The DiskSim Simulation Environment Version 1.0 Reference Manual

Gregory R. Ganger, Bruce L. Worthington, and Yale N. Patt

CSE-TR-358-98

February 27, 1998

Computer Science and Engineering Division
Room 3402, EECS Building
Department of Electrical Engineering and Computer Science
The University of Michigan, Ann Arbor, Michigan 48109-2122

The DiskSim Simulation Environment
Version 1.0 Reference Manual

Gregory R. Ganger, Bruce L. Worthington, and Yale N. Patt

Department of EECS
University of Michigan, Ann Arbor
{ganger,worthing,patt}@eecs.umich.edu

February 27, 1998

Abstract

DiskSim is an efficient, accurate and highly-configurable disk system simulator developed at the University of Michigan to support research into various aspects of storage subsystem architecture. It includes modules that simulate disks, intermediate controllers, buses, device drivers, request schedulers, disk block caches, and disk array data organizations. In particular, the disk drive module simulates modern disk drives in great detail and has been carefully validated against several production disks (with accuracy that exceeds any previously reported simulator).

This manual describes how to configure and use DiskSim, which has been made publicly available with the hope of advancing the state-of-the-art in disk system performance evaluation in the research community. The manual also briefly describes DiskSim's internal structure and various validation results.

Acknowledgements

We thank John Fordemwalt, Richie Lary, Chris Ruemmler and John Wilkes for their insights on various disk drive characteristics. Our research into the I/O subsystem benefited greatly from the financial and technical support of several industrial partners, including NCR, Hewlett-Packard, Scientific and Engineering Software, and Digital Equipment Corporation.

TABLE OF CONTENTS

1 Introduction

Because of trends in both computer technology advancement (e.g., CPU speeds vs. disk access times) and application areas (e.g., on-demand video, global information access), storage system performance is becoming an increasingly large determinant of overall system performance. As a result, the need to understand storage performance under a variety of workloads is growing. Disks drives, which are still the secondary storage medium of choice, continue to expand in capacity and reliability while decreasing in unit cost, price/capacity, and power requirements. Performance characteristics also continue to change as the result of maturing technologies as well as new advances in materials, sensors, and electronics. New storage subsystem architectures may be needed to better exploit current and future generations of disk devices. The DiskSim simulation environment was developed as a tool for two purposes: understanding storage performance and evaluating new architectures.

1.1 What DiskSim does

DiskSim is an efficient, accurate, highly-configurable disk system simulator. It is written in C and requires no special system software. It includes modules for most secondary storage components of interest, including device drivers, buses, controllers, adapters and disk drives. Some of the major functions (e.g., request queueing/scheduling, disk block caching, disk array data organizations) that can be present in several different components (e.g., operating system software, intermediate controllers, disk drives) have been implemented as separate modules that are linked into components as desired. Some of the component modules are highly detailed (e.g., the disk module), and the individual components can be configured and interconnected in a variety of ways. DiskSim has been used in a variety of published studies (and several unpublished studies) to understand modern storage subsystem performance [Ganger93a, Worthington94], to understand how storage performance relates to overall system performance [Ganger93, Ganger95, Ganger95a], and to evaluate new storage subsystem architectures [Worthington95a].

DiskSim has been validated both as part of a more comprehensive system-level model [Ganger93, Ganger95a] and as a standalone subsystem [Worthington94, Worthington95]. In particular, the disk module (which is extremely detailed) has been carefully validated against five different disk drives from three different manufacturers. The accuracy demonstrated far exceeds that of any other disk simulator known to the authors (e.g., see [Ruemmler94]).

DiskSim can be driven by externally-provided I/O request traces or internally-generated synthetic workloads. Several trace formats have been used and new ones can be easily added. The synthetic trace generation module is quite flexible, particularly in the request arrival model (which can mimic an open process, a closed process or something in between). DiskSim was originally part of a larger, system-level model [Ganger93, Ganger95] that modeled each request's interactions with executing processes, but has been separated out for public dissemination.[1] As a result, it can be integrated into full system simulators (e.g., simulators like SimOS [Rosenblum95]) with little difficulty.

[1] The system-level model includes several portions of a proprietary OS, allowing it to achieve a close match to the real system behavior [Ganger95] but also preventing it from being publicly released.

1.2 What DiskSim does not do

DiskSim, by itself, simulates and reports on only the performance-related aspects of the storage subsystem. It does not model the behavior of the other computer system components or interactions between them and the storage subsystem.[2] Because storage subsystem performance metrics are not absolute indicators of overall system performance (e.g., see [Ganger95a]), promising architectures should be evaluated in the context of a more comprehensive system model (or a real system). In such cases, DiskSim becomes one component of the full model (just as a storage subsystem is one component of a full system).

DiskSim models the performance behavior of disk systems, but does not actually save or restore data for each request. If such functionality is desired (as, for example, when building a full system simulator like SimOS), it can easily be provided independently of DiskSim, which will still provide accurate timings for I/O activity.

1.3 Limitations of version 1.0

The first released version of DiskSim has two main known short-comings. First, specification of component interconnection in DiskSim is less general and more complicated than it should be. Also, DiskSim is not very helpful in debugging imperfect interconnection specifications. Second, several aspects of DiskSim's software engineering are suboptimal. In particular, the DiskSim code base suffers from inadequate type management, overloaded variables and parameters, and oldstyle function prototypes.

1.4 Organization of Manual

This manual describes how to configure and use DiskSim and how to interpret the output generated. Section 2 explains the DiskSim command line and how to execute the simulator given the appropriate configuration files. Section 3 describes the parameter file format, the various parameters and the additional configuration files required. Section 4 describes how to provide an input workload of I/O requests to DiskSim – options include external traces, internally-generated synthetic workloads, and interactions with a larger simulation environment. Section 5 describes the contents of the output file. Section 6 provides validation data (all of which has been published previously [Worthington94, Worthington95, Ganger95a]) showing that DiskSim accurately models the behavior of several high-performance disk drives produced in the early 1990's.

This manual does not provide details about DiskSim's internals. We refer those that wish to understand DiskSim more thoroughly and/or modify it to the appendices of [Ganger95].

[2] Actually, a rudimentary system model was kept in place to support the internal synthetic generation module. However, it should not be viewed as representative of any real system's behavior.

2 Running DiskSim

DiskSim requires five command line arguments and accepts zero or more parameter override descriptions:

disksim parfile outfile tracetype tracefile synthgen? [*par_override* ...]

where:

- *disksim* is the name of the executable.
- *parfile* is the name of the parameter file (whose format is described in chapter 3).
- *outfile* is the name of the output file (whose format is described in chapter 5). Output can be directed to stdout by specifying "stdout" for this argument.
- *tracetype* identifies the format of the trace input, if any (options are described in chapter 4).
- *tracefile* identifies the trace file to be used as input. Input is taken from stdin when "stdin" is specified for this argument.
- *synthgen?* determines whether or not the synthetic workload generation portion of the simulator should be enabled (any value other than "0" enables synthetic workload generation). The synthetic generator(s) are configured by values in the parameter file, as described in chapter 4. Currently, the simulator cannot use both an input trace and an internally-generated synthetic workload at the same time.
- *par_override* allows default parameter values or parameter values from *parfile* to be replaced by values specified in the command line. The exact syntax is described in the following section.

2.1 Parameter Overrides

When using DiskSim to examine the performance impacts of various storage subsystem design decisions (e.g., sensitivity analyses), the experimental configurations of interest are often quite similar. To avoid the need for numerous parameter files with incremental differences, DiskSim allows some parameter values to be overridden with command line arguments. The parameter overrides are applied after the parameter file has been read, but before the internal configuration phase begins. Each parameter override is described by the four command line arguments in the following sequence:

par_override = module_name instance_range par_name par_value

1. *module_name* is a string representing the module whose parameter is to be overridden. For example, "disk" represents the disk drive module and "iosim" represents the general I/O subsystem simulation setup.
2. *instance_range* identifies the range of module instances whose parameters are to be overridden, specified as "X–Y" (or "1" to specify that all instances should be affected). The range is inclusive and X may equal Y. The concept of module instances should be less ambiguous after the parameter file description (chapter 3) is understood.

3. *par_name* is a string identifying the parameter to be overridden. The string chosen for a parameter is generally consistent with its internal name, and the easiest way to discover the internal name is to search (e.g., grep) the source code for the exact text that precedes the parameter value in the parameter file. If the parameter is for a module that is included in a more comprehensive module, the larger module is given as *module_name* and the "submodule" is (by convention) identified in the parameter name. So, for example, a disk request scheduling algorithm (which is a parameter of the "ioqueue" module) is identified by the string "ioqueue_schedalg".

4. *par_value* is the overriding value for the parameter for the specified instances of the specified module.

The set of parameters whose values can be overridden from the command line is largely defined as the set of parameters incrementally varied in previous studies (as opposed to some objective decision process). Adding new parameters to this set is straightforward: simply modify the "*_param_override()" function of the appropriate module to handle the new override. The error checks for the parameter should be taken from the corresponding "*_readparams()" code (which, like the internal variable name, can be found by searching the source code for specific text from the parameter file) to maintain consistency.

2.2 Example command line

An example may be useful to demonstrate the command line syntax. The below text:

disksim parms.1B stdout ascii t.Jan6 0 disk 1–16 segsize 64 disk 1 ioqueue_schedalg 4

executes DiskSim as follows:

- initial parameters are read from file *parms.1B*;
- output (e.g., statistics) is sent to *stdout*;
- the *ascii* input trace is read from file *t.Jan6*;
- there is no synthetically generated activity;
- the cache segment size parameter values of disks 1 through 16, as specified in the parameter file (*parms.1B*), are overridden with a value of 64 (sectors); and
- the scheduling algorithm parameter value for all disks is overridden with a value of 4 (which corresponds to a Shortest-Seek-Time-First algorithm).

3 The parameter file

DiskSim can be configured via the parameter file to model a wide variety of storage subsystems. To provide this flexibility, the parameter file (and auxiliary configuration files) is very long, requiring several hundred lines in most cases. Although it can sometimes be unwieldy, the configuration file is organized so as to simplify reconfiguration.

The parameter file consists of several sections. Each section (more precisely, each line) must appear a specific order. No reordering will be understood by DiskSim. The parameter file begins with some general I/O subsystem parameters, including directives for which results should be included in the output. This is followed by two sections that specify the components that will be part of the simulated subsystem and their interconnections, respectively. The separation of component definitions and their interconnections greatly reduces the effort required to develop and integrate new components as well as the effort required to understand and modify the existing components [Satya86]. This is followed by brief sections that specify which disk drives (if any) should be rotationally synchronized with each other and which disk drives should be part of particular disk array data organizations. The final section of the parameter file, which is only needed when the internal synthetic workload generator is used, specifies the rough characteristics of both the overall system and the synthetic generator.

Several example parameter files are provided with the software distribution. The remainder of this section describes the parameters in the order they must appear in a parameter file. When only specific values are allowed, they are shown in brackets.

3.1 General Simulation Parameters

Byte Order (Endian): String ["Big" or "Little"] specifying the endian-ness of the machine on which the simulation will be run. This is only relevant when the simulation run uses raw (binary) input traces (which must have an associated endian-ness) and DiskSim can not automatically determine the endian-ness of the host machine.

Init Seed: Integer specifying the initial seed for the random number generator. The string, "PID", can be provided (instead of an integer) to specify that the process ID of the simulation run should be used as the seed. The initial seed value is applied at the very beginning of the simulation and is used during the initialization phase (e.g., for determining initial rotational positions). Explicitly specifying the random generator seed enables experiment repeatability.

Real Seed: Integer specifying the simulation seed for the random number generator. The string, "PID", can be provided (instead of an integer) to specify that the process ID of the simulation run should be used as the seed. The 'real' seed value is applied after the initialization phase and is used during the simulation phase (e.g., for synthetic workload generation). This allows multiple synthetic workloads (with different simulation seeds) to be run on equivalent configurations (i.e., with identical initial seeds, as specified above).

Statistic warm-up period: String ["FF.FF seconds" or "III I/Os"] specifying the warm-up period, after which the statistics will be reset to their initial values. It can be given as a Float [nonnegative (FF.FF)] followed by the "seconds" qualifier, indicating a fixed amount of simulated time (in seconds), or as an Integer [nonnegative (III)] followed by the "I/Os" qualifier, indicating a fixed number of disk requests.

Stat (dist) definition file: String specifying the name of the input file containing the specifications for the statistical distributions to collect. This file allows the user to control the number and sizes of histogram bins into which data are collected. This file is mandatory, and section 5.1 describes its use.

Output file for trace of I/O requests simulated: String specifying the name of the output file to contain a trace of disk request arrivals (in the default ASCII trace format described in section 4.1). This allows instances of synthetically generated workloads to be saved and analyzed after the simulation completes. This is particularly useful for analyzing (potentially pathological) workloads produced by a system-level model. A value of "0" or of "null" disables this feature.

PRINTED I/O SUBSYSTEM STATISTICS: This section contains a series of Boolean [1 or 0] parameters that specify whether or not particular groups of statistics are reported. The *driver* parameters control statistics output from both the device drivers (individual values and overall totals) and any driver-level disk array logical data organizations (referred to as **logorgs**). The *controller* values are irrelevant because there are no controller-level statistics in the released version of DiskSim. The *disk* parameters control statistics output for the disks themselves (individually, overall, and combined with the other disks in a particular logorg). The different print-control parameters (corresponding to particular statistics groups) will be identified with individual statistics in section 5.

I/O Trace Time Scale: Float [positive] specifying a value by which each arrival time in a trace is multiplied. For example, a value of 2.0 doubles each arrival time, lightening the workload by stretching it out over twice the length of time. Conversely, a value of 0.5 makes the workload twice as heavy by compressing inter-arrival times. This value has no effect on workloads generated internally (by the synthetic generator).

Number of I/O Mappings: Integer [nonnegative] specifying the number of multi-purpose mappings provided in subsequent lines of the file that enable translation of disk request sizes and locations in an input trace into disk request sizes and locations appropriate for the simulated environment. When the simulated environment closely matches the traced environment, these mappings may be used simply to reassign disk device numbers. However, if the configured environment differs significantly from the trace environment, or if the traced workload needs to be scaled (by request size or range of locations), these mappings can be used to alter the the traced "logical space" and/or scale request sizes and locations. One mapping is allowed per traced device.

