univ. of Michigan
{worthing,ganger,patt}@eecs.umich.edu

[‡] Hewlett-Packard Laboratories
wilkes@hpl.hp.com

Abstract

Sophisticated disk scheduling algorithms require accurate, detailed disk drive specifications, including data about mechanical delays, on-board caching and prefetching algorithms, command and protocol overheads, and logical-to-physical block mappings. Comprehensive disk models used in storage subsystem design require similar levels of detail. We describe a suite of general-purpose algorithms and techniques for acquiring the necessary information from a SCSI disk drive. Using only the ANSI-standard interface, we demonstrate how the important parameter values of a modern SCSI drive can be determined accurately and efficiently.

1 Introduction

The magnetic disk drive remains firmly established as the preferred component for secondary data storage. Given the growing disparity between processor and disk speeds, achieving high system performance requires that disk drives be used intelligently. Previous work has demonstrated that disk request scheduling algorithms can significantly reduce seek and rotational latency delays [Jaco91, Selt90], and scheduling algorithms that exploit the on-board disk cache can further boost performance [Wort94]. Such algorithms require detailed knowledge of disk characteristics, including mechanical delays, on-board caching and prefetching algorithms, command and protocol overheads, and logical-to-physical data block mappings. Comprehensive disk models utilized in storage subsystem research require similar levels of detail [Ruem94].

Information about disk drives can be obtained by reading technical reference manuals, directly monitoring disk activity at the hardware level, and/or talking directly to the drive firmware developers and mechanical engineers. Unfortunately, disk technical manuals are typically incomplete and their accuracy is not guaranteed; direct observation requires expensive signal-tracing equipment and a great deal of time; and it is difficult to reach the correct people at disk drive company XYZ and convince them to release what may be perceived as sensitive data. This paper explains how to extract the required information directly from the on-line drives themselves. In particular, we focus on disks that conform to the popular Small Computer Systems Interconnect [SCSI93].

In some cases, we simply use standard SCSI commands to request parameter values from the disk. Such values are typically static in nature (or nearly so). Although such *interrogative extraction* is convenient, it has two drawbacks: (1) Because the ANSI standard allows a disk drive manufacturer to implement a subset of the full SCSI specification, a given disk drive may not support all interrogative commands. (2) The parameter values returned may be inaccurate or represent nominal figures (e.g., the mean number of sectors per track on a multi-zone disk).

In cases where interrogation cannot reliably obtain the information, such as for timing-related parameters (e.g., seek curves), the host acquires the values by monitoring disk behavior. Our *empirical extraction* techniques exercise a drive using carefully designed sequences of disk requests, or *test vectors*. Parameter values are then deduced from an analysis of the corresponding request service times. Much of this paper is concerned with the selection of appropriate test vectors. Our algorithms employ test vectors that allow quick and accurate extraction of parameter values despite the complexity of modern SCSI disk drive controllers.

Depending on the purpose of the extraction, test vectors can be tuned for higher efficiency or greater precision. Our baseline test vectors produce accurate parameter values within minutes. Alternately, they can be reconfigured to execute in considerably less time (i.e., a few seconds) at the cost of less reliable results. We have tested our extraction techniques on several SCSI disk drives and confirmed that accurate disk models can be constructed from the measured data. Our detailed disk simulator closely matches real disk behavior when configured with the extracted parameters.

The remainder of this paper is organized as follows. Section 2 contains a brief description of modern SCSI disk drives. For a more detailed discussion, the reader is referred to [Ruem94]. Section 3 identifies the system facilities needed for parameter extraction and describes our experimental setup. Sections 4 and 5 describe interrogative and empirical extraction techniques, respectively. Section 6 validates the extraction techniques. Section 7 discusses how disk request schedulers can utilize extracted data. Section 8 contains concluding remarks and discusses future enhancements.

2 Modern SCSI disk drives

2.1 Data layout

A disk drive consists of a set of rapidly rotating platters (on a common axis) coated on both sides with magnetic media. Data blocks on each *surface* are written in concentric *tracks*, which are divided into *sectors*. A *cylinder* is a set of tracks (one from each surface) equidistant from the center of the disk. Longer tracks (i.e., those farther out from the axis) can contain more data. As a result, many drives partition their set of cylinders into multiple *zones*. The number of sectors per track increases from innermost zone to outermost zone.

Defects detected during the course of a disk's lifetime require data layout adjustments. Spare regions are reserved to allow for defect *slipping* and *reallocation*. Sector or track slipping takes place during the formatting process. In the case of sector (track) slipping, the formatting process skips each defective sector (track) when initializing the data layout mapping. Each slipped region changes the mapping of subsequent logical blocks. Defect reallocation (a.k.a. *remapping* or *sparing*) occurs when defective sectors (tracks) are discovered during normal disk use. The disk dynamically remaps affected data blocks to spare regions and redirects subsequent accesses accordingly.

2.2 Mechanical components

A disk's read/write heads are held "over" a specific cylinder by a set of disk arms ganged together on a common actuator. Most SCSI disks allow only one active read/write head at any given time. A *seek* moves the disk arms to a new cylinder. The time spent waiting for the target sector(s) to rotate around to the active read/write head is called *rotational latency*. Depending on the type of servo information used, switching active heads may require a slight repositioning of the disk arms. The data layout is designed to reduce the performance impact of such *head switches* by offsetting the logical-to-physical mapping between logically sequential tracks. The offset, or *skew*, prevents a request that crosses a track or cylinder boundary from "just missing" the next logical block and waiting most of a full rotation for it to come around again.

2.3 Disk controllers

A modern SCSI disk uses one or more embedded controllers to decode and service SCSI *commands*. The disk maps a simple linear address space of logical blocks to physical media locations. A host issues a data request in terms of a starting *logical block number* (LBN) and a total request size. The details of servicing the request are typically hidden from the host. This offloads most of the management overhead associated with actual data storage from the host (or intermediate controller). As a result, entities outside of the drive itself often have little or no knowledge of the data layout on the media, the status of the on-board disk cache, and the various overheads associated with servicing a request.

Some SCSI commands, such as those used by our interrogative extraction techniques, are used to access control and configuration information (e.g., MODE SENSE, MODE SELECT, and SEND/RECEIVE DIAGNOSTIC). Other commands, typically used to access data (e.g., READ and WRITE), can be used to empirically extract performance characteristics.

2.4 Disk caches

Disk drives originally used on-board memory to buffer (i.e., speed-match) media and bus transfers. Modern SCSI drives also use on-board memory as a data cache. *Segmented* disk caches consist of multiple independent cache lines, or *segments*. On-board disk logic can automatically *prefetch* data to more quickly satisfy sequential read requests. Disks with *read-on-arrival* (or *write-on-arrival*) can access blocks in the order they pass under the read/write head rather than in ascending LBN order.

2.5 Extraction complications

Disk behavior can appear unpredictable for many reasons. Our extraction techniques handle:

- overlapping disk controller overheads, SCSI bus data transfers, and mechanical delays;
- contention for shared resources along the I/O path (e.g., buses and intermediate controllers);
- segmented on-board caches;
- simple or aggressive prefetching algorithms;
- non-uniform performance characteristics (e.g., actuator positioning algorithms optimized for very short seek distances);
- large, seemingly non-deterministic delays (e.g., thermal recalibration);
- minor fluctuations in timing.

Parameter	DEC RZ26	Seagate ST41601N	HP C2490A
Capacity	1.03 GB	1.37 GB	2.13 GB
RPM	5400	5400	6400
Diameter	$3\,{}^1\!/_2''$	$5\,{}^1\!/_4''$	$3\,{}^1\!/_2''$
Surfaces	14	17	18
Cylinders	2570	2098	2582
Zones	1	14	11
Sectors/Track	58	61–85	68–108

Table 1: Basic parameters for our experimental disks.

3 Extraction environment

Our extraction methodology requires a host computer system with direct, low-level SCSI access (bypassing any file system or other translation) and a high-resolution timer to measure the service times of individual disk requests. Some systems (such as HP-UX [Clegg86]) have built-in monitoring and trace-points that provide the necessary timing capability. In general, our extraction techniques require access to the O/S source code, a device driver development environment, or an unprotected I/O card (e.g., on a machine running MS/DOS).

The host system in our experimental setup is an NCR 3550 symmetric multi-processor running an MP-safe version of SVR4 UNIX[1]. Arbitrary SCSI commands can be issued directly to specific disk drives using a library of SCSI-related functions provided by NCR. To obtain fine-grained timing data, we modified the device driver to measure service times using a diagnostic counter with a resolution of 840 nanoseconds. During the extraction process, we keep our host system as idle as possible to reduce algorithm execution time. Although our algorithms produce valid results in the presence of timing noise (e.g., additional CPU load and bus contention), they take longer to do so.

We developed and tested our extraction techniques using three disk drives from different manufacturers: a DEC RZ26, a Seagate ST41601N [Seag92, Seag92a], and an HP C2490A [HP93]. Table 1 lists some basic characteristics of these drives.

4 Interrogative extraction

SCSI disk drives can supply a number of parameter values upon request. Most disks implement both ANSI-standard and vendor-specific methods for requesting configuration information. Unless otherwise noted, our techniques use only ANSI-standard features supported by all of the modern SCSI disks with which we are familiar.[2]

[1]UNIX is a registered trademark of X/Open Corporation.

[2]We have accumulated a broad range of disk reference manuals from various manufacturers, including DEC, Fujitsu, Hewlett-Packard, Maxtor, Micropolis, Quantum, and Seagate.

Mode Page & Parameter	Location
Disconnect-Reconnect (0x02)	
Buffer Full Ratio	B2
Buffer Empty Ratio	B3
Bus Inactivity Limit	B4-B5
Disconnect Time Limit	B6-B7
Connect Time Limit	B8-B9
Maximum Burst Size	B10-B11
Data Transfer Disconnect Control	B12b0-b1
Format Device (0x03)	
Tracks Per Zone[ab]	B2-B3
Alternate Sectors Per Zone[ab]	B4-B5
Alternate Tracks Per Zone[ab]	B6-B7
Alternate Tracks Per Logical Unit[a]	B8-B9
Sectors Per Track[a]	B10-B11
Data Bytes Per Physical Sector[a]	B12-B13
Interleave[ac]	B14-B15
Track Skew Factor[a]	B16-B17
Cylinder Skew Factor[a]	B18-B19
Surface Mode[d]	B20b4
Rigid Disk Geometry (0x04)	
Number of Heads	B5
Medium Rotation Rate	B20-B21
Caching (0x08)	
Read Cache Disable	B2b0
Multiplication Factor	B2b1
Write Cache Enable[d]	B2b2
Disable Prefetch Transfer Length	B4-B5
Minimum Prefetch	B6-B7
Maximum Prefetch	B8-B9
Maximum Prefetch Ceiling	B10-B11
Notch (0x0C)	
Logical Or Physical Notch[a]	B2b6
Maximum Number of Notches	B4-B5
Active Notch	B6-B7
Starting Boundary[a]	B8-B11
Ending Boundary[a]	B12-B15

[a] Active Notch must be set for per-notch values.
[b] For this parameter, "Zone" refers to a region over which spare sectors are allocated.
[c] Must be zero or one for our extraction techniques.
[d] Must be zero for our extraction techniques.

Table 2: Some relevant Mode Page parameters.

SCSI *Mode Pages* allow on-line access to various disk drive configuration parameters. Table 2 lists some relevant Mode Page parameters.[3] Each Mode Page holds a specific type of information (e.g., Caching, Rigid Disk Drive Geometry, Read-Write Error Recovery, etc.). The

[3]Hexadecimal values are prefixed with "0x", byte offsets with "B", and bit offsets with "b". Byte and bit offsets start at zero.

MODE SENSE command returns the contents of a requested Mode Page. The MODE SELECT command attempts to overwrite the contents of a specific Mode Page.[4]

Unfortunately, Mode Page information may be misleading. In some cases, the standard SCSI Mode Pages cannot adequately reflect the behavior of the disk controller. In other cases, the manufacturer may have decided that a nominal value would suffice. As a result, we use empirical extraction methods (see section 5) to verify data acquired via interrogative extraction.

4.1 Data layout parameters

An accurate logical-to-physical mapping of individual data blocks is vital for disk scheduling and modeling efforts. Our current extraction methodology relies heavily on the READ DEFECT DATA command and the TRANSLATE ADDRESS form of the SEND/RECEIVE DIAGNOSTIC commands. Alternately, most of the mapping information can be obtained from the Format Device and Notch Mode Pages (see table 2).