Mapping: A mapping from a device as identified in the trace to the storage subsystem device(s) being modeled. The total number of mappings must match the value of the parameter described above. Each mapping consists of five values:

- Hex Integer [nonnegative] specifying the traced device affected by this mapping.
- Hex Integer [nonnegative] specifying the simulated device such requests should access.
- Integer [positive] specifying a value by which a traced disk request location is multiplied to generate the starting location (in bytes) of the simulated disk request. For example, if the input trace specifies locations in terms of 512-byte sectors, a value of 512 would result in an equivalent logical space of requests.
- Integer [positive] specifying a value by which a traced disk request size is multiplied to generate the size (in bytes) of the simulated disk request.
- Integer [bytes] specifying a value to be added to each simulated request's starting location. This is especially useful for combining multiple trace devices' logical space into the space of a single simulated device.

3.2 I/O Subsystem Component Specifications

DiskSim recognizes four component types: device driver, bus, controller and disk. For configuration purposes, everything falls into one of these categories. In addition, there are two subcomponents (queues/schedulers and caches) that can be part of multiple components and are therefore described separately. For each component, the parameter file identifies how many exist in the modeled subsystem and then has one or more component specifications (the instances of a component need not have identical characteristics). Each component specification can define one or more instances of the component. For the interconnection specifications, the components of a given type are numbered 1 thru N (in order of specification). Internally, they are numbered 0 thru N-1 (in the same order).

3.2.1 Device Drivers

There must be exactly one configured device driver per storage subsystem configuration for the released version of DiskSim.

Device driver type: Integer [1] included for extensibility purposes.

Constant access time: Overloaded Float [-3.0, -2.0, -1.0, 0.0, or positive in milliseconds] specifying any of several forms of storage simulation abstraction. A positive value indicates a fixed access time (after any queueing delays) for each disk request. With this option, requests do not propagate to lower levels of the storage subsystem (and the stats and configuration of lower levels are therefore meaningless). -1.0 indicates that the trace provides a measured access time for each request, which should be used instead of any simulated access times. -2.0 indicates that the trace provides a measured queue time for each request, which should be used instead of any simulated queue times. (Note: this can cause problems if multiple requests end up outstanding simultaneously to disks that don't support queueing.) -3.0 indicates that the trace provides measured values for both the access time and the queue time. Finally, 0.0 indicates that the simulation should compute all access and queue times as appropriate given the changing state of the storage subsystem.

The next twelve parameters in the parameter file configure the queue/scheduler subcomponent of the device driver specification and are described in section 3.2.5.

Use queueing in subsystem: Boolean [1 or 0] specifying whether or not the device driver allows more than one request to be outstanding (in the storage subsystem) at any point in time. During initialization, this parameter is combined with the parameterized capabilities of the subsystem itself to determine whether or not queueing in the subsystem is appropriate.

3.2.2 Buses

Bus Type: Integer [1 or 2] specifying the type of bus. 1 indicates an exclusively-owned (tenured) bus (i.e., once ownership is acquired, the owner gets 100% of the bandwidth available until ownership is relinquished voluntarily). 2 indicates a shared bus where multiple bulk transfers are interleaved (i.e., each gets a fraction of the total bandwidth).

Arbitration type: Integer [1 or 2] specifying the type of arbitration used for exclusively-owned buses (see above parameter description). 1 indicates slot-based priority (e.g., SCSI buses), wherein the order of attachment determines priority (i.e., the first device attached has the highest priority). 2 indicates First-Come-First-Served (FCFS) arbitration, wherein bus requests are satisfied in arrival order.

Arbitration time: Float [nonnegative milliseconds] specifying the time required to make an arbitration decision.

Read block transfer time: Float [nonnegative milliseconds] specifying the time required to transfer a single 512-byte block in the direction of the device driver / host.

Write block transfer time: Float [nonnegative milliseconds] specifying the time required to transfer a single 512-byte block in the direction of the disk drives.

Print stats for bus: Boolean [1 or 0] specifying whether or not to report the collected statistics.

3.2.3 Controllers

Controller Type: Integer [1, 2, or 3] specifying the type of controller. 1 indicates a simple controller that acts as nothing more than a bridge between two buses, passing everything straight through to the other side. 2 indicates a very simple, driver-managed controller based roughly on the NCR 53C700. 3 indicates a more complex controller that decouples lower-level storage component peculiarities from higher-level components (e.g., device drivers). The complex controller queues and schedules its outstanding requests and possibly contains a cache. As indicated below, it requires several parameters in addition to those needed by the simpler controllers.

Scale for delays: Float [nonnegative] specifying a multiplicative scaling factor for the various processing delays incurred by the controller. Default overheads for the 53C700-based controller and the more complex controller are hard-coded into the "read_specs" procedure of the controller module (and are easily changed). For the simple pass-thru controller, the scale factor represents the per-message propagation delay (because the hard-coded value is 1.0). 0.0 results in no controller overheads or delays. When the overheads/delays of the controller(s) cannot be separated from those of the disk(s), as is usually the case for single-point tracing of complete systems, the various disk overhead/delay parameter values should be populated and this parameter should be set to 0.0.

Bulk sector transfer time: Float [nonnegative milliseconds] specifying the time necessary to transfer a single 512-byte block to, from or through the controller. Transferring one block over the bus takes the maximum of this time, the block transfer time specified for the bus itself, and the block transfer time specified for the component on the other end of the bus transfer.

Maximum queue length: Integer [nonnegative] specifying the maximum number of requests that can be outstanding to the controller concurrently. The device driver discovers this value during initialization and respects it during operation. For the simple types of controllers (see above parameter description), 0 is assumed.

Print stats for controller: Boolean [1 or 0] specifying whether or not to report statistics for the controller. It is meaningless for the simple types of controllers (see above parameter description), as no statistics are collected for such controllers in the released version of DiskSim.

The remaining controller parameters are only included in specifications of the more complex controller type (3).

The next twelve parameters in the parameter file configure the queue/scheduler subcomponent of the device driver specification and are described in section 3.2.5.

The next nineteen parameters in the parameter file configure the cache subcomponent of the controller specification (for the complex controller type) and are described in section 3.2.6.

Max per-disk pending count: Integer [positive] specifying the maximum number of requests that the controller can have outstanding to each attached disk (i.e., the maximum number of requests that can be dispatched to a single disk). This parameter only affects the interaction of the controller with its attachments; it is not visible to the device driver.

3.2.4 Disks

The parameters for a disk specification can either be included in the parameter file directly or pulled from a separate file. In the latter case, the main parameter file includes just two lines, providing a unique identifier (e.g., product name) for the disk specification and a file name in which to find it. The actual specification is then found by searching the given file for a duplicate of the unique identifier. In either case, the parameters for a disk specification are:

Access time (in msecs): Overloaded Float [-2.0, -1.0, nonnegative milliseconds] specifying the method for computing mechanical delays. A nonnegative value indicates a fixed per-request access time (i.e., actual mechanical activity is not simulated). -1.0 indicates that seek activity should be simulated but rotational latency is assumed to be equal to one half of a rotation, which is the statistical mean for random disk access. -2.0 indicates that both seek and rotational activity should be simulated.

Seek time (in msecs): Overloaded Float [-5.0, -4.0, -3.0, -2.0, -1.0, nonnegative milliseconds] specifying the method for computing seek delays. A nonnegative value indicates a fixed per-request seek time. -1.0 indicates that the single-cylinder seek time, the average seek time and the full-strobe seek time parameters should be used to compute the simulated seek time via linear interpolation. -2.0 indicates that the same three parameters should be used with the seek equation described in [Lee93]:

$$seekTime(x) = \begin{cases} 0 & : \quad if x = 0 \\ a\sqrt{x-1} + b(x-1) + c & : \quad if x > 0 \end{cases}, \text{where}$$

x is the seek distance in cylinders,
$a = (-10minSeek + 15avgSeek - 5maxSeek)/(3\sqrt{numCyl})$,
$b = (7minSeek - 15avgSeek + 8maxSeek)/(3numCyl)$, and
$c = minSeek$.

-3.0 indicates that the six-value "HPL" parameter (see below) should be used with the seek equation described in [Ruemmler94]. -4.0 indicates that the six-value "HPL" parameter (see below) should be used with the seek equation described in [Ruemmler94] for all seeks greater than 10 cylinders in length. For smaller seeks, use the 10-value "First seeks" parameter (see below) as in [Worthington94]. -5.0 indicates that a more complete seek curve (provided in a separate file) should be used, with linear interpolation used to compute the seek time for unspecified distances.

Single cylinder seek time: Float [nonnegative milliseconds] specifying the time necessary to seek to an adjacent cylinder.

Average seek time: Overloaded Float [nonnegative milliseconds] or String specifying either the mean time necessary to perform a random seek or, if a complete seek curve is to be provided (see above), the name of the input file containing the seek curve data. The format of such a file is very simple, consisting of a single line containing the number of seek distances for which corresponding seek times are given (Integer [positive]), followed by a line for each seek distance containing two values: the distance (Integer [positive cylinders]) and the seek time (Float [nonnegative milliseconds]). Examples are provided with the released software.

Full strobe seek time: Float [nonnegative milliseconds] specifying the full-strobe seek time (i.e., the time to seek from the innermost cylinder to the outermost cylinder).

Add. write settling delay: Float [nonnegative milliseconds] specifying the additional time required to precisely settle the read/write head for writing (after a seek or head switch). As this parameter implies, the seek times computed using the above parameter values are for read access.

HPL seek equation values: Six Integers specifying the variables $V1$ through $V6$ of the seek equation described in [Ruemmler94]:

Seek distance	Seek time
1 cylinder	$V6$
<$V1$ cylinders	$V2 + V3 * \sqrt{dist}$
>=$V1$ cylinders	$V4 + V5$ * dist

, where *dist* is the seek distance in cylinders.

If $V6 == -1$, single-cylinder seeks are computed using the second equation. $V1$ is specified in cylinders, and $V2$ through $V6$ are specified in milliseconds.

First 10 seek times: Ten Integers [nonnegative milliseconds] specifying the seek time for seek distances of 1 through 10 cylinders.

Head switch time: Float [nonnegative milliseconds] specifying the time required for a head switch (i.e., activating a different read/write head in order to access a different media surface).

Rotation speed (in rpms): Integer [positive rotations per minute] specifying the rotation speed of the disk platters.

Percent error in rpms: Float [nonnegative percentage] specifying the maximum deviation in the rotation speed specified above. During initialization, the simulated rotation speed for each disk is randomly chosen from a uniform distribution of the specified rotation speed $\pm$ the maximum allowed error.

Number of data surfaces: Integer [positive] specifying the number of magnetic media surfaces (not platters!) on which data are recorded. Dedicated servo surfaces should not be counted for this parameter.

Number of cylinders: Integer [positive] specifying the number of physical cylinders. All cylinders that impact the logical to physical mappings should be included.

Blocks per disk: Integer [positive] specifying the number of data blocks. This capacity is exported by the disk (e.g., to a disk array controller). It is not used directly during simulation, but is compared to a similar value computed from other disk parameters. A warning is reported if the values differ.

Sparing scheme used: Integer [0, 1, 2, or 3] specifying the type of sparing used by the disk. Later parameters determine where spare space is allocated. 0 indicates that no spare sectors are allocated. 1 indicates that entire tracks of spare sectors are allocated at the "end" of some or all zones (sets of cylinders). 2 indicates that spare sectors are allocated at the "end" of each cylinder. 3 indicates that spare sectors are allocated at the "end" of each track.

Per-request overhead time: Float [nonnegative milliseconds] specifying a per-request processing overhead that takes place immediately after the arrival of a new request at the disk. It is additive with various other processing overheads described below, but in general either the other overheads are set to zero or this parameter is set to zero.

Time scale for overheads: Float [nonnegative] specifying a multiplicative scaling factor for all processing overhead times. For example, 0.0 eliminates all such delays, 1.0 uses them at face value, and 1.5 increases them all by 50%.

Bulk sector transfer time: Float [nonnegative] specifying the time for the disk to transfer a single 512-byte block over the bus. Recall that this value is compared with the corresponding bus and controller block transfer values to determine the actual transfer time (i.e., the maximum of the three values).

Hold bus entire read xfer: Boolean [1 or 0] specifying whether or not the disk retains ownership of the bus throughout the entire transfer of "read" data from the disk. If false (0), the disk may release the bus if and when the current transfer has exhausted all of the available data in the on-board buffer/cache and must wait for additional data sectors to be read off the disk into the buffer/cache.

Hold bus entire write xfer: Boolean [1 or 0] specifying whether or not the disk retains ownership of the bus throughout the entire transfer of "write" data to the disk. If false (0), the disk may release the bus if and when the current transfer has filled up the available space in the on-board buffer/cache and must wait for data from the buffer/cache to be written to the disk.

Allow sneaky full read hits: Boolean [1 or 0] specifying whether or not a new read request whose data are entirely contained in a single segment of the disk cache is allowed to immediately transfer that data over the bus while another request is moving the disk actuator and/or transferring data between the disk cache and the disk media. In essence, the new read request "sneaks" its data out from the disk cache without interrupting the current (active) disk request.

Allow sneaky partial read hits: Boolean [1 or 0] specifying whether or not a new read request whose data are partially contained in a single segment of the disk cache is allowed to immediately transfer that data over the bus while another request is moving the disk actuator and/or transferring data between the disk cache and the disk media. In essence, the new read request "sneaks" the cached portion of its data out from the disk cache without interrupting the current (active) disk request.

Allow sneaky intermediate read hits: Boolean [1 or 0] specifying whether or not the on-board queue of requests is searched during idle bus periods in order to find read requests that may be partially or completely serviced from the current contents of the disk cache. That is, if the current (active) request does not need bus access at the current time, and the bus is available for use, a queued read request whose data are in the cache may obtain access to the bus and begin data transfer. "Full" intermediate read hits are given precedence over "partial" intermediate read hits.

Allow read hits on write data: Boolean [1 or 0] specifying whether or not data placed in the disk cache by write requests are considered usable by read requests. If false (0), such data are removed from the cache as soon as they have been copied to the media.

Allow write prebuffering: Boolean [1 or 0] specifying whether or not the on-board queue of requests is searched during idle bus periods for write requests that could have part or all of their data transferred to the on-board cache (without disturbing an ongoing request). That is, if the current (active) request does not need bus access at the current time, and the bus is available for use, a queued write request may obtain access to the bus and begin data transfer into an appropriate cache segment. Writes that are contiguous to the end of the current (active) request are given highest priority in order to facilitate continuous transfer to the media, followed by writes that have already "prebuffered" some portion of their data.

Preseeking level: Integer [0, 1, or 2] specifying how soon the actuator is allowed to start seeking towards the media location of the next request's data. 0 indicates no preseeking, meaning that the actuator does not begin relocation until the current request's completion has been confirmed by the controller (via a completion "handshake" over the bus). 1 indicates that the actuator can begin relocation as soon as the completion message has been prepared for transmission by the disk. 2 indicates that the actuator can begin relocation as soon as the access of the last sector of the current request (and any required prefetching) has been completed. This allows greater parallelism between bus activity and mechanical activity.

Never disconnect: Boolean [1 or 0] specifying whether or not the disk retains ownership of the bus during the entire processing and servicing of a request (i.e., from arrival to completion). If false (0), the disk may release the bus whenever it is not needed for transferring data or control information.

Print stats for disk: Boolean [1 or 0] specifying whether or not to report statistics for the disk.

Avg sectors per cylinder: Integer [nonnegative] specifying (an approximation of) the mean number of sectors per cylinder. This value is exported to any external schedulers[3] requesting it and is not used by the disk itself.

Max queue length at disk: Integer [positive] specifying the maximum number of requests that the disk can have in service or queued for service at any point in time. During initialization, other components request this information and respect it during simulation.

The next twelve parameters in the parameter file configure the queue/scheduler subcomponent of the disk specification and are described in section 3.2.5.

Number of buffer segments: Integer [positive] specifying the number of segments in the on-board buffer/cache. A buffer segment is similar to a cache line, in that each segment contains data that is disjoint from all other segments. However, segments tend to be organized as circular queues of logically sequential disk sectors, with new sectors pushed into an appropriate queue either from the bus (during a write) or from the disk media (during a read). As data are read from the buffer/cache and either transferred over the bus (during a read) or written to the disk media (during a write), they are eligible to be pushed out of the segment (if necessary or according to the dictates of the buffer/cache management algorithm).

Maximum number of write segments: Integer [positive] specifying the number of cache segments available for holding "write" data at any point in time. Because write-back caching is typically quite limited in current disk cache management schemes, some caches only allow a subset of the segments to be used to hold data for write requests (in order to minimize any interference with sequential read streams).

Segment size (in blks): Integer [positive sectors] specifying the size of each buffer segment, assuming a static segment size. Some modern disks will dynamically resize their buffer segments (and thereby alter the number of segments) to respond to perceived patterns of workload behavior, but DiskSim does not currently support this functionality.

---

[3]Some scheduling algorithms available in DiskSim utilize approximations of the actual layout of data on the disk(s) when reordering disk requests.

Use separate write segment: Boolean [1 or 0] specifying whether or not a single segment is statically designated for use by all write requests. This further minimizes the impact of write requests on the caching/prefetching of sequential read streams.

Low (write) water mark: Float [0.0–1.0] specifying the fraction of segment size or request size (see below) corresponding to the *low water mark*. When data for a write request are being transferred over the bus into the buffer/cache, and the buffer/cache segment fills up with "dirty" data, the disk may disconnect from the bus while the buffered data are written to the disk media. When the amount of dirty data in the cache falls below the low water mark, the disk attempts to reconnect to the bus to continue the interrupted data transfer.

High (read) water mark: Float [0.0–1.0] specifying the fraction of segment size or request size (see below) corresponding to the *high water mark*. When data for a read request are being transferred over the bus from the buffer/cache, and the buffer/cache segment runs out of data to transfer, the disk may disconnect from the bus until additional data are read from the disk media. When the amount of available data in the cache reaches the high water mark, the disk attempts to reconnect to the bus to continue the interrupted data transfer.

Set watermark by reqsize: Boolean [1 or 0] specifying whether the watermarks are computed as fractions of the individual request size or as fractions of the buffer/cache segment size.

Calc sector by sector: Boolean [1 or 0] specifying whether or not media transfers should be computed sector by sector rather than in groups of sectors. This optimization has no effect on simulation accuracy, but potentially results in shorter simulation times (at a cost of increased code complexity). It has not been re-enabled since the most recent modifications to DiskSim, so the simulator currently functions as if the value were always true (1).

Enable caching in buffer: Boolean [1 or 0] specifying whether or not on-board buffer segments are used for data caching as well as for speed-matching between the bus and the disk media. Most (if not all) modern disk drives utilize their buffers as caches.

Buffer continuous read: Integer [0–4] specifying the type of prefetching performed by the disk. 0 disables prefetching. 1 enables prefetching up to the end of the track containing the last sector of the read request. 2 enables prefetching up to the end of the cylinder containing the last sector of the read request. 3 enables prefetching up to the point that the current cache segment is full. 4 enables prefetching up to the end of the track following the track containing the last sector of the read request, provided that the current request was preceded in the not-too-distant past by another read request that accessed the immediately previous track. In essence, the last scheme enables a type of prefetching that tries to stay one logical track "ahead" of any sequential read streams that are detected.

Minimum read-ahead (blks): Integer [nonnegative] specifying the minimum number of disk sectors that must be prefetched after a read request before servicing another (read or write) request. A positive value may be beneficial for workloads containing multiple interleaved sequential read streams, but 0 is typically the appropriate value.

Maximum read-ahead (blks): Integer [nonnegative] specifying the maximum number of disk sectors that may be prefetched after a read request (regardless of all other prefetch parameters).

Read-ahead over requested: Boolean [1 or 0] specifying whether or not newly prefetched data can replace (in a buffer segment) data returned to the host as part of the most recent read request.

Read-ahead on idle hit: Boolean [1 or 0] specifying whether or not prefetching should be initiated by the disk when a read request is completely satisfied by cached data (i.e., a "full read hit").

Read any free blocks: Boolean [1 or 0] specifying whether or not disk sectors logically prior to the requested sectors should be read into the cache if they pass under the read/write head prior to reaching the requested data (e.g., during rotational latency).

Fast write level: Integer [0, 1, or 2] specifying the type of write-back caching implemented. 0 indicates that write-back caching is disabled (i.e., all dirty data must be written to the disk media prior to sending a completion message). 1 indicates that write-back caching is enabled for contiguous sequential write request streams. That is, as long as each request arriving at the disk is a write request that "appends" to the current segment of dirty data, a completion message will be returned for each new request as soon as all of its data have been transferred over the bus to the disk buffer/cache. 2 indicates that write-back caching is enabled for contiguous sequential write request streams even if they are intermixed with read or non-appending write requests, although before any such request is serviced by the disk, all of the dirty write data must be flushed to the media. A scheduling algorithm that gives precedence to sequential writes would maximize the effectiveness of this option.

Immediate buffer read: Boolean [1 or 0] specifying whether or not disk sectors should be transferred into the on-board buffer in the order that they pass under the read/write head rather than in strictly ascending logical block order. This is known as *zero-latency reads* or *read-on-arrival*. It is intended to improve response times by reducing rotational latency (by not rotating all the way around to the first requested sector before beginning to fill the buffer/cache).

Immediate buffer write: Boolean [1 or 0] specifying whether or not disk sectors should be transferred from the on-board buffer in the order that they pass under the read/write head rather than in strictly ascending logical block order. This is known as *zero-latency writes* or *write-on-arrival*. It is intended to improve response times by reducing rotational latency (by not rotating all the way around to the first "dirty" sector before beginning to flush the buffer/cache).

Combine seq writes: Boolean [1 or 0] specifying whether or not sequential data from separate write requests can share a common cache segment. If true (1), data are typically appended at the end of a previous request's dirty data. However, if all of the data in a cache segment are dirty, and no mechanical activity has begun on behalf of the request(s) using that segment, "prepending" of additional dirty data are allowed provided that the resulting cache segment contains a single contiguous set of dirty sectors.

Stop prefetch in sector: Boolean [1 or 0] specifying whether or not a prefetch may be aborted in the "middle" of reading a sector off the media. If false (0), prefetch activity is only aborted at sector boundaries.

Disconnect write if seek: Boolean [1 or 0] specifying whether or not the disk should disconnect from the bus if the actuator is still in motion (seeking) when the last of a write request's data has been transferred to the disk buffer/cache.

Read directly to buffer: Boolean [1 or 0] specifying whether or not space for a sector must be available in the buffer/cache prior to the start of the sector read. If false (0), a separate sector buffer is assumed to be available for use by the media-reading electronics, implying that the data for a sector is transferred to the main buffer/cache only after it has been completely read (and any error-correction algorithms completed).

Immed transfer partial hit: Boolean [1 or 0] specifying whether or not a read request whose initial (but not all) data are present in the disk buffer/cache has that data immediately transferred over the bus. If false (0), the data are immediately transferred only if the amount of requested data that are present in the buffer/cache exceed the *high water mark* (see above).

The following six parameters specify per-request command processing overheads that are applied after the request arrives at the disk and before any corresponding bus or read/write head activity is initiated. Values for these parameters can be determined empirically (see [Worthington95]) or via some form of documentation (e.g., technical manual or disk descriptor file). An additional fifteen parameters are included for use in high-precision disk simulation. Obtaining appropriate values for these parameters may require access to additional (e.g., confidential) documentation or detailed analysis of a given model of disk drive in a tightly-controlled environment.

Read hit over. after read: Float [nonnegative milliseconds] specifying the processing time for a read request that hits in the on-board cache when the immediately previous request was also a read. This delay is applied before any data are transferred from the disk buffer/cache.

Read hit over. after write: Float [nonnegative milliseconds] specifying the processing time for a read request that hits in the on-board cache when the immediately previous request was a write. This delay is applied before any data are transferred from the disk buffer/cache.

Read miss over. after read: Float [nonnegative milliseconds] specifying the processing time for a read request that misses in the on-board cache when the immediately previous request was also a read. This delay is applied before any mechanical positioning delays or data transfer from the media.

Read miss over. after write: Float [nonnegative milliseconds] specifying the processing time for a read request that misses in the on-board cache when the immediately previous request was a write. This delay is applied before any mechanical positioning delays or data transfer from the media.

Write over. after read: Float [nonnegative milliseconds] specifying the processing time for a write request when the immediately previous request was a read. This delay is applied before any mechanical positioning delays and before any data are transferred to the disk buffer/cache.

Write over. after write: Float [nonnegative milliseconds] specifying the processing time for a write request when the immediately previous request was also a write. This delay is applied before any mechanical positioning delays and before any data are transferred to the disk buffer/cache.

Read completion overhead: Float [nonnegative milliseconds] specifying the processing time for completing a read request . This overhead is applied just before the completion message is sent over the (previously acquired) bus and occurs in parallel with any background disk activity (e.g., prefetching or preseeking).

Write completion overhead: Float [nonnegative milliseconds] specifying the processing time for completing a write request . This overhead is applied just before the completion message is sent over the (previously acquired) bus and occurs in parallel with any background disk activity (e.g., preseeking).

Data preparation overhead: Float [nonnegative milliseconds] specifying the additional processing time necessary when preparing to transfer data over the bus (for either reads or writes). This command processing overhead is applied after obtaining access to the bus (prior to transferring any data) and occurs in parallel with any ongoing media access.

First reselect overhead: Float [nonnegative milliseconds] specifying the processing time for reconnecting to (or "reselecting") the controller for the first time during the current request.[4] This command processing overhead is

[4]Reselection implies that the disk has explicitly disconnected from the bus at some previous point while servicing the current request

applied after the disk determines that reselection is appropriate (prior to attempting to acquire the bus) and occurs in parallel with any ongoing media access.

Other reselect overhead: Float [nonnegative milliseconds] specifying the processing time for reconnecting to the controller after the first time during the current request (i.e., the second reselection, the third reselection, etc.). This command processing overhead is applied after the disk determines that reselection is appropriate (prior to attempting to acquire the bus) and occurs in parallel with any ongoing media access.

Read disconnect afterread: Float [nonnegative milliseconds] specifying the processing time for a read request that disconnects from the bus when the previous request was also a read. This command processing overhead is applied after the disk determines that disconnection is appropriate (prior to requesting disconnection from the bus) and occurs in parallel with any ongoing media access.

Read disconnect afterwrite: Float [nonnegative milliseconds] specifying the processing time for a read request that disconnects from the bus when the previous request was a write request. This command processing overhead is applied after the disk determines that disconnection is appropriate (prior to requesting disconnection from the bus) and occurs in parallel with any ongoing media access.

Write disconnect overhead: Float [nonnegative milliseconds] specifying the processing time for a write request that disconnects from the bus (which generally occurs after the data are transferred from the host to the on-board buffer/cache). This command processing overhead is applied after the disk determines that disconnection is appropriate (prior to requesting disconnection from the bus) and occurs in parallel with any ongoing media access.

Extra write disconnect: Boolean [1 or 0] specifying whether or not the disk disconnects from the bus after processing the write command but before any data have been transferred over the bus into the disk buffer/cache. Although there are no performance or reliability advantages to this behavior, it has been observed in at least one production SCSI disk and has therefore been included in DiskSim. If true (1), the next five parameters configure additional overhead values specifically related to this behavior.

Extradisc command overhead: Float [nonnegative milliseconds] specifying the processing time for a write request that disconnects from the bus before transferring any data to the disk buffer/cache. This overhead is applied before requesting disconnection from the bus and before any mechanical positioning delays. This parameter (when enabled) functions in place of the above "Write over." parameters.

Extradisc disconnect overhead: Float [nonnegative milliseconds] specifying the additional processing time for a write request that disconnects from the bus before transferring any data to the disk buffer/cache. This overhead is also applied before requesting disconnection from the bus, but it occurs in parallel with any mechanical positioning delays. This parameter (when enabled) functions in place of the above "Write disconnect" parameter for initial write disconnections.