4.1.1 READ DEFECT DATA

Previous work has shown that scheduling algorithms and disk models can safely ignore reallocated defects [Wort94]. Slipped defects, on the other hand, can cause major data layout perturbations (especially in the case of track-based slipping). As reallocated defects may be converted into slipped defects during the formatting process, data layout extraction must be repeated after reformatting. READ DEFECT DATA requests a list of defective sectors (or tracks) from the disk in cylinder/head/sector (or cylinder/head) form. To obtain the complete set of defects, the Primary Defect List Bit (B2b4) and Grown Defect List Bit (B2b3) should both be set in the READ DEFECT DATA request.[5]

4.1.2 SEND/RECEIVE DIAGNOSTIC

The TRANSLATE ADDRESS form of SEND DIAGNOSTIC requests the exact logical-to-physical mapping for a single logical block (to be returned by a subsequent RECEIVE DIAGNOSTIC). Obtaining a full logical-to-physical map by iterating through every logical block number can take several hours. We use a variety of shortcuts to reduce the number of SEND/RECEIVE DIAGNOSTIC pairs required to obtain the mapping information. Our extraction algorithms first determine the boundaries, sectors per track, track skew, and cylinder skew for each zone (or verify this information if already obtained from the Mode Pages). Mapping information for the "expected" first and last logical block of each cylinder is then requested.[6] Mismatches between actual and expected values indicate the existence of slipped defects or spare regions, which can be pinpointed with additional SEND/RECEIVE DIAGNOSTIC pairs.

Of all the manufacturers for which we have disk specifications, only Quantum does not support the TRANSLATE ADDRESS form of SEND/RECEIVE DIAGNOSTIC. Since Quantum drives **do** implement the Notch Mode Page and the full READ CAPACITY command, it is possible (albeit potentially slow) to automatically extract an accurate logical-to-physical map for these drives.

5 Empirical extraction

Although some disk parameters and characteristics can be obtained via interrogation, many must be deduced by observing disk behavior. This section describes our empirical extraction algorithms and their use in measuring disk performance characteristics. In particular, it contains descriptions of our general empirical extraction methodology, algorithms for extracting mechanical parameter values, algorithms for command and completion overhead extraction, techniques for characterizing cache behavior, and a simple algorithm to compute a raw transfer rate for the I/O path.

The test vectors used in our empirical extraction techniques range from tens to thousands of requests in length. As some disks must "warm-up" after they have been idle for some period of time, we prefix each test vector with 20–50 random disk requests and discard their service time measurements. Without this precaution, the first 10–20 requests may take up to ten times longer than normal to complete.

For modeling purposes, mean or 50th percentile parameter values are typically sufficient. In other cases, some (probabilistic) upper bound values are more appropriate. For example, a disk request scheduler configured with mean values could underpredict command overhead or disk arm positioning times, causing unnecessary rotational delays. For this reason, we also extract maximum or 95th percentile parameter values.

5.1 Minimum time between request completions

Many of our empirical extraction techniques rely on measuring the *minimum time between request completions* for specific pairs of request types. *MTBRC(*X,Y*)* denotes the minimum time between the completions of a request of type X and a request of type Y. For example, the MTBRC for a 1-sector write request on one cylinder followed by a 1-sector read request on an adjacent cylinder is referred to as MTBRC(1-sector write, 1-sector read requiring a 1-cylinder seek).

[4] Some Mode Page fields are read-only and cannot be overwritten.

[5] The Grown Defect List contains all defects detected after manufacture, both reallocated and slipped.

[6] Checking every cylinder may be excessive for some disks (e.g., those using track sparing).

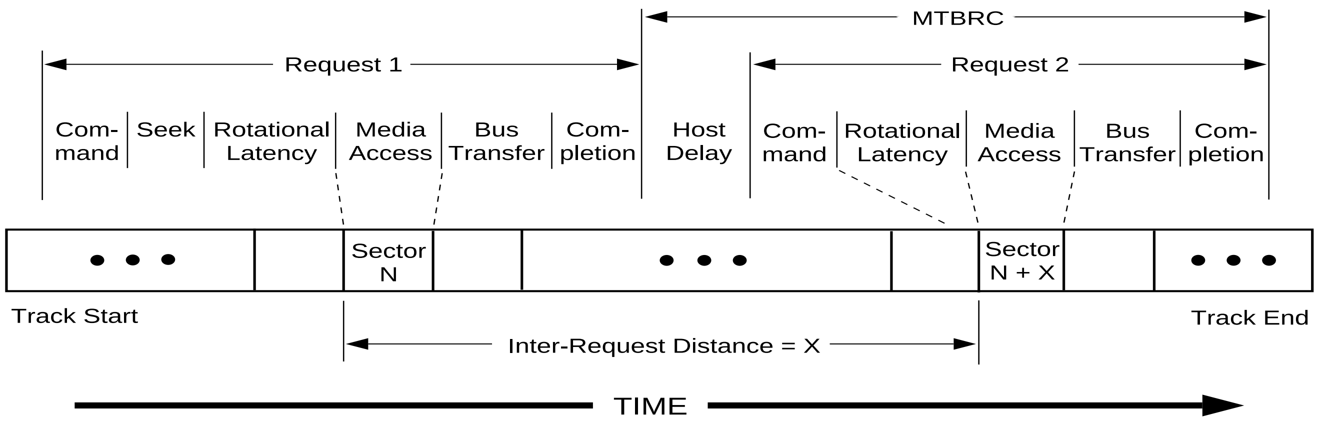


Figure 1: MTBRC example. The time-line shows the service time components for an example MTBRC(1-sector READ, 1-sector READ on same track) request pair. The track accessed by the two requests is also shown (as a straight line rather than as a circle). Timestamps are taken at the initiation and completion of each request, providing the host delay and the MTBRC timings. The inter-request distance is taken from the starting locations of the requests. Note that request 2 has less than one sector of rotational latency.

Extracting an MTBRC is an iterative process, wherein the rotational distance between the request pairs is varied until a minimum is reached. This essentially eliminates rotational latency from the measurement. The appendix provides a detailed algorithm for extracting MTBRC values.