Extradisc inter-disconnect delay: Float [nonnegative milliseconds] specifying the time between the initial disconnect from the bus and the subsequent reconnection attempt for a write request that disconnects from the bus before transferring any data to the disk buffer/cache. It occurs in parallel with any mechanical positioning delays.

Extradisc 2nd disconnect overhead: Float [nonnegative milliseconds] specifying the processing time for a

---

and is now attempting to reestablish communication with the controller. Disconnection and subsequent reselection result in some additional command processing and protocol overhead, but they may also improve the overall utilization of bus resources shared by multiple disks (or other peripherals).

write request that disconnects from the bus after data has been transferred but previously had disconnected without transferring any data to the disk buffer/cache. This command processing overhead is applied after the disk determines that disconnection is appropriate (prior to requesting disconnection from the bus) and occurs in parallel with any ongoing media access. This parameter (when enabled) functions in place of the above "Write disconnect" parameter for non-initial write disconnections.

Extradisc seek delta: Float [nonnegative milliseconds] specifying the additional delay between the completion of the initial command processing overhead and the initiation of any mechanical positioning for a write request that disconnects from the bus before transferring any data to the disk buffer/cache. This delay occurs in parallel with ongoing bus activity and related processing overheads.

Minimum seek delay: Float [nonnegative milliseconds] specifying the minimum media access delay for a nonsequential write request. That is, a nonsequential write request (after any command processing overheads) must wait at least this amount of time before accessing the disk media.

Number of bands: Integer [positive] specifying the number of *zones* (or *bands*) into which the set of disk cylinders is partitioned. Each zone (identified by an increasing positive integer) is described by the following nine parameters. A header line of the format "Band #N" precedes the parameters for each zone, where "N" is the zone identifier.

First cylinder number: Integer [nonnegative] specifying the first physical cylinder in the zone.

Last cylinder number: Integer [nonnegative] specifying the last physical cylinder in the zone.

Blocks per track: Integer [positive] specifying the number of sectors (independent of logical-to-physical mappings) on each physical track in the zone.

Offset of first block: Integer [nonnegative sectors] specifying the physical offset of the first logical sector in the zone. Physical sector 0 of every track is assumed to begin at the same angle of rotation.

Empty space at zone front: Integer [nonnegative sectors] specifying the size of the "management area" allocated at the beginning of the zone for internal data structures. This area can not be accessed during normal activity and is not part of the disk's logical-to-physical mapping.

Skew for track switch: Integer [nonnegative] specifying the number of physical sectors that are skipped when assigning logical block numbers to physical sectors at a track crossing point. Track skew is computed by the manufacturer to optimize sequential access.

Skew for cylinder switch: Integer [nonnegative] specifying the number of physical sectors that are skipped when assigning logical block numbers to physical sectors at a cylinder crossing point. Cylinder skew is computed by the manufacturer to optimize sequential access.

Number of spares: Integer [nonnegative] specifying the number of spare storage locations (sectors or tracks, depending on the sparing scheme chosen) allocated per region of coverage (track, cylinder, or zone, depending on the sparing scheme chosen). For example, if the sparing scheme is 1, indicating that spare tracks are allocated at the end of the zone, the value of this parameter indicates how many spare tracks have been allocated for this zone.

Number of slips: Integer [nonnegative] specifying the number of previously detected defective media locations (sectors or tracks, depending upon the sparing scheme chosen) that were skipped-over (or "slipped") when the logical-to-physical mapping was last created. Subsequent lines in the parameter file indicate exactly which media locations were slipped. Each such line has the form "**Slip X**" where X indicates the slipped (defective) location (sector or track, depending upon the sparing scheme chosen).

Number of defects: Integer [nonnegative] specifying the number of previously detected defective media locations (sectors or tracks, depending upon the sparing scheme chosen) that have been remapped to alternate physical locations. Subsequent lines in the parameter file describe how the logical-to-physical mapping has changed as a result. Each such line has the form "**Defect X Y**" where X indicates the original (defective) location and Y indicates the replacement location. (Note that **X** and **Y** will both be either a physical sector number or a physical track number, depending on the sparing scheme chosen.)

3.2.5 Queue/Scheduler Subcomponents

Scheduling policy: Integer [1–21] specifying the primary scheduling algorithm employed for selecting the next request to be serviced. A large set of algorithms have been implemented, ranging from common choices like First-Come-First-Served (FCFS) and Shortest-Seek-Time-First (SSTF) to new algorithms like Shortest-Positioning-(w/Cache)-Time-First (described in [Worthington94]). The full list of mappings from values to corresponding algorithms can be found at the top of the source file named "disksim_ioqueue.c".

Cylinder mapping strategy: Integer [0–6] specifying the level of detail of physical data layout information available to the scheduler. 0 indicates that the only information available to the scheduler is the logical block numbers specified in the individual requests. 1 indicates that the scheduler has access to information about zone boundaries, the number of physical sectors/zone, and the number of physical sectors/track in each zone. 2 indicates that the scheduler also has access to the layout of spare sectors or tracks in each zone. 3 indicates that the scheduler also has access to the list of any slipped sectors/tracks. 4 indicates that the scheduler also has access to the list of any remapped sectors/tracks, thereby providing an exact data layout (logical-to-physical mapping) for the disk. 5 indicates that the scheduler uses the cylinder number given to it with the request, allowing experiments with arbitrary mappings. In particular, some traces include the cylinder number as part of the request record. 6 indicates that the scheduler only has access to (an approximation of) the mean number of sectors per cylinder. The value used in this case is that specified in the disk parameter "Avg. sectors per cylinder".

Write initiation delay: Float [nonnegative milliseconds] specifying an approximation of the write request processing overhead(s) performed prior to any mechanical positioning delays. This value is used by scheduling algorithms that select the order of request service based (at least in part) on expected positioning delays.

Read initiation delay: Float [nonnegative milliseconds] specifying an approximation of the read request processing overhead(s) performed prior to any mechanical positioning delays. This value is used by scheduling algorithms that select the order of request service based (at least in part) on expected positioning delays.

Sequential stream scheme: Integer comprising a Boolean Bitfield specifying the type of sequential stream detection and/or request concatenation performed by the scheduler (see [Worthington95a] for additional details). Bit 0 indicates whether or not sequential read requests are concatenated by the scheduler. Bit 1 indicates whether or not sequential write requests are concatenated by the scheduler. Bit 2 indicates whether or not sequential read requests are always scheduled together. Bit 3 indicates whether or not sequential write requests are always scheduled together. Bit 4 indicates whether or not sequential requests of any kind (e.g., interleaved reads and writes) are always scheduled together.

Maximum concat size: Integer [nonnegative sectors] specifying the maximum request size resulting from concatenation of sequential requests. That is, if the sum of the sizes of the two requests to be concatenated exceeds this value, the concatenation will not be performed by the scheduler.

Overlapping request scheme: Integer [0-2] specifying the scheduler's policy for dealing with overlapping requests. 0 indicates that overlapping requests are treated as independent. 1 indicates that requests that are completely overlapped by a completed request that arrived after them are subsumed by that request. 2 augments this policy by also allowing read requests that arrive after the completed overlapping request to be subsumed by it, since the necessary data are known. This support was included for experiments in [Ganger95] in order to partially demonstrate the correctness problems of open workloads (e.g., feedback-free request traces). In most real systems, overlapping requests are almost never concurrently outstanding.

Sequential stream diff maximum: Integer [nonnegative sectors] specifying the maximum distance between two "sequential" requests in a sequential stream. This allows requests with a small stride or an occasional "skip" to still be considered for inclusion in a sequential stream.

Scheduling timeout scheme: Integer [0, 1, or 2] specifying the type of multi-queue timeout scheme implemented. 0 indicates that requests are not moved from the *base* queue to a higher-priority queue because of excessive queueing delays. 1 indicates that requests in the base queue whose queueing delays exceed the specified timeout value (see below) will be moved to one of two higher-priority queues (the *timeout* queue or the *priority* queue) based on the scheduling priority scheme (see below). 2 indicates that requests in the base queue whose queueing delays exceed half of the specified timeout value (see below) will be moved to the next higher priority queue (the timeout queue). Furthermore, such requests will be moved to the highest priority queue (the priority queue) if their total queueing delays exceed the specified timeout value (see below).

Timeout time/weight: Overloaded Float [nonnegative] specifying either the timeout value (in seconds) for excessive queueing delays or the time/aging factor used in calculating request priorities for various age-sensitive scheduling algorithms. The time/aging factor is additive for some algorithms and multiplicative for others.

Timeout scheduling: Integer [1–21] specifying the scheduling algorithm employed for selecting the next request to be serviced from the *timeout* queue. The options are the same as those available for the "Scheduling policy" parameter above.

Scheduling priority scheme: Boolean [1 or 0] specifying whether or not requests flagged as high priority (i.e., time-critical or time-limited requests [Ganger93]) are automatically placed in the highest-priority queue (the *priority* queue).

Priority scheduling: Integer [1–21] specifying the scheduling algorithm employed for selecting the next request to be serviced from the *priority* queue. The options are the same as those available for the "Scheduling policy" parameter above.

3.2.6 Disk Block Cache Subcomponents

The following parameters configure the generic disk block cache subcomponent, which is currently only used in the intelligent controller type (3). The disk module has its own cache submodule, which is configured by disk parameters described in section 3.2.4.

Cache size (in 512B blks): Integer [nonnegative 512B blocks] specifying the total size of the cache

Cache segment count (SLRU): Integer [positive] specifying the number of segments for the segmented-LRU algorithm [Karedla94] if it is specified as the cache replacement algorithm (see below). Note that the value has a hard-coded upper limit (necessary for allocating certain static arrays) that is set to 10 in the current release of DiskSim.

Cache line size (in blks): Integer [nonegative 512B blocks] specifying the cache line size (i.e., the unit of cache space allocation/replacement).

Cache blocks per bit: Integer [positive 512B blocks] specifying the number of blocks covered by each "present" bit and each "dirty" bit. The value must divide the cache line size evenly. Higher values (i.e., coarser granularities) can result in increased numbers of installation reads (i.e., fill requests required to complete partial-line writes [Otoole94]).

Cache lock granularity: Integer [positive 512B blocks] specifying the number of blocks covered by each lock. The value must divide the cache line size evenly. Higher values (i.e., coarser granularities) can lead to increased lock contention.

Cache shared read lock: Boolean [1 or 0] specifying whether or not read locks are sharable. If false (0), read locks are exclusive.

Cache max request size: Integer [nonnegative] specifying the maximum request size to be served by the cache. This value does not actually affect the simulated cache's behavior. Rather, higher-level system components (e.g., the device driver in DiskSim) acquire this information at initialization time and break up larger requests to accommodate it. 0 indicates that there is no maximum request size.

Cache replacement policy: Integer [1–4] specifying the line replacement policy. 1 indicates First-In-First-Out (FIFO), 2 indicates segmented-LRU [Karedla94], 3 indicates random replacement, and 4 indicates Last-In-First-Out (LIFO).

Cache allocation policy: Integer [0 or 1] specifying the line allocation policy. 0 indicates that the cache replacement policy is strictly followed; if the selected line is dirty, the allocation waits for the required write-back request to complete. 1 indicates that "clean" lines are considered for replacement prior to "dirty" lines (and background write-back requests are issued for each dirty line considered).

Cache write scheme: Integer [1, 2, or 3] specifying the policy for handling write requests. 1 indicates that new data are always synchronously written to the backing store before indicating completion. 2 indicates a write-through scheme where requests are immediately initiated for writing out the new data to the backing store, but the original write requests are considered complete as soon as the new data is cached. 3 indicates a write-back scheme where completions are reported immediately and dirty blocks are held in the cache for some time before being written out to the backing store.

Cache flush policy: Integer [0 or 1] specifying the policy for flushing dirty blocks to the backing store (assuming a write-back scheme for handling write requests). 0 indicates that dirty blocks are written back "on demand" (i.e., only when the allocation/replacement policy needs to reclaim them). 1 indicates write-back requests are periodically initiated for all dirty cache blocks.

Cache flush period: Float [nonnegative milliseconds] specifying the time between periodic write-backs of all dirty cache blocks (assuming a periodic flush policy).

Cache flush idle delay: Float [-1.0 or nonnegative milliseconds] specifying the amount of contiguous idle time that must be observed before background write-backs of dirty cache blocks are initiated. Any front-end request processing visible to the cache resets the idle timer. -1.0 indicates that idle background flushing is disabled.

Cache flush max line cluster: Integer [positive] specifying the maximum number of cache lines that can be combined into a single write-back request (assuming "gather" write support).

Cache prefetch type (read): Integer [0–3] specifying the prefetch policy for handling read requests. Prefetching is currently limited to extending requested fill accesses to include other portions of requested lines. 0 indicates that prefetching is disabled. 1 indicates that unrequested data at the start of a requested line are prefetched. 2 indicates that unrequested data at the end of a requested line are prefetched. 3 indicates that any unrequested data in a requested line are prefetched (i.e., full line fills only).

Cache prefetch type (write): Integer [0–3] specifying the prefetch policy for handling installation reads (caused by write requests). Prefetching is currently limited to extending the requested fill accesses to include other portions of the requested lines. 0 indicates that prefetching is disabled. 1 indicates that unrequested data at the start of a requested line are prefetched. 2 indicates that unrequested data at the end of a requested line are prefetched. 3 indicates that any unrequested data in a requested line are prefetched (i.e., full line fills only).

Cache line-by-line fetches: Boolean [0 or 1] indicating whether or not every requested cache line results in a separate fill request. If false (0), multi-line fill requests can be generated when appropriate.

Cache scatter/gather max: Integer [nonnegative] specifying the maximum number of non-contiguous cache lines (in terms of their memory addresses) that can be combined into a single disk request, assuming that the correspond to contiguous disk addresses. (DiskSim currently treats every pair of cache lines as non-contiguous in memory.) 0 indicates that any number of lines can be combined into a single request (i.e., there is no maximum).

3.3 I/O Subsystem Interconnection Specifications

The allowed interconnections are roughly independent of the components themselves except that a device driver must be at the "top" of any subsystem and storage devices must be at the "bottom." Exactly one or two controllers must be between the device driver and each disk, with a bus connecting each such pair of components along the path from driver to disk. Each disk or controller can only be connected to one bus from the host side of the subsystem. A bus can have no more than 15 disks or controllers attached to it. A controller can have no more than 4 back-end buses (use of more than one is not well tested). The one allowable device driver is connected to the top-level bus.