When extracting an MTBRC value, we also measure the mean *host delay* and the *inter-request distance*. The host delay is simply the time between the **completion** of the first request and the **initiation** of the second. This delay is due solely to processing time at the host (e.g., executing device driver code, configuring intermediate controllers, and performing memory-to-memory copies). The inter-request distance is the rotational distance (in sectors) between the physical starting locations of each MTBRC request pair. Given the rotation speed and the number of sectors per track, the inter-request distance can be converted into a measurement representing the time from the beginning of the first media transfer to the beginning of the second media transfer. Figure 1 shows the service time components for an MTBRC request pair, indicating which components contribute to the MTBRC value, the mean host delay, and the inter-request distance.

When using MTBRCs to extract a given parameter, we collect MTBRC values for two (or more) pairs of request types. Each MTBRC value provides an equation with one or more unknown variables. The parameter of interest is then determined via simple algebraic techniques. In Section 5.4, for example, we extract the effective head switch time from MTBRC(1-sector write, 1-sector read on the same track) and MTBRC(1-sector write, 1-sector read on a different track of the same cylinder). The resulting equations are:

$$MTBRC_1 = Host_1 + Cmd + Media + Bus + Comp$$
$$MTBRC_2 = Host_2 + Cmd + HdSw + Media + Bus + Comp$$

Via substitution and re-organization, we get:

$$HdSw = (MTBRC_2 - Host_2) - (MTBRC_1 - Host_1)$$

Because all values on the right are measured, the effective head switch time is easily calculated. The mean host delays are measured and subtracted from each MTBRC value (as shown above) to remove any variation in host processing times.

5.2 Test vector considerations

Test vectors must be carefully designed to prevent unexpected mechanical delays and/or interference from the on-board cache (even when it is only functioning as a speed-matching buffer):

- Any write-back (a.k.a. *fast-write*) activity should be disabled during extraction (e.g., via the Write Cache Enable Bit listed in table 2).
- When READ misses are required, the cache must be initialized by issuing as many random READs as there are cache segments.
- As prefetching may result in unwanted switching between tracks (or cylinders), test vectors should use WRITE requests or choose initial READ targets from the lowest LBNs on a track. The latter option depends on short host delays.
- The overlap between media and bus transfers can be eliminated by using 1-sector requests or setting the Buffer Ratio fields (see table 2), provided that the disk controller utilizes these values.

5.3 Seek curve

A *seek curve* graphs seek time as a function of seek distance. The seek time between two cylinders is a complex function of the position of the cylinders (i.e., their distance from the spindle), the direction of the seek (inward or outward), various environmental factors (e.g., vibration and thermal variation), the type of servo information (e.g., dedicated, embedded, or hybrid), the mass and flexibility of the head-arm assembly, the current available to the actuator, etc. Our algorithms extract "maximum" and mean seek curves for use in request schedulers and disk models, respectively.

Given an accurate logical-to-physical map (see section 4.1), the SEEK command can be used to obtain seek curves. Figure 2 shows seek curves for the Seagate test disk, extracted using the following algorithm:

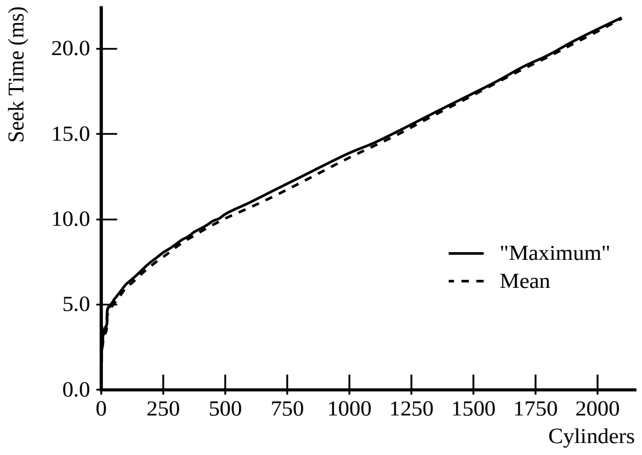
(1) For each seek time to be extracted, select 5 starting points evenly spaced across the physical cylinders.[7] From each starting point, perform 10 inward and 10 outward SEEKs of the appropriate distance. Disregard excessive service times (such as those experienced during thermal recalibrations).

(2) Average the 20 measured service times to obtain a single service time value for each starting point. For a mean seek curve, use the mean of the service times extracted at the 5 points. For a "maximum" seek curve, use the maximum of the 5 values.

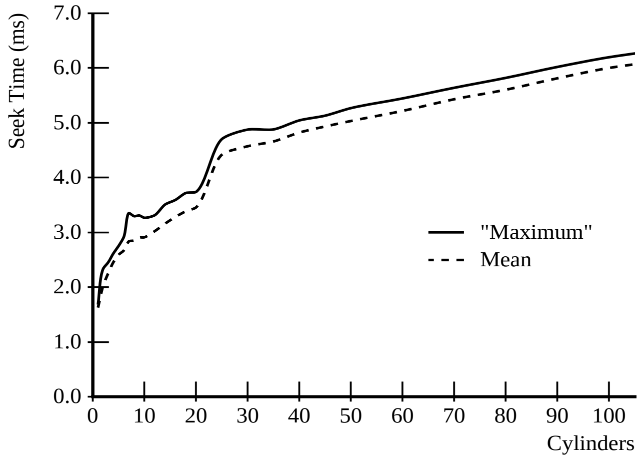
(3) Measure MTBRC(1-sector write, 1-sector read on same track) and MTBRC(1-sector write, 1-sector read incurring a 1-cylinder seek). The difference between these two values represents the time necessary to mechanically seek across one cylinder. The difference between this value and the extracted 1-cylinder SEEK service time represents the non-mechanical overheads associated with processing a SEEK request. This processing delay should be subtracted from each SEEK service time, producing a pair of seek curves usable for disk modeling or request scheduling purposes.

As the seek distance increases, seek curves become linear. Therefore, the majority of seek times for larger seek distances can be computed using linear extrapolation. The seek curves in figure 2 contain seek times for every seek distance between 1 and 10 cylinders, every 2nd distance up to 20 cylinders, every 5th distance up to 50 cylinders, every 10th distance up to 100 cylinders, every 25th distance up to 500 cylinders, and every 100th seek distance beyond 500 cylinders. Using this method, the extraction process takes less than two minutes for each of the three test disks. By reducing the number of starting points, repetitions, and/or discrete seek distances, the extraction time can easily be reduced.

[7]For the longest seek distances, only starting points at the innermost and outermost edges of the disk can be used.



(a) Full seek curve



(b) Expanded view of short seeks

Figure 2: Extracted seek curves for Seagate ST41601N

5.4 Head switch and write settling

The time necessary to switch read/write heads depends primarily on the type of servo information utilized by the disk drive (e.g., dedicated, embedded, or hybrid) and various environmental factors (e.g., vibration and thermal variation). For the current generation of disk drives, any seek activity will typically subsume the head switch time. WRITE requests incurring head switches (or seeks) may require additional settling time to more closely align the read/write heads.[8] Track skew can be used as an upper bound on the combined head switch and write settling delays.

We define the *effective head switch time* as the portion of the head switch that is not overlapped with command processing or data transfer. This time can be

[8]Data can be retrieved from the media successfully even when the read/write heads are slightly misaligned, but data must be written as close to the middle of a track as possible in order to prevent interference between adjacent tracks of data.

Disk parameter		DEC RZ26	HP C2490A
Head switch	Effective	0.698 ms	0.996 ms
	Published	not avail.	< 1.0 ms
Write settle	Effective	-0.446 ms	-0.631 ms
	Published	not avail.	0.750 ms
Minimum track skew		1.544 ms	1.500 ms

Table 3: Head switch and write settling times. Minimum track skew times (across all zones) are provided for reference.

computed from MTBRC(1-sector write, 1-sector read on the same track) and MTBRC(1-sector write, 1-sector read on a different track of the same cylinder), as described in section 5.1.

We define the *effective write settling time* as the portion of the settling time that is not overlapped with command processing or data transfer. This time can be computed from the effective head switch time, MTBRC(1-sector write, 1-sector write on same track) and MTBRC(1-sector write, 1-sector write on a different track of the same cylinder). The two MTBRC values provide a sum of the effective head switch and write settling times. Effective write settling time can then be computed by subtracting the effective head switch time. The extracted effective write settling times are negative for two of the test disks, indicating that additional overlap occurs between head switching/write settling and command processing/bus transferring for WRITEs.[9] The Seagate test disk has negligible extracted effective head switch and write settling times, as it uses only dedicated servo information.[10] Table 3 presents extracted and published head switch and write settling times for the HP and DEC test disks.

5.5 Rotation speed

Rotation speed can be determined empirically by performing a series of 1-sector writes to the same location and calculating the mean time between request completions (the time per rotation). A small number of repetitions (e.g., 32) is sufficient to obtain an accurate value. Extracted rotation speeds for our three test disks are well within the manufacturer-specified tolerances.

[9]As long as the magnitude of the negative effective write settling time is greater than that of both the effective head switch and 1-cylinder seek times, a disk model or request scheduler should still be able to use the value. It is only used in combination with these other values, and the combined values will be positive.

[10]For disks using only dedicated servo information, a second effective write settling time should be extracted using 1-cylinder SEEKs instead of head switches.

Overhead	Value
Read Hit After Read	0.896 ms
Read Hit After Write	0.506 ms
Read Miss After Read	1.433 ms
Read Miss After Write	1.433 ms
Write After Read	2.059 ms
Write After Write	1.404 ms
Read Completion	0.232 ms
Write Completion	0.0 ms
Published Per-Command	0.700 ms

Table 4: Seagate ST41501N command overheads.

5.6 Command and completion overheads

Due to our "black-box" extraction methodology, it is difficult to accurately isolate certain important parameters. Command, completion, and other overheads can only be measured as combined quantities unless the extraction process has access to timing information for when the actual data transfers begin and end. We estimate command and completion overheads by solving multiple linear equations (obtained from MTBRC extractions) for the various unknown values.[11] For our disk model, we solve eight equations containing eight unknowns. Inconsistent equations sometimes suggest conflicting values for a particular overhead. In such cases, we use the mean of the conflicting values. We sometimes find it necessary to reduce the number of unknowns by setting one of them to zero (we typically choose a completion overhead).

The number of different overhead values extracted depends on the desired accuracy of the resulting model. Command and completion overheads can depend on the command (e.g., READ or WRITE), the state of the cache (e.g., hit or miss), and the immediately previous request's characteristics. To configure our disk simulator, we extract six command overheads and two completion overheads. Table 4 shows extracted and published overhead values for the Seagate disk drive. Note the significant variation between command overhead values.

5.7 On-board disk caches

The standard SCSI specification provides for a wide range of cache-management policies (and manufacturers are free to extend this range). This section describes our approach to characterizing on-board disk caches. Each cache extraction algorithm tests specific hypotheses about cache behavior. Test vectors are devised to prove or disprove the hypotheses. Boolean parameter values (e.g., can a READ hit on cached data from a pre-

[11]For disk schedulers, combined measurements can be used directly (see section 7).

vious WRITE?) typically require only one extraction test vector. When determining the number of cache segments, on the other hand, feedback from each disproven hypothesis helps to select the next hypothesis (and a new test vector). Designing a complete set of hypotheses is difficult (if not impossible); the on-board cache can only be characterized to the extent of the extraction algorithms' knowledge of possible cache characteristics.

5.7.1 Cache segments (number, type, and size)

Although modern SCSI disk drives typically contain Mode Page information regarding the number and size of on-board cache segments, such data are not required by the ANSI standard. For this reason, our techniques rely on empirical extraction of these parameters.

The following technique will determine the maximum number of cache segments available to READs:

Choosing the Hypothesis

(1) Set N, the hypothesized number of segments, to a large value (e.g., 32).[12]

Loading the Cache

(2) Perform 1-sector READs of the first logical blocks of the first N-1 data cylinders.

(3) Perform a 1-sector READ of the first logical block of the last data cylinder.

Testing the Hypothesis

(4) Perform a 1-sector READ of the first logical block from the first data cylinder. A READ hit will occur if the number of cache segments is N or greater. For our test disks, we record a READ hit if this request is serviced in less that 3/4 of a rotation. A longer service time indicates that mechanical activity was necessary to service this request. If a READ miss occurs, decrement N and start over from step 2.