The specification of the physical component organization occurs in a specific order: device driver, controllers, buses.

The device driver's interconnections are specified by three fixed lines in the parameter file:

Driver #1: the first (and only) device driver.

of connected buses: Integer [must be 1] specifying that the top-level bus is connected to the device driver.

Connected buses: Integer [positive] specifying the bus number (bus numbers are assigned in ascending order of the bus specifications, starting with "1" for the first specified bus) of the top-level bus. The top-level bus should be a "shared" bus (i.e., bus type 2).

The necessary controller interconnections are described by one or more sets of parameters:

Controller #X: where X is an Integer [positive] or String ["III–III"] specifying the controller or the range of controllers covered by this specification. A range of controller numbers can be specified using the formatted String, which contains two Integers [positive (III)] separated by a dash. Controller numbers are assigned in ascending order of the controller specifications, starting with "1" for the first specified controller.

of connected buses: Integer [positive] specifying the number of buses connected to each of the controllers covered by this specification.

For each attached bus, one instance of the following line must be present in the specification.

Connected buses: Integer [positive] or String [#±III] specifying the bus or buses connected to each of the controllers covered by this specification. If the String format is used, the Integer value (±III) following the "#" in the String is added to the corresponding controller number to determine the bus number (for each controller). For example, if the first line of a controller parameter set specified "Controller #5–10" and the value of this line was "#+2", bus 7 would be attached to controller 5, bus 8 would be attached to controller 6, and so on. If more than one controller is described by this specification, then the String format must be used.

The necessary bus interconnections are described by one or more sets of parameters:

Bus #X: where X is an Integer [positive] or String ["III–III"] specifying the bus or the range of buses covered by this specification. A range of bus numbers can be specified using the formatted String, which contains two Integers [positive (III)] separated by a dash. Bus numbers are assigned in ascending order of the bus specifications, starting with "1" for the first specified bus.

of utilized slots: Integer [positive] specifying the number of disks and/or controllers connected to each of the buses covered by this specification.

One or more instances of the following line must be present in the parameter file in order to specify the attachment of disks and/or controllers to the bus or buses covered by this specification.

Slots: Two Strings, the first String ["Controllers" or "Disks"] specifying whether the current line refers to attached controllers or disks, and the second String ["III", "III–III", or "#±III"] specifying which instances of the disks or controllers are connected to the bus(es) covered by this specification. The second string has three potential formats: (1) an Integer [positive (III)] specifying a component number, (2) two Integers [positive (III)] separated by a dash, specifying a range of component numbers (e.g., "5–8"), or (3) a "#" followed by an Integer value (±III) that is added to the corresponding bus number to determine the attached component number. For example, if the first line of a bus parameter set specified "Bus #5–10" and the String values on this line were "Disk" and "#+2", disk 7 would be attached to bus 5, disk 8 would be attached to bus 6, and so on. If more than one bus is described by this specification, then the third format must be used.

3.4 Rotational Synchronization of Disks

DiskSim can be configured to simulate rotationally synchronized disks via the following parameters. Rotationally synchronized disks are always at exactly the same rotational offset, which requires that they begin the simulation at the same offset and rotate at the same speed. Non-synchronized disks are assigned a random initial rotational offset at the beginning of the simulation and are individually assigned a rotational speed based on the appropriate disk parameters.

Number of synchronized sets: Integer [nonnegative] specifying the number of separate sets of disks that are rotationally synchronized.

One instance of the following pair of parameters must be present in the parameter file for each synchronized set.

Number of disks in set #X: Integer [positive] specifying the number of disks in the Xth set, where X is an Integer [positive] corresponding to the number of the set being specified. The sets must be described in ascending order, starting with "1".

Synchronized disks: String ["III–III"] specifying the range of disks to be synchronized. Two Integers [positive (III)] in the String are separated by a dash and specify the first and last disks of the set (in order of configuration specification, with the first specified disk being "1").

3.5 Disk Array Data Organizations

DiskSim can simulate a variety of logical data organizations [Ganger94], including striping and various RAID architectures. Although DiskSim is organized so as to allow for such organizations both at the system-level (i.e., at front end of the device drivers) and at the controller-level, only system-level organizations are supported in the first released version. Each logical organization is configured with the following parameters:

Organization #X: Four Strings specifying the general description of the Xth logical organization of this specification, where X is an Integer [positive] corresponding to the number of the data organization being specified. The sets must be described in ascending order, starting with "1".

- The first String ["Array" or "Parts"] specifies how the logical data organization is addressed. "Array" indicates that there is a single logical device number for the entire logical organization. "Parts" indicates that back-end storage devices are addressed as though there were no logical organization, and requests are re-mapped appropriately.
- The second String ["Asis", "Striped", "Random", or "Ideal"] specifies the data distribution scheme (which is orthogonal to the redundancy scheme). "Asis" indicates that no re-mapping occurs. "Striped" indicates that data are striped over the organization members. "Random" indicates that a random disk is selected for each request. "Ideal" indicates that an idealized data distribution (from a load balancing perspective) should be simulated by assigning requests to disks in a round-robin fashion. Note that the last two schemes do not model real data layouts. In particular, two requests to the same block will often be sent to different devices. However, these data distribution schemes are useful for investigating various load balancing techniques [Ganger93a].
- The third String ["Noredun", "Shadowed", "Parity_disk", or "Parity_rotated"] specifies the redundancy scheme (which is orthogonal to the data distribution scheme). "Noredun" indicates that no redundancy is employed.

"Shadowed" indicates that one or more replicas of each data disk are maintained. "Parity_disk" indicates that one parity disk is maintained to protect the data of the other organization members. "Parity_rotated" indicates that one disk's worth of data (spread out across all disks) are dedicated to holding parity information that protects the other N-1 disks' worth of data in an N-disk organization.

- The fourth String ["Whole" or "Partial"] specifies whether the data organization's component members are entire disks ("Whole") or partial disks ("Partial"). Only the former option is supported in the first released version of DiskSim.

Number of disks: Integer [positive] specifying the number of disks comprising the logical organization.

Disks: String ["III–III"] specifying the range of disks comprising the logical organization. Two Integers [positive (III)] in the String are separated by a dash and specify the first and last disks in the organization (in order of configuration specification, with the first specified disk being "1").

High-level device number: Integer [positive] specifying the device number used to address the logical organization if "Array" addressing is specified above. Otherwise, this parameter is ignored.

Stripe unit (in sectors): Integer [nonnegative 512B sectors] specifying the stripe unit size. 0 indicates fine-grained striping (e.g., bit or byte striping), wherein all data disks in the logical organization contain an equal fraction of every addressable data unit.

Synch writes for safety: Boolean [1 or 0] specifying whether or not an explicit effort should be made to do the N+1 writes of a parity-protected logical organization at "the same time" when handling a front-end write request with the read-modify-write (RMW) approach to parity computation. If true (1), then all reading of old values (for computing updated parity values) must be completed before the set of back-end writes is issued. If false (0), then each back-end write is issued immediately after the corresponding read completes (perhaps offering improved performance).

Number of copies: Integer [positive] specifying the number of copies of each data disk if the logical organization employs "Shadowed" redundancy. Otherwise, this parameter is ignored.

Copy choice on read: Integer [1–6] specifying the policy used for selecting which disk (from a set of "Shadowed" replicas) should service a given read request (since any of them can potentially do so). 1 indicates that all read requests are sent to a single primary replica. 2 indicates that one of the replicas should be randomly selected for each read request. 3 indicates that requests should be assigned to replicas in a round-robin fashion. 4 indicates that the replica that would incur the shortest seek distance should be selected and ties are broken by random selection. 5 indicates that the replica that has the shortest request queue should be selected and ties are broken by random selection. 6 indicates that the replica that has the shortest request queue should be selected and ties are broken by policy 4 (see above). This parameter is ignored if "Shadowed" replication is not chosen.

RMW vs. reconstruct: Integer [nonnegative] specifying the breakpoint in selecting Read-Modify-Write (RMW) parity updates (verses complete reconstruction) as the fraction of data disks that are updated. If the number of disks updated by the front-end write request is smaller than the breakpoint, then the RMW of the "old" data, "old" parity, and "new" data is used to compute the new parity. Otherwise, the unmodified data in the affected stripe are read from the corresponding data disks and combined with the new data to calculate the new parity. This parameter is ignored unless some form of parity-based replication is chosen.

Parity stripe unit: Integer [nonnegative 512B sectors] specifying the stripe unit size used for the "Parity_rotated" redundancy scheme. This parameter is ignored for other schemes. The parity stripe unit size does not have to be equal to the stripe unit size, but one must be a multiple of the other. Use of non-equal stripe unit sizes for data and parity has not been thoroughly tested in the current release of DiskSim.

Parity rotation type: Integer [1–4] specifying how parity is rotated among the disks of the logical organization. The four options, as described in [Lee91], are left symmetric (1), left asymmetric (2), right asymmetric (3), and right symmetric (4). This parameter is ignored unless "Parity_rotated" redundancy is chosen.

The next four parameters configure DiskSim's per-logorg mechanism for collecting information about instantaneous per-device queue lengths at regular intervals.

Time stamp interval: Float [nonnegative milliseconds] specifying the interval between "time stamps". A value of 0.0 for this parameter disables the time stamp mechanism.

Time stamp start time: Float [nonnegative milliseconds] specifying the simulated time (relative to the beginning of the simulation) of the first time stamp.

Time stamp stop time: Float [nonnegative milliseconds] specifying the simulated time (relative to the beginning of the simulation) of the last time stamp.

Time stamp file name: String specifying the name of the output file to contain a log of the instantaneous queue lengths of each of the organization's back-end devices at each time stamp. Each line of the output file corresponds to a single time stamp and contains the queue lengths of each device separated by white space. A value of "0" or of "null" disables this feature (as does disabling the time stamp mechanism).

3.6 Process-Flow Parameters

PRINTED PROCESS-FLOW STATISTICS: This section contains a series of Boolean [1 or 0] parameters that specify whether or not particular groups of statistics are reported. The different print-control values are identified with individual statistics in section 5.

Number of processors: Integer [positive] specifying the number of processors used by the simple system-level model. These processors (and, more generally, DiskSim's system-level model) are only used for the synthetic generation module.

Process-Flow Time Scale: Float [nonnegative] specifying a multiplicative scaling factor for computation times "executed" by a simulated processor. For example, 2.0 doubles each computation time, and 0.5 halves each computation time.

The various parameters involved with configuring the synthetic workload generation module are described in section 4.2.

4 Input workloads: traces and synthetic workloads

DiskSim can be exercised with I/O requests in several ways, including external traces, internally-generated synthetic workloads, and interactions with a containing simulation environment (e.g., a full system simulator). This section describes each of these options.

4.1 Traces

DiskSim can accept traces in several formats, and new formats can be added with little difficulty. This subsection describes the default input format and briefly describes how to add support for new trace formats. The DiskSim 1.0 distribution supports the default format ("ascii"), a validation trace format ("validate"), the raw format ("raw") of the disk request traces described in [Ganger93, Ganger93a], and the raw format ("hpl", or "hpl2" if the trace file header has been stripped) of the disk request traces described in [Ruemmler93].

4.1.1 Default format

The default input format is a simple ASCII stream (or file), where each line contains values for five parameters (separated by white space) describing a single disk request. The five parameters are:

1. **Request arrival time**: Float [nonnegative milliseconds] specifying the time the request "arrives" relative to the start of the simulation (at time 0.0). Requests must appear in the input stream in ascending time order.
2. **Device number**: Integer specifying the device number (i.e., the storage component that the request accesses). The device mappings (see section 3), if any, are applied to this value.
3. **Block number**: Integer [nonnegative] specifying the first device address of the request. The value is specified in the appropriate access unit of the logical device in question, which may be modified by the device mappings (see section 3).
4. **Request size**: Integer [positive] specifying the size of the request in device blocks (i.e., the access unit of the logical device in question).
5. **Request flags**: Hex Integer comprising a Boolean Bitfield specifying additional information about the request. For example, bit 0 indicates whether the request is a read (1) or a write (0). Other bits specify information that is most appropriate to a full system simulation environment (e.g., request priority). Valid bitfield values are listed in "disksim_global.h".

An example trace in this format is included with the distribution.

4.1.2 Adding support for new trace formats

Adding support for a new trace format requires only a few steps:

1. Add a new trace format constant to "disksim_global.h".

2. Select a character string to represent the trace format on the command line. Add a format name comparison to “iotrace_set_format” in “disksim_iotrace.c”.

3. Create a procedure (“iotrace_XXXX_get_ioreq_event”) in “disksim_iotrace.c” to read a single disk request description from an input trace of the new format (“XXXX”) and construct a disk request event in the internal format (described briefly below). The functions “iotrace_read_[char,short,int32]” can simplify this process. Incorporate the new function into main switch statement in “iotrace_get_ioreq_event”. The internal DiskSim request structure (at request arrival time) is not much more complicated than the default (ascii) trace format. It contains “time”, “devno”, “blkno”, “bcount”, and “flags” fields that correspond to the five non-auxiliary fields described above. The other fields do not have to be initialized, except that “opid” should be zeroed. See “iotrace_ascii_get_ioreq_event” for an example.

4. If the trace file has an informational header (useful or otherwise), then create a procedure (“iotrace_XXXX_initialize_file”) in “disksim_iotrace.c” and add it into the if/else statement in “iotrace_initialize_file”.

4.2 Synthetic workloads

DiskSim includes a simple synthetic workload generating module that can be used to drive storage subsystem simulations. The parameter file specifies the number of generators (similar to processes) and the characteristics of the workloads generated by each. Each synthetic generator “executes” as a process in a very simple system-level model, issuing I/O requests after periods of “think time” and, when appropriate, waiting for them to complete. This module can be configured to generate a wide range of synthetic workloads, both with respect to the disk locations accessed and the request arrival times.

4.2.1 Configuration

The synthetic generation module is configured with the parameters (described below) specified in the last section of the parameter file. The other “Process-Flow Input Parameters” are also relevant. In particular, the number and time scaling of the simulated processors is important, since these processors “execute” the inter-request think times as computation times (one process's computation per processor at a time).