If no segments are detected using the above algorithm, modify step 4 to read the second logical block from the first data cylinder. This will handle disks that "throw away" all requested data in favor of prefetched data. Also, the above algorithm is designed for caches with multiple segments. A simple modification to the last iteration will allow the extraction algorithm to detect the existence of a single cache segment.

Some on-board caches have dedicated segments for use by WRITEs, while others allow all segments to be used by READs or WRITEs. Additional test vectors can determine whether or not WRITEs appropriate one or more of the N READ segments.

Extracting the size of the cache segments can be quite simple or very difficult depending on the prefetching behavior of the disk. Our basic technique computes the segment size by filling a cache segment with data (e.g., via a large READ) and then determining the boundaries of the segment by detecting READ hits (or misses) against the cached data. We allow sufficient time for all prefetching activity to cease before probing the segment boundaries. Of course, the cache segment must be re-initialized after each "probe" request, as READ misses or additional prefetching may change the segment contents. The most difficult scenario involves caches that "throw away" all requested data in favor of (potentially) prefetched data. If the prefetching activity halts before filling the segment, the extraction algorithm may underestimate the size of the cache segments. For such disks, an attempt must first be made to maximize the prefetching activity by changing the appropriate Mode Page parameters.

Some disks dynamically modify the number (and size) of cache segments based on recent request patterns. It may be difficult to accurately characterize the on-board caching behavior of such disks.

[12]A relatively trivial modification to this algorithm allows it to successfully determine the maximum number of READ segments even if it is larger than the initially hypothesized value.

5.7.2 Buffer ratios

The Buffer Full Ratio and Buffer Empty Ratio values specified on the Disconnect-Reconnect Mode Page are needed for comprehensive disk modeling. They indicate how full a read cache segment should be and how empty a write cache segment should be before the disk attempts reconnection to the host. However, some disks utilize adaptive algorithms when these values are set to 0x00. To determine if this is the case, set the Buffer Full Ratio to 0x00 and compute MTBRC(1-sector write, segment-sized read) for the outermost zone. Repeat the extraction with Buffer Full Ratio set to 0x01. If the 0x00 MTBRC is significantly different from the 0x01 MTBRC (i.e., the 0x00 case is attempting to utilize bus resources more efficiently), an adaptive algorithm may be in use. Of the three test disks, our extraction techniques indicate that only the HP disk uses adaptive reconnection algorithms.

5.7.3 Prefetch

Table 2 includes several prefetch parameters obtainable from the Mode Pages. The empirical extraction of prefetching behavior follows the model of hypothesizing, devising a test vector, and detecting READ hits (or misses). The amount of prefetched data can be determined by an algorithm similar to the one used to compute the cache segment size (see above). Example boolean hypotheses include: Do prefetched data replace requested data in the current segment? Are all requested data always "thrown away"? Does prefetching stop on physical boundaries (e.g., tracks and cylinders)? Is the amount of prefetching activity related to the request size?

Disk drive	Path Transfer Rate	
	Published Max	Extracted
DEC RZ26	not avail.	7.24 MB/s
Seagate ST41601N	10.0 MB/s	5.36 MB/s
HP C2490A	10.0 MB/s	3.45 MB/s

Table 5: Path transfer rates. Note that extracted path transfer rates should **not** be used to compare the performance of the individual drives, since the bus and intermediate controller characteristics varied between disks in our experimental setup.

5.7.4 Miscellaneous

There are many additional hypotheses that our extraction techniques test. For example: Does the disk implement read-on-arrival? write-on-arrival? Is cache space allocated on a track- or sector-level basis? Can READs hit on data placed in the cache by WRITEs? For segmented caches, what is the segment replacement algorithm? As our understanding of cache management algorithms improves, we refine and augment our set of extraction hypotheses.

5.8 Path transfer rate

The data transfer rate between the disk and main memory is a function of the bandwidth of each component in the I/O data path (e.g., controllers, buses, adapters, and disks). Our extraction methodology determines a single value for the path transfer rate by comparing service times for different-sized READ requests that hit in the on-board disk cache. Table 5 presents extracted path transfer rates for our three test disk drives.

6 Model validation

To validate our techniques, we extracted a complete set of parameter values for each of the three experimental disks. We configured a detailed disk simulator with the extracted values to allow comparison between measured and modeled disk behavior. For all three disks, the mean service times differ by less than 1%.

Greater insight can be achieved by comparing the service time distributions [Ruem94]. Figures 3–5 show distributions of measured and simulated service times for equivalent validation workloads. Ruemmler and Wilkes define the root mean square horizontal distance between the two distribution curves as a *demerit figure* for disk model calibration. The demerit figures for the DEC, Seagate, and HP disks are 0.19 ms[13] (1.2% of the

[13]In validating our DEC disk extraction, we observed excessive service times for a few requests (28 out of 10000), probably caused by some infrequent disk firmware activity (e.g., recalibration). If these requests are **not** removed from the computation, the demerit figure is 0.67 ms (4.3% of the corresponding mean service time).

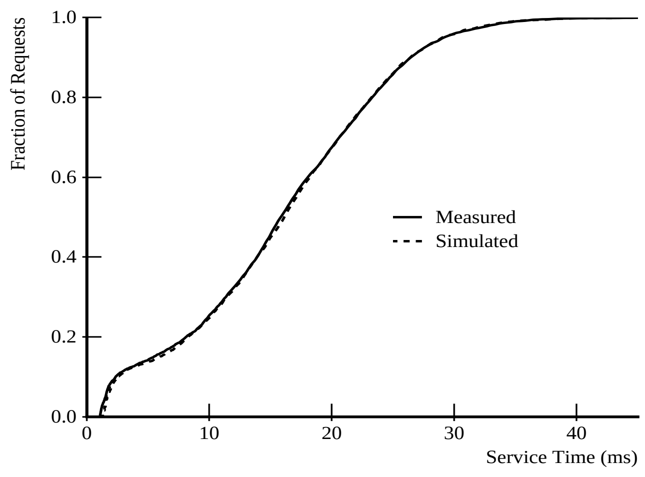


Figure 3: DEC RZ26 service time distributions.