The parameters for the synthetic generation module are:

Number of generators: Integer [positive] specifying the number of independent, concurrent, request-generating processes.

Number of I/O requests to generate: Integer [positive] specifying the maximum number of I/O requests to generate before ending the simulation run. A simulation run continues until either the specified number of requests is generated or the maximum simulation time (see below) is reached.

Maximum time of trace generated (in seconds): Integer [positive seconds] specifying the maximum simulated time of a simulation run driven by a synthetic workload. A simulation run continues until either the maximum number of requests (see above) is generated or the specified maximum simulation time is reached.

System call/return with each request: Boolean [1 or 0] specifying whether or not each request occurs within the context of a system call (which may affect the behavior of the associated process in the system-level model). If true (1), each request will be preceded by a system call event and followed by a system call return event.

Think time from call to request: Float [milliseconds] specifying the think time (i.e., computation time) between the system call event and the disk request event. This parameter is only relevant if the above Boolean parameter is set to true (1).

Think time from request to return: Float [milliseconds] specifying the think time (i.e., computation time) between the disk request event and the system call return event. This parameter is only relevant if the above Boolean parameter is set to true (1).

One or more instances of the following parameters must be present in the parameter file in order to specify the configuration of the individual generator(s).

Generator description #X: where X is an Integer [positive] specifying the number of the generator type being specified. Generator types are assigned in ascending order of the generator specifications, starting with "1" for the first specified generator type.

Generators with description: Integer [positive] specifying the number of generators (out of the total number specified) that conform to the current specification.

Storage capacity per device (in blocks): Integer [positive] specifying the number of unique storage addresses per storage device (in the corresponding device's unit of access) accessible to generators of this type.

Number of storage devices: Integer [positive] specifying the number of storage devices accessible to generators of this type. The generated device numbers will range from 0 to this value minus one.

Blocking factor: Integer [positive] specifying a unit of access for generated requests that is a multiple of the storage devices' unit of access. All generated request starting addresses and sizes will be a multiple of this value.

Probability of sequential access: Float [0.0–1.0] specifying the probability that a generated request is sequential to the immediately previous request. A sequential request starts at the address immediately following the last address accessed by the previously generated request.

Probability of local access: Float [0.0–1.0] specifying the probability that a generated request is "local" to the immediately previous request. A local request begins some short distance away from the previous request's starting address, where the distance is computed via a random variable definition described below.

These last two values (probabilities) determine how a generated request's starting address is assigned. Their sum must be less than or equal to 1.0. Each request's starting address is sequential, local or random. A random request is assigned a device and starting address from a uniform distribution spanning the entire available storage space (as specified by the above parameters).

Probability of read access: Float [0.0–1.0] specifying the probability that a generated request is a read. (Otherwise, it is a write.)

Probability of time-critical request: Float [0.0–1.0] specifying the probability that a generated request is time-critical. That is, the corresponding generator process "blocks" and waits for the request to complete before continuing with its sequence of work (i.e., its next think time) [Ganger93, Ganger95].

Probability of time-limited request: Float [0.0–1.0] specifying the probability that a generated request is time-limited. That is, the corresponding generator process "blocks" and waits for the request to complete (if it is not already complete) after a given amount of think time (specified by the below "time limit" parameters) [Ganger93, Ganger95].

These last two values (probabilities) determine how a generated request's criticality is assigned. Their sum must be less than or equal to 1.0. Each request is time-critical, time-limited or time-noncritical. A generator process nevers "blocks" and waits for time-noncritical requests; it simply generates them and continues with its work.

With these probabilities, the synthetic generators can be configured to emulate open and closed subsystem models, as well as a range of intermediate options. An open subsystem model (like most trace-driven simulations) generates new requests independently of the completion times of previous requests. It can be created by setting the probabilities of time-critical and time-limited requests both to zero (0.0) and configuring the system-level model to have the same number of processors as there are generators. The inter-request computation times will therefore be the inter-arrival times (per generator). A closed subsystem model (like many queueing models) generates a new request only after the previous request completes, keeping a constant number of requests in the system (either hidden in think times or being serviced). It can be created by setting the probability that a request is time-critical to one (1.0). Setting the inter-request computation times (below) to zero (0.0) eliminates think times and results in a constant number of requests "in service" at all times.

The remaining six parameters specify the random variable distributions of the various times, sizes, and distances encountered when generating streams of requests. The values for a parameter consists of an Integer [0–4] specifying the type of distribution followed by up to three additional values necessary to fully specify the configuration of the chosen distribution. The number of parameters and their purposes depends on the value given for the type of distribution:

- 0 indicates a uniform distribution, requiring two additional Floats specifying minimum and maximum values.
- 1 indicates a normal distribution, requiring two additional Floats specifying mean and variance values. As the second value is a variance, it must be nonnegative.
- 2 indicates an exponential distribution, requiring two additional Floats specifying base and mean values.
- 4 indicates a poisson distribution, requiring two additional Floats specifying base and mean values.
- 5 indicates a "two value" distribution, requiring three additional Floats specifying a default value, a secondary value, and a probability indicating how often the secondary value should be returned. As the last Float is a measure of probability, it must be between 0.0 and 1.0.

All of the distributions are computed as specified, but value generation is repeated whenever an illegal value results (e.g., a negative inter-arrival time).

Time-limited think times (in milliseconds): A random variable distribution specifying the time limit for a time-limited request. Note that the generated time limit (i.e., the computation time occuring before the generator process "blocks" and waits for the request to complete) may differ from the actual time limit (due to CPU contention).

General inter-arrival times (in milliseconds): A random variable distribution specifying the inter-request think time preceding the generated request if the generated request's starting address is unrelated to the previous request's starting address (i.e., if the generated request's address is "random" rather than "sequential" or "local").

Sequential inter-arrival times (in milliseconds): A random variable distribution specifying the inter-request think time preceding the generated request if the generated request's starting address is "sequential" to the previous request's starting address.

Local inter-arrival times (in milliseconds): A random variable distribution specifying the inter-request think time preceding the generated request if the generated request's starting address is "local" to the previous request's starting address.

Local distances (in blocks): A random variable distribution specifying the distance from the previous request's starting address when generating a "local" request's starting address.

Size (in blocks): A random variable distribution specifying the request size.

4.3 Incorporating DiskSim into system-level simulators

With a modicum of effort, DiskSim can be incorporated into a full system-level simulator in order to provide accurate timings for the handling of storage I/O requests. This section briefly describes one method for doing so, and "disksim_interface.c" stubs out a possible implementation. Using this approach (which assumes only a few characteristics of the system simulator), DiskSim will act as a slave of the system simulator, providing disk request completion indications in time for an interrupt to be generated in the system simulation. Specifically, DiskSim code will only be executed when invoked by one of the following procedures, called as appropriate by the containing simulator:

- **disksim_initialize**: for initializing the DiskSim state.
- **disksim_shutdown**: for printing statistics at the end of a simulation run.
- **disksim_dump_stats**: for printing the current running statistics during the simulation (e.g., at a statistics checkpoint).
- **disksim_internal_event**: for "calling back" into DiskSim so that it can update its internal simulation "time" to match the system-level simulator's global "time" and handle any internal DiskSim events that occur in the intervening time. Note that this function is called by the system-level simulator *on behalf* of DiskSim, since DiskSim no longer has control over the global simulation "time." Additional details are given below.
- **disksim_request_arrive**: for issuing an I/O request into DiskSim.

Using this interface requires only two significant functionalities of the system simulation environment:

1. The ability to function correctly without knowing when a disk request will complete at the time that it is initiated. The system simulation will be informed at some later point (in its view of time) that the request is completed. At this time, the appropriate "disk request completion" interrupt could be inserted into the system simulation.
2. The ability for DiskSim to register callbacks with the system simulation environment. That is, this interface code must be able to request (of the sytem-level simulator) an invocation of a callback function (such as

disksim_internal_event, described above) when the simulated time reaches a DiskSim-specified value. It is also helpful (but not absolutely necessary) to be able to "de-schedule" a callback at some point after it has been requested. For example, a callback requested to indicate the end of some disk prefetching activity may be superceded by a new request arriving at the disk (and interrupting the ongoing prefetch).

If the actual content on the disk media (i.e., the "data") must be maintained during the course of a system-level simulation, this functionality can easily be provided by code outside of DiskSim, which does not itself provide such functionality.

5 The output file

At the beginning of the simulation, the values for the numerous configuration parameters are copied into the beginning of the output file. The remainder of the output file contains the aggregate statistics of the simulation run, including both the characteristics of the simulated workload (if enabled) and the performance indicators of the various storage components. Each line of the latter portion of the output file (excluding the statistic distributions) is unique, simplifying the process of searching through the file for a particular result.

DiskSim collects a large number of statistics about the simulated storage components. As discussed in section 3, the size of the output file can be reduced by configuring DiskSim not to report undesired sets of statistics. At each statistic is described below, it will also be noted whether or not the statistic can be pruned from the output file via corresponding Boolean (enable/disable) input parameters.

5.1 The STATDEFS file

Although some of the results collected by DiskSim are simple counts or sums, aggregate statistics (average, standard deviation, distribution) are collected for many values. In particular, statistics reported as distributions make use of the STATDEFS file for configuring the "bins" which capture specific ranges of observed values. Each statistic description consists of four lines, as described below. Comments or other extraneous information may be placed within the file as long as each 4-line description is contiguous.

1. String specifying the name of the statistic being described. DiskSim searches for this value on a line by itself to identify the beginning of the corresponding statistic description. So, the order of the statistics in the STATDEFS file is unimportant.

2. **Distribution size**: Integer [positive] specifying the number of bins into which to partition the observed values. When this value is less than or equal to the internal DiskSim constant DISTSIZE, the entire distribution are reported on two lines (for output file compactness and readability). The output for larger distributions consists of one bin per line.

3. **Scale/Equals**: Two Integers separated by a "/". The first Integer [nonzero] specifies a multiplication factor to be applied to observed values before selecting an appropriate bin. It is useful mainly because the bin boundaries are specified as Integers (see below). For example, if a specific "response time" statistic's distribution is specified using microsecond-based boundaries, the "Scale" value should be set to 1000 (since internal times in DiskSim millisecond-based). The second Integer [nonnegative] specifies how many of the bins (starting from the first one) should collect only observed values that are exactly equal to the specified boundary, rather than values that are less than the boundary (see below).

4. A description of the bin boundaries, in one of two formats. If the number of bins is less than or equal to the internal DiskSim constant DISTSIZE, then this line contains nine Integers. If the "Equals" value (see above) is set to N, the first N Integers specify bins that will hold exact observed values rather than ranges of observed values. The remaining Integers specify bins holding observed values below the specified Integer boundary. When DiskSim categorizes an observed value, the bins are checked in the sequence in which they are found on the line. So, bin boundaries beyond the "Equals" values should be in ascending order to avoid unclear results.

If the number of bins is greater than DISTSIZE, the format of this line is a String ["Start III step III grow III"] containing three Integers (III). The first specifies the first bin boundary, the second specifies a constant value to be added to the current boundary value when computing the next boundary value, and the third specifies a percent of the previous bin value to be added to the current boundary value when computing the next boundary value. The combination of a multiplicative scaling factor ("Scale"), an additive step function ("step"), and a multiplicative step function ("grow") provides the flexibility to concisely specify a wide range of typical statistic distributions.

5.2 Simulation results

Each statistic in the output file is identified by a unique string. A typical single-value statistic is reported as "SSS: VVV" where the initial String (SSS) uniquely identifies the statistic and the value (VVV) specifies the observed value. In some cases a single value is inadequate to describe the statistic, so a set of four aggregate statistics are reported instead: the average value, the standard deviation ("std.dev."), the maximum value, and the distribution (as specified by the corresponding entry in the STATDEF file).

The format for reporting the first three aggregates mentioned above is "SSS: VVV", where the initial String (SSS) contains the context of the statistic (e.g., "Disk #2"), the name of the statistic used to index into the STATDEF file, and the aggregate type ["average", "std.dev.", or "maximum"]. The corresponding result value (VVV) is specified on the same line. The format for reporting distributions is similar to the other three aggregates, but the value (VVV) is reported on lines following the indentifying String (SSS). If the number of bins specified in the corresponding entry in the STATDEFS file is greater than DISTSIZE, then the distribution is reported on two lines: one for the bin boundaries and one for the observed bin counts. Otherwise, the distribution is reported on the next N lines, where N is the number of bins. Each line contains four values: (1) the bin boundary, (2) the count of observed values falling into the bin, (3) the measured probability that an observed value falls into the bin (i.e., the count divided by the number of observations), and (4) the measured probability that an observed value falls in the bin or any previous bin.

Each individual statistic found in the output file is described below. *Unless otherwise specified, all statistics measuring simulation "time" are reported in milliseconds.*

Total time of run: The simulated time at the end of the simulation run.

Warm-up time: The simulation warm-up time not covered by the reported statistics.

The next set of statistics falls into one of several categories, depending on the style of input. If synthetic generation is used, then statistics for the simple system-level model are reported (section 5.2.1). If a validation trace is used, then statistics about the behavior measured for the real disk are reported (section 5.2.2). If the traces described in [Ruemmler93] (referred to in DiskSim as "HPL" traces) are used, then statistics about performance observed for the traced system are reported (section 5.2.3). If any other external trace is used, then no statistics are reported in this section of the output file, and the I/O driver statistics are next in the output file (section 5.2.5), followed by the disk drive statistics (section 5.2.6), controller statistics (section 5.2.7), and bus statistics (section 5.2.8).

5.2.1 Process-flow statistics

The following statistics are reported only when the internal synthetic workload generator is enabled.

CPU Total idle milliseconds: the sum of the idle times for all CPU's.

CPU Idle time per processor: the average per-CPU idle time.

CPU Percentage idle cycles: the average percentage of time that each CPU spent idle.

CPU Total false idle ms: the sum of the false idle times for all CPU's. "False idle time" is that time that a processor spends idle because processes are blocked waiting for I/O (e.g., disk requests), as opposed to real idle time during which there is no work for the CPU to do.

CPU Percentage false idle cycles: the average percentage of each CPU's time that was consumed by false idle time.

CPU Total idle work ms: the sum of the idle work times for all CPU's. "Idle work time" is useful computation (e.g., interrupt handlers and background tasks) completed while in the idle state (because no user processes are runnable).

CPU Context Switches: the number of context switches.

CPU Time spent context switching: the aggregate CPU time consumed by context switch overheads on all CPU's.

CPU Percentage switching cycles: the average percentage of each CPU's time that was consumed by context switching overheads.

CPU Number of interrupts: the total number of interrupts received by all CPU's.

CPU Total time in interrupts: the total computation time consumed by interrupt handlers.

CPU Percentage interrupt cycles: the average percentage of each CPU's time that was consumed by interrupt handling.

CPU Time-Critical request count: the total number of time-critical requests generated.

CPU Time-Critical Response time stats: aggregate statistics for the response times observed for all time-critical requests.

CPU Time-Limited request count: the total number of time-limited requests generated.

CPU Time-Limited Response time stats aggregate statistics for the response times observed for all time-limited requests.

CPU Time-Noncritical request count: the total number of time-noncritical requests generated.

CPU Time-Noncritical Response time stats aggregate statistics for the response times observed for all time-noncritical requests.

The next four statistics are not reported if "Print all interrupt stats?" is set to false (0).

CPU Number of IO interrupts: the total number of I/O interrupts received by all CPU's.

CPU Time spent in I/O interrupts: the total computation time spent on I/O interrupt handlers.

CPU Number of clock interrupts: the total number of I/O interrupts received by all CPU's.

CPU Time spent in clock interrupts: the total computation time spent on clock interrupt handlers.

The next four statistics are not reported if "Print sleep stats?" is set to false (0).

Number of sleep events: the total number of sleep events "executed" by all processes.

Number of I/O sleep events: the total number of sleep events "executed" in order to wait for I/O requests.

Average sleep time: the average length of time between a sleep event and the corresponding wake-up event.

Average I/O sleep time: the average length of time that between a sleep event that waits for an I/O request and the corresponding wake-up event (i.e., the I/O request's completion).

If there is more than one CPU, then per-CPU statistics are reported. The per-CPU statistics are the same as the aggregate CPU statistics described above. The per-CPU statistics are not reported if "Print per-CPU stats?" is set to false (0).

Process Total computation time: the total computation time of all processes (other than the idle processes) in the system.

Process Last event time: the simulated time of the last process event "executed".

Process Number of I/O requests: the total number of I/O requests generated.

Process Number of read requests: the total number of read I/O requests generated.

Process Number of C-switches: the total number of context switches to or from non-idle processes.

Process Number of sleeps: the total number of sleep events "executed."

Process Average sleep time: the average time between a process's sleep event and the corresponding wake-up event.

Process Number of I/O sleeps: the total number of sleep events "executed" in order to wait for I/O requests.

Process Average I/O sleep time: the average time between a process's sleep event that waits for an I/O request and the corresponding wake-up event (i.e., the I/O request's completion).

Process False idle time: the total amount of false idle time. This value can be greater than the total measured false idle time because more than one process can contribute to any given period of false idle time.

Process Read Time limits measured: the number of time limits observed for read I/O requests. (This value differs from the number of time-limited read requests if the simulation ends before one or more read I/O request time limits expire.)

Process Read Time limit duration stats: aggregate statistics for read I/O request time limits.

Process Write Time limits measured: the number of time limits measured for write I/O requests. (This value differs from the number of time-limited write requests if the simulation ends before one or more write I/O request time limits expire.)

Process Write Time limit duration stats: aggregate statistics for write I/O request time limits.

Process Read Time limits missed: the number of time limits missed by read I/O requests.

Process Missed Read Time limit duration stats: aggregate statistics for missed read I/O request time limits.

Process Write Time limits missed: the number of time limits missed by write I/O requests.

Process Missed Write Time limit duration stats: aggregate statistics for the missed write I/O request time limits.

If there is more than one simulated process, then per-process statistics are reported. The per-process statistics are the same as the aggregate process statistics described above. The per-process statistics are not reported if "Print per-process stats?" is set to false (0).

5.2.2 Validation trace statistics

The following statistics are reported only when an external validation trace (with a format of "validate") is used as the input workload.

VALIDATE Trace access time stats: aggregate statistics for the access times measured for the corresponding real storage subsystem. (The access time measured for each request is part of the input trace format.)

VALIDATE Trace access diff time stats: aggregate statistics for the per-request differences between the simulated and measured access times.

VALIDATE Trace write access diff time stats: aggregate statistics for the measured access times for write requests.

VALIDATE Trace write access diff time stats: aggregate statistics for the per-request differences between the simulated and measured access times for write requests.

The remaining statistics for validate workloads are historical in nature and are primarily useful for debugging DiskSim's behavior. The information needed to trigger them is not included in most of the validation traces.

VALIDATE double disconnects: the number of requests incurring two bus disconnects during the request's lifetime.

VALIDATE triple disconnects: the number of requests incurring three bus disconnects during the request's lifetime.

VALIDATE read buffer hits: the number of read requests that were serviced directly from the disk's on-board cache.

VALIDATE buffer misses: the number of requests that required actual magnetic media access.

5.2.3 HPL trace statistics

The following statistics are reported only when an external HPL trace (i.e., a trace in the HPLabs SRT format) is used as the input workload.

Total reads: the number of read requests in the trace, followed by the fraction of all requests that were reads.

Total writes: the number of write requests in the trace, followed by the fraction of all requests that were writes.

Sync Reads: the number of read requests marked (by a flag value) as synchronous, meaning that an application process will be blocked until the request completes. This value is followed by the fraction of all requests that were synchronous reads and the fraction of all read requests that were synchronous.

Sync Writes: the number of write requests marked (by a flag value) as synchronous, meaning that an application process will be blocked until the request completes. This value is followed by the fraction of all requests that were synchronous writes and the fraction of write requests that were synchronous.

Async Reads: the number of read requests not marked as synchronous, followed by the fraction of requests that were asynchronous reads and the fraction of all read requests that were asynchronous.

Async Writes: the number of write requests not marked as synchronous, followed by the fraction of all requests that were asynchronous writes and the fraction of all write requests that were asynchronous.

Mapped disk #X Trace queue time stats: aggregate statistics for the per-request queue times measured for disk X in the traced system.

Mapped disk #X Trace response time stats: aggregate statistics for the per-request response times (i.e., request arrival to request complete, including queue delays and service time) measured for disk X in the traced system.

Mapped disk #X Trace access time stats: aggregate statistics for the per-request access times (i.e., service times) measured for disk X in the traced system.

Mapped disk #X Trace queue length stats: aggregate statistics for the instantaneous queue lengths observed by each request for disk X in the traced system.

Mapped disk #X Trace non-queue time stats: aggregate statistics for the measured per-request times spent in the device driver for disk X in the traced system between request arrival and delivery to the storage controller when no other requests are pending. This provides insight into the device driver overheads for the traced systems.

5.2.4 System-level logical organization statistics

The following statistics are reported for each system-level logical organization. (Note: every DiskSim simulation involves at least one system-level logical organization, even if it simply maps each logical device onto an equivalent physical device.)

System logorg #X Number of requests: the number of requests submitted to the front-end of logical organization x.

System logorg #X Number of read requests: the number of read requests submitted to the front-end of logical organization X, followed by the fraction of all front-end requests that were reads.

System logorg #X Number of accesses: the number of accesses passed to the back-end of logical organization X. Striping, data redundancy, and other logical aspects can cause this value to differ from the number of front-end requests.

System logorg #X Number of read accesses: the number of accesses passed to the back-end of logical organization X in response to front-end read requests, followed by the fraction of all back-end requests that were reads.

System logorg #X Average outstanding: the average number of front-end requests in progress at any point in time to logical organization X.

System logorg #X Maximum outstanding: the maximum number of front-end requests in progress at any point in time to logical organization X.

System logorg #X Avg nonzero outstanding: the average number of front-end requests in progress at times when there was at least one outstanding request to logical organization X.

System logorg #X Completely idle time: the amount of time during which no front-end requests are outstanding to logical organization X.

System logorg #X Response time stats: aggregate statistics for the response times observed for all front-end requests to logical organization X.

System logorg #X Time-critical reads: the number of front-end read requests to logical organization X marked (by a flag field) as time-critical.

System logorg #X Time-critical write: the number of front-end write requests to logical organization X marked (by a flag field) as time-critical.

The next ten statistics are not reported if "Print driver locality stats?" is set to false (0).

System logorg #X Inter-request distance stats: aggregate statistics for the distances between the starting addresses of subsequent accesses to the same device in logical organization X.

System logorg #X Sequential reads: the number of back-end read accesses whose starting addresses were sequential to the immediately previous access to the same device in logical organization X, followed by the fraction of back-end accesses that were sequential reads and the fraction of back-end reads that were sequential.

System logorg #X Sequential writes: the number of back-end write accesses whose starting addresses were sequential to the immediately previous access to the same device in logical organization X, followed by the fraction of back-end accesses that were sequential writes and the fraction of back-end writes that were sequential.

System logorg #X Interleaved reads: the number of back-end read accesses whose starting addresses were almost sequential to (i.e., less than 16 sectors beyond the end of) the immediately previous access to logical organization X, followed by the fraction of back-end accesses that were "interleaved" reads and the fraction of back-end reads that were "interleaved."

System logorg #X Interleaved writes: the number of back-end write accesses whose starting addresses were almost sequential to (i.e., less than 16 sectors beyond the end of) the immediately previous access to logical organization X, followed by the fraction of back-end accesses that were "interleaved" writes and the fraction of back-end writes that were "interleaved."

System logorg #X Logical sequential reads: the number of front-end read requests whose starting addresses were logically sequential to the immediately previous request to logical organization X.

System logorg #X Logical sequential writes: the number of front-end write requests whose starting addresses were logically sequential to the immediately previous request to logical organization X.

System logorg #X Sequential disk switches: the number of back-end accesses generated for logically sequential front-end requests for logical organization X that (because of striping or some such) accessed a different device than the immediately previous request.

System logorg #X Logical local accesses: the number of front-end requests marked (by a flag) as logically "local" to the immediately previous request to logical organization X.

System logorg #X Local disk switches: the number of back-end accesses generated for front-end requests marked (by a flag) as logically "local" that (because of striping or some such) accessed a different device than the immediately previous request to logical organization X.

The next two statistics are not reported if "Print driver interfere stats?" is set to false (0).

System logorg #X Sequential step S: the number of back-end accesses to logical organization X that were sequential to the back-end access S+1 accesses prior, followed by the fraction of back-end accesses that fall into this category. These statistics are only reported if the fraction is greater than 0.002.

System logorg #X Local (D) step S: the number of back-end accesses to logical organization X whose starting addresses were D device sectors from the back-end access S+1 accesses prior, followed by the fraction of back-end accesses that fall into this category. These statistics are only reported if the fraction is greater than 0.002.

The next two statistics are not reported if "Print driver blocking stats?" is set to false (0).

System logorg #X Blocking factor: B: the number of back-end accesses to logical organization X whose size is an integer multiple of B sectors, followed by the fraction of back-end accesses that fall into this category. These statistics are only reported if the fraction is greater than 0.002.

System logorg #X Alignment factor: A: the number of back-end accesses to logical organization X whose starting address is an integer multiple of A sectors, followed by the fraction of back-end accesses that fall into this category. These statistics are only reported if the fraction is greater than 0.002.

The next three statistics are not reported if "Print driver intarr stats?" is set to false (0).

System logorg #X Inter-arrival time stats: aggregate statistics for the inter-arrival times of front-end requests to logical organization X.

System logorg #X Read inter-arrival time stats: aggregate statistics for the inter-arrival times of front-end read requests to logical organization X.

System logorg #X Write inter-arrival time stats: aggregate statistics for the inter-arrival times of front-end write requests to logical organization X.

The next two statistics are not reported if "Print driver streak stats?" is set to false (0).

System logorg #X Number of streaks: the number of sequences of back-end accesses to logical organization X addressed to the same device with no interleaved accesses to other devices.

System logorg #X Streak length stats: aggregate statistics for the lengths of sequences of back-end accesses to logical organization X addressed to the same device with no interleaved accesses to other devices.

The next three statistics are not reported if "Print driver stamp stats?" is set to false (0).

System logorg #X Timestamped # outstanding distribution: a distribution of the number of requests outstanding to logical organization X at regular simulated time intervals (specified by the "Time stamp interval" parameter).

System logorg #X Timestamped avg # outstanding difference distribution: a distribution of the average difference between the number of requests outstanding to each back-end device of logical organization X and the average number of requests outstanding per back-end device of logical organization X (measured at regular simulated time intervals).

System logorg #X Timestamped max # outstanding difference distribution: a distribution of the maximum difference between the number of requests outstanding to a particular back-end device of logical organization X and the average number of requests outstanding per back-end device of logical organization X (measured at regular simulated time intervals).

The next three statistics are not reported if "Print driver size stats?" is set to false (0).

System logorg #X Request size stats: aggregate statistics for the sizes of front-end requests to logical organization X.

System logorg #X Read request size stats: aggregate statistics for the sizes of front-end read requests to logical organization X.

System logorg #X Write request size stats: aggregate statistics for the sizes of front-end write requests to logical organization X.

The next two statistics are not reported if "Print driver idle stats?" is set to false (0).

System logorg #X Number of idle periods: the number of time periods during which no requests were outstanding to logical organization X.

System logorg #X Idle period length stats: aggregate statistics for the durations of time periods during which no requests were outstanding to logical organization X.

The remaining system-level logorg statistics are aggregate statistics over the set of disks in the logical organization. The statistics reported are the same as those described in section 5.2.6 under "Disk statistics".

5.2.5 I/O driver statistics

All of the I/O driver statistics are generated by the request queue module (which is also used by the disk and controller modules). None of them are reported if "Print driver queue stats?" is set to false (0).

IOdriver Total Requests handled: the number of requests completed from the driver's point of view.

IOdriver Requests per second: the number of requests completed per second of simulated time.

IOdriver Completely idle time: the total amount of time that no requests were outstanding.

IOdriver Response time stats: aggregate statistics for request response times.