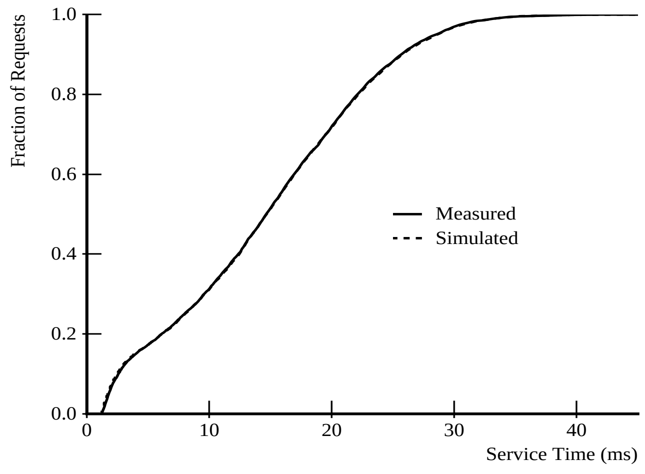


Figure 4: Seagate ST41601N service time distributions.

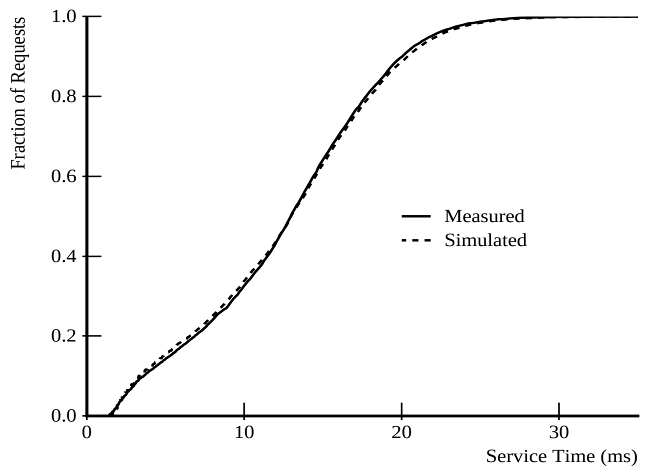


Figure 5: HP C2249A service time distributions.

corresponding mean service time), 0.075 ms (0.5% of the corresponding mean), and 0.26 ms (2.0% of the corresponding mean), respectively. These values approach those for the most aggressive (and accurate) models discussed by Ruemmler and Wilkes.

The very close match between the actual disks and the simulator configured with extracted parameters provides strong support for the accuracy of both the model and the extraction process. Also, the generality of our extraction techniques is supported by their success with these very different disk drives.

7 Use of extracted data in request schedulers

This section gives insight into how disk request schedulers can effectively use extracted data. Aggressive scheduling algorithms often attempt to reduce overall mechanical delays (i.e., combined seek and rotational latency). This reduction is achieved by dynamically reordering the queue(s) of pending requests. Such algorithms require accurate data layouts, seek curves, head switch and settling delays, and detailed knowledge of various command and completion overheads. In particular, the scheduler needs to know how soon the disk can perform media access after receiving a command. A request scheduler external to a disk must also know (or estimate) the disk's exact current rotational position.[14] This is difficult to deduce because of overlaps between command processing, media transfer, and bus transfer. The inter-request distances measured for MTBRCs provide an alternate method for accomplishing disk request scheduling.

Inter-request distances represent the minimum rotational distance between specific consecutive request pairs. That is, the second request of the pair is serviced with less than one sector of rotational latency. Therefore if the rotational distance between the requests is even one sector less, a rotational "miss" will occur.

If MTBRC extraction is performed using requests of a common size (e.g., the file system or database block size), a scheduler can use the associated inter-request distances to predict positioning delays for all pending requests. Each time a request of the specified size is serviced, the scheduler obtains a *reference point* that remains valid for a short period of time (e.g., several rotations). Reference points need to be re-established whenever possible due to fluctuations in rotation speed.

The computations necessary to predict positioning delays are best explained with an example:

(1) Assume the most recent reference point is an N-sector WRITE starting at physical sector 25. It completed exactly 1.10 rotations ago. There are 100 sectors per track in this zone. Therefore, the current reference point is equivalent to an N-sector WRITE starting at physical sector **35** (1.10 rotations beyond sector 25) that has just completed.

(2) The inter-request distance previously computed for MTBRC(N-sector WRITE, 1-sector READ on the same track)[15] is 0.20 rotations. A READ request to sector **55** (0.20 rotations beyond sector 35) would therefore incur (essentially) zero rotational latency.

(3) There are two READs in the pending queue, one starting at sector 85 and the other at sector 40. The first request would incur 0.30 rotations of latency (the distance between sectors 55 and 85). The second would incur 0.85 rotations of latency (the distance between sectors 55 and 40). Therefore, the first request should be scheduled for service.

If a pending request is not on the same track as the current reference point, additional steps are necessary to account for the appropriate seek, head switch, and/or write settling delays. If the scheduler has extracted cache parameters, it should give precedence to requests that will hit in the cache [Wort94].

8 Conclusions and future work

Aggressive disk scheduling and accurate performance modeling both require thorough, quantified understanding of disk drive behavior. We have described a set of general techniques for efficiently gathering the necessary information from modern SCSI disk drives. Interrogative extraction algorithms "ask" the drive to provide desired parameter values directly. Empirical extraction algorithms deduce performance characteristics from measured service times for specific sequences of disk requests. Both methods are needed to obtain the requisite data quickly and accurately. Using a detailed disk simulator configured with extracted parameter values, we have demonstrated the accuracy of our extraction methodology.

Because our empirical extraction techniques rely on host-observed request service times, they have two major limitations: (1) It is often not possible to distinguish disk performance characteristics from those of other I/O path components. (2) SCSI bus utilization can not be extracted, as our techniques have no way of determining when disconnect/reconnect activity occurs. Other sources (e.g., additional trace-points or a SCSI bus analyzer) are needed to acquire this information.

Our techniques rely heavily on the TRANSLATE ADDRESS form of the SEND/RECEIVE DIAGNOSTIC commands to obtain full logical-to-physical data mappings. We do not currently have an automated extraction technique for disks without this functionality. We believe

[14] SCSI disks often include a *spindle synchronization* signal that would allow a scheduler to track rotational position using specialized hardware support.

[15] Note that the size of the second MTBRC request type does not have to match the size of the pending request, as request size does not affect initial positioning delay.

that alternative methods can be devised to extract accurate mapping information for such drives (e.g., using the READ CAPACITY command and the Notch Mode Page).

Modern SCSI drives typically support *command queueing* at the disk (i.e., multiple pending commands). We plan to extend our suite of extraction techniques to gather information about command queueing, including the scheduling algorithm employed and the levels of concurrency allowed between the command processing, bus transfer, and media access of multiple "in-flight" commands.

9 Acknowledgements

We thank John Fordemwalt, Richie Lary, and Chris Ruemmler for their insights on various disk drive characteristics. We thank Jim Pike of AT&T/GIS for providing the initial funding for research into the I/O subsystem at the University of Michigan. We also acknowledge our research groups at the University of Michigan and Hewlett-Packard Laboratories. In particular, Bernard Yeh assisted in our validation efforts. Finally, the University of Michigan research group is very fortunate to have the financial and technical support of several industrial partners, including Hewlett-Packard, AT&T/GIS, Intel, Motorola, Scientific and Engineering Software, and Digital Equipment Corporation.

References

[Clegg86] F. Clegg, G. Ho, S. Kusmer, J. Sontag, "The HP-UX operating system on HP Precision Architecture computers", *Hewlett-Packard Journal*, 37 (12), December 1986, pp. 4-22.

[HP93] Hewlett-Packard Company, "HP C2490A 3.5-inch SCSI-2 Disk Drives, Technical Reference Manual", Part Number 5961-4359, Edition 3, September 1993.

[Jaco91] D. Jacobson, J. Wilkes, "Disk Scheduling Algorithms Based on Rotational Position", Hewlett-Packard Technical Report, HPL-CSP-91-7, Feb. 26, 1991.

[Ruem94] C. Ruemmler, J. Wilkes, "An Introduction to Disk Drive Modeling", *IEEE Computer*, March 1994, pp. 17-28.

[SCSI93] "Small Computer System Interface-2", ANSI X3T9.2, Draft Revision 10k, March 17, 1993.

[Seag92] Seagate Technology, Inc., "SCSI Interface Specification, Small Computer System Interface (SCSI), Elite Product Family", Document #64721702, Revision D, March 1992.

[Seag92a] Seagate Technology, Inc., "Seagate Product Specification, ST41600N and ST41601N Elite Disc Drive, SCSI Interface", Document #64403103, Revision G, October 1992.

[Selt90] M. Seltzer, P. Chen, J. Ousterhout, "Disk Scheduling Revisited", *Winter USENIX*, 1990, pp. 313-324.

[Wort94] B. Worthington, G. Ganger, Y. Patt, "Scheduling Algorithms for Modern Disk Drives", *SIGMETRICS*, 1994, pp. 241-251. [An extended version: "Scheduling for Modern Disk Drives and Non-Random Workloads", U. of Michigan, Technical Report CSE-TR-194-94, 1994.]

Appendix MTBRC Extraction

We obtain 50th and 95th percentile **MTBRC(X,Y)** values in the following manner:

(1) Select 3 defect-free cylinders from each of the first, middle, and last zones (nine cylinders total).[16] If the X and Y requests are to be on different cylinders, select 3 adjacent cylinders in each zone. Perform steps 2 through 8 for each set of 3 cylinders.

(2) Begin with a short (e.g., 1 sector) *inter-request distance*. Perform the following steps using slowly increasing inter-request distances until the 50th and 95th percentile values have been determined. To improve extraction efficiency, adjust the inter-request distance in 5-sector increments during the "coarse" tuning phase and 1-sector increments during the subsequent "fine" tuning phase.

(3) Pick 20 random targets for the X requests. If the X requests are to be READs, pick targets near the logical "beginning" of each available track (see section 5.2). Compute 20 appropriate Y request targets from the 20 X request targets, the current inter-request distance and any specified track switch or cylinder seek values for the Y requests.

(4) Perform 10 repetitions for each X,Y request pair (20 target pairs $\times$ 10 repetitions = 200 pairs total). To improve extraction efficiency, reduce the number of repetitions during the "coarse" tuning phase. If either of the requests are READs, intersperse the requests such that the chances of a hit in the on-board cache are minimized (or re-initialize the cache between each 20 pair repetition). The individual completion time differences for all 200 pairs should be measured.

(5) Count and discard completion time differences wherein the Y request suffered a rotational "miss". For our test disks, we record a rotational miss if the Y request service time exceeds 3/4 of a rotation. Compute the percentage of requests that were not discarded (i.e., those that did **not** suffer a rotational miss).

(6) Compute the mean time between the completion of the X requests and the start of the Y requests(the mean *host delay*) for the request pairs not discarded in step 5.

(7) As the inter-request distance increases, there will come a point where more than 50% of the Y requests do **not** suffer a rotational miss. The 50th percentile MTBRC for the current zone is taken as the mean of the non-discarded request completion differences at this inter-request distance.

(8) As the inter-request distance continues to increase, there will come a point where more than 95% of the Y requests do **not** suffer a rotational miss. The 95th percentile MTBRC for the current zone is taken as the mean of the non-discarded request completion differences at this inter-request distance.

(9) Combine the information gathered from all three zones by computing the mean of the 50th percentile values and maximum of the 95th percentile values. The inter-request distances (measured in fractions of a rotation) and mean host delays at which the 50% and 95% marks were crossed should also be noted.

Additional steps may be taken to detect and remove excessive noise. For example, if the three per-zone 50th percentile values computed in step 7 are not within 5–15% of each other, return to step 3 and repeat the process with a new set of physical request pairs. A filter can also be inserted between steps 4 and 5 that ignores request pairs with completion differences greater than two full rotations[17] or excessive host delay values. Also, the algorithm implementation should minimize any host delay between the requests in each pair (e.g., reduce the code executed by the host between request pairs).

Using the above algorithm, the 50th and 95th percentile MTBRC values for a specific pair of request types may be determined in a few minutes. By reducing the number of physical request pairs chosen in step 3 and/or the number of repetitions performed in step 4, the extraction cycle can easily be reduced.

[16]We chose nine cylinders as a reasonable trade-off between extraction accuracy and efficiency.

[17]We do not understand why a double (or even triple) rotational miss can occur, but the phenomenon occurred several times during the course of our study.