IOdriver Overlaps combined: the number of requests made unnecessary because they completely overlap with another outstanding request, followed by the fraction of requests that fall into this category. (Note that this is an extremely unusual event in real systems, but the situation may arise frequently in trace-driven simulation [Ganger95].)

IOdriver Read overlaps combined: the number of read requests made unnecessary because they completely overlap with another outstanding request, followed by the fraction of requests that fall into this category.

The next eight statistics are not reported if "Print driver crit stats?" is set to false (0).

IOdriver Critical Reads: the number of read requests marked (by a flag) as time-critical, followed by the fraction of requests that are time-critical reads.

IOdriver Critical Read Response time stats: aggregate statistics for the response times of read requests marked time-critical.

IOdriver Non-Critical Reads: the number of read requests not marked (by a flag) as time-critical, followed by the fraction of requests that are reads not marked time-critical.

IOdriver Non-Critical Read Response time stats: aggregate statistics for the response times of read requests not marked time-critical.

IOdriver Critical Writes: the number of write requests marked (by a flag) as time-critical, followed by the fraction of requests that are time-critical writes.

IOdriver Critical Write Response time stats: aggregate statistics for the response times of write requests marked time-critical.

IOdriver Non-Critical Writes: the number of write requests not marked (by a flag) as time-critical, followed by the fraction of requests that are writes not marked time-critical.

IOdriver Non-Critical Write Response time stats: aggregate statistics for the response times of write requests not marked time-critical.

IOdriver Number of reads: the number of read requests, followed by the fraction of requests that are reads.

IOdriver Number of writes: the number of write requests, followed by the fraction of requests that are writes.

IOdriver Sequential reads: the number of read requests whose starting addresses are sequential to the immediately previous request to the same device, followed by the fraction of requests that are sequential reads.

IOdriver Sequential writes: the number of write requests whose starting addresses are sequential to the immediately previous request to the same device, followed by the fraction of requests that are sequential writes.

The next twelve statistics are not reported if "Print driver queue stats?" is set to false (0).

IOdriver Average # requests: the average number of requests outstanding (in queues or in service).

IOdriver Maximum # requests: the maximum number of requests outstanding.

IOdriver end # requests: the number of requests outstanding when the simulation ended.

IOdriver Average queue length: the average length of the request queue.

IOdriver Maximum queue length: the maximum length of the request queue.

IOdriver End queued requests: the length of the request queue when the simulation ended.

IOdriver Queue time stats: aggregate statistics for the queue times incurred by requests.

IOdriver Avg # read requests: the average number of read requests outstanding.

IOdriver Max # read requests: the maximum number of read requests outstanding.

IOdriver Avg # write requests: the average number of write requests outstanding.

IOdriver Max # write requests: the maximum number of write requests outstanding.

IOdriver Physical access time stats: aggregate statistics for the request access times (i.e., excluding any queueing times).

The next three statistics are not reported if "Print driver intarr stats?" is set to false (0).

IOdriver Inter-arrival time stats: aggregate statistics for request inter-arrival times.

IOdriver Read inter-arrival time stats: aggregate statistics for read request inter-arrival times.

IOdriver Write inter-arrival time stats: aggregate statistics for write request inter-arrival times.

The next two statistics are not reported if "Print driver idle stats?" is set to false (0).

IOdriver Number of idle periods: the number of time periods during which no requests were outstanding.

IOdriver Idle period length stats aggregate statistics for the durations of time periods during which no requests were outstanding.

The next three statistics are not reported if "Print driver size stats?" is set to false (0).

IOdriver Request size stats: aggregate statistics for the sizes of requests.

IOdriver Read request size stats: aggregate statistics for the sizes of read requests.

IOdriver Write request size stats: aggregate statistics for the sizes of write requests.

IOdriver Instantaneous queue length stats: aggregate statistics for the queue lengths observed at the points in time when each new request arrived.

IOdriver Sub-optimal mapping penalty stats: aggregate statistics for the seek distance penalties incurred due to the use of inaccurate mapping information in translating request starting locations to cylinder/track/sector locations (by a scheduler).

Some of the disk request scheduling algorithms supported by DiskSim employ more than one sub-queue (e.g., for request prioritization). If this is the case, then several of the above statistics (from "IOdriver Response time stats" to "IOdriver Physical access time stats") are reported for each of the sub-queues in addition to the above aggregate values. Also, depending upon which sub-queues are employed, up to four additional statistics may be reported:

IOdriver Requests switched to timeout queue: the number of requests that were switched to the higher priority *timeout* queue (as described in section 3.2.5), because they were queued in the *base* queue for longer than their specified timeout time.

IOdriver Timed out requests: the number of requests that did not complete within their timeout value. Such requests are only switched to the *timeout* queue if they have not yet been initiated.

IOdriver Half-way timed out requests: the number of requests that did not complete within half of their timeout value. One of the supported scheduler options gives such requests an intermediate priority level for the remainder of their timeout period.

IOdriver Requests switched to priority queue: the number of requests that were switched to the high-priority *timeout* queue because of information delivered from higher-level system components (e.g., the process scheduler) after the request was queued. One reason for such a switch might be that a process must wait for the request to complete [Ganger93]; if a high-priority process is waiting on the request completion, the request's priority may be increased at the I/O driver.

If there is more than one device (or more than one driver), then separate per-driver-per-device statistics are reported. The statistics reported are the same as those described above (as aggregate "IOdriver" statistics). The per-driver-per-device statistics are not reported if "Print driver per-device stats?" is set to false (0).

5.2.6 Disk statistics

The first set of disk statistics is generated by the request queue module. The specific statistics reported are the same as the "IOdriver" statistics described in section 5.2.5, except that they apply to each disk's individual request queue(s) (and are denoted accordingly). The "Print ... stats?" parameters for the queue statistics are the same as for the corresponding driver parameters with the word, "driver", replaced by "disk".

The next three statistics are not reported if "Print disk seek stats?" is set to false (0).

Disk Seeks of zero distance: the number of requests resulting in media accesses that require no "seek" (i.e., movement of the disk's read/write head from one cylinder to another), followed by the fraction of all requests requiring no seek.

Disk Seek distance stats: aggregate statistics for the seek distances observed for requests requiring media access.

Disk Seek time stats: aggregate statistics for the seek times observed for requests requiring media access.

The next three statistics are not reported if "Print disk latency stats?" is set to false (0).

Disk Full rotation time: the amount of time required for the disk platters to complete a full revolution. This statistic is only reported for single-disk configurations or in sets of per-disk statistics (see below).

Disk Zero rotate latency: the number of media accesses that incur no rotational latency, followed by the fraction of all media accesses incurring no rotational latency.

Disk Rotational latency stats: aggregate statistics for the rotational latencies for requests requiring media access.

The next statistic is not reported if "Print disk xfer stats?" is set to false (0).

Disk Transfer time stats: aggregate statistics for the media transfer times for requests requiring media access.

The next statistic is not reported if "Print disk acctime stats?" is set to false (0).

Disk Access time stats: aggregate statistics for media access times for requests requiring media access.

The next two statistics are not reported if “Print disk interfere stats?” is set to false (0).

Disk Sequential interference: the number of requests marked (by a flag) as logically sequential that were not temporally and physically sequential due to interference with other request streams or data mapping algorithms (e.g., striping).

Disk Local interference: the number of requests marked (by a flag) as logically “local” that were not temporally or physically local due to interference with other request streams or data mapping algorithms (e.g., striping).

The next seventeen statistics are not reported if “Print disk buffer stats?” is set to false (0).

Disk Number of buffer accesses: the number of requests that check the disk’s on-board cache for specific contents.

Disk Buffer hit ratio: the number of requests that check the disk’s on-board cache and find some “usable” data, followed by the fraction of all requests that check the disk’s on-board cache and find some “usable” data. For example, a read request whose first sector of requested data is in the cache (or is currently being read into the cache) would fall into this category. Also, a write request may fall into this cateogory if the on-board controller allows its data to be appended or prepended to an existing quantity of “dirty” data. In the latter case, the existing dirty data is “usable” because the new request may be combined with it (i.e., is logically sequential to it).

Disk Buffer miss ratio: the number of requests that check the disk’s on-board cache and do not find any “usable” data, followed by the fraction of all requests that check the disk’s on-board cache and do not find some “usable” data. For example, a read request whose first sector of requested data is not in the cache (and is not currently being read into the cache) would certainly fall into this category. Also, a write request would fall into this cateogory if the request’s data cannot be combined with any existing “dirty” data in the cache.

Disk Buffer read hit ratio: the number of read requests that check the disk’s on-board cache and find all of the requested data already present, followed by the fraction of all read requests and the fraction of all requests that fall into this category.

Disk Buffer prepend hit ratio: the number of all write requests that check the disk’s on-board cache and are combined with existing write requests where the new request’s data are logically prepended to the existing “dirty” data, followed by the fraction of all write requests that fall into this category.

Disk Buffer append hit ratio: the number of all write requests that check the disk’s on-board cache and are combined with existing write requests where the new request’s data are logically appended to the existing “dirty” data, followed by the fraction of all write requests that fall into this category.

Disk Write combinations: the number of all write requests that check the disk’s on-board cache and are combined with existing write requests (either logically prepended or appended), followed by the fraction of all write requests that are combined with existing write requests.

Disk Ongoing read-ahead hit ratio: the number of all read requests that check the disk’s on-board cache and find an initial portion of the requested data already present and additional data being actively prefetched into the same cache segment, followed by the fraction of all read requests and the fraction of all requests that fall into this category.

Disk Average read-ahead hit size: the average amount of requested data found in the cache for read requests that check the disk's on-board cache and find an initial portion of the requested data already present and additional data being actively prefetched into the same cache segment.

Disk Average remaining read-ahead: the average amount of requested data remaining to be fetched into the cache for read requests that check the disk's on-board cache and find an initial portion of the requested data already present and additional data being actively prefetched into the same cache segment.

Disk Partial read hit ratio: the number of read requests that check the disk's on-board cache and find an initial portion of the requested data already present (with no ongoing prefetch), followed by the fraction of all read requests and the fraction of all requests that fall into this category.

Disk Average partial hit size: the average amount of requested data found in the cache for read requests that check the disk's on-board cache and find an initial portion of the requested data (with no ongoing prefetch).

Disk Average remaining partial: the average amount of requested data remaining to be fetched into the cache for read requests that check the disk's on-board cache and find an initial portion of the requested data (with no ongoing prefetch).

Disk Total disk bus wait time: the total amount of time spent waiting for access to a bus (i.e., arbitration delay).

Disk Number of disk bus waits: the total number of times a delay occured when attempting to access the bus (i.e., the bus was "owned" by another entity when access was requested).

Per-disk statistics are reported for multi-disk configurations. Statistics for specific disks can be enabled or disabled by setting the corresponding "Print stats for disk" configuration parameter (see section 3.2.4) true (1) or false (0).

5.2.7 Controller statistics

No statistics are reported for the two simple controller models. The following statistics are reported only for "CTLR_SMART" controllers, which include a cache and are capable of queueing/scheduling requests for one or more attached storage devices. All of the cache statistics are reported for individual controllers only (i.e., no aggregates across controllers are reported). The controller cache statistics are not reported if "Print controller cache stats?" is set to false (0).

Controller #X cache requests: the number of requests serviced by the cache of controller X.

Controller #X cache read requests: the number of read requests serviced by the cache of controller X, followed by the fraction of serviced requests that are reads.

Controller #X cache atoms read: the number of cache atoms accessed by read requests to controller X, followed by the fraction of cache atom accesses that are reads. A "cache atom" is the minimal unit of cache access. In the current version of DiskSim, the cache atom size is always equal to the sector size of the underlying storage devices.

Controller #X cache read misses: the number of cache read requests to controller X for which no useful data are found in the cache, followed by the fraction of all requests that are cache read misses and the fraction of all read requests that are misses.

Controller #X cache read full hits: the number of cache read requests to controller X for which all necessary data are found in the cache, followed by the fraction of all requests that are cache read full hits and the fraction of all read requests that are full hits.

Controller #X cache fills (read): the number of cache fill accesses issued to the underlying storage devices by controller X, followed by the fraction of requests that require a cache fill and the fraction of read requests that require a cache fill.

Controller #X cache atom fills (read): the number of atoms read by cache fill accesses issued to the underlying storage devices by controller X, followed by the fraction of cache atom accesses that require a cache fill and the fraction of cache atom read accesses that require a cache fill.

Controller #X cache write requests: the number of write requests serviced by the cache of controller X, followed by the fraction of requests that are writes.

Controller #X cache atoms written: the number of cache atoms written by write requests to controller X, followed by the fraction of cache atom accesses that are writes.

Controller #X cache write misses: the number of cache write requests to controller X that do not overlap at all with data found in the cache, followed by the fraction of all requests that are cache write misses and the fraction of write requests that are misses.

Controller #X cache write hits (clean): the number of cache write requests to controller X that overlap only with clean data found in the cache, followed by the fraction of all requests that are clean cache write hits and the fraction of write requests that are clean hits.

Controller #X cache write hits (dirty): the number of cache write requests to controller X that overlap with some amount of dirty data found in the cache, followed by the fraction of all requests that are dirty cache write hits and the fraction of write requests that are dirty hits.

Controller #X cache fills (write): the number of cache fill accesses (i.e., installation reads [Otoole94]) that were required to complete write requests to controller X, followed by the fraction of all requests that require an installation read and the fraction of all write requests that require an installation read.

Controller #X cache atom fills (write): the number of atoms read into the cache in order to complete write requests to controller X, followed by the fraction of all cache atom accesses requiring an installation read and the fraction of all cache atom writes requiring an installation read.

Controller #X cache destages (write): the number of destage accesses (i.e., write-backs) initiated by controller X, followed by the fraction of all requests that (eventually) generated a destage access and the fraction of all write requests that generated a destage access.

Controller #X cache atom destages (write): the number of atoms written back from the cache of controller X to the storage devices, followed by the fraction of all atom accesses generating (eventually) a destage access and the fraction of all atom write accesses generating a destage access.

Controller #X cache end dirty atoms: the number of dirty atoms left in the cache of controller X at the end of the simulation, followed by the fraction of all cache atom accesses that remain dirty at the end of the simulation.

In addition to the per-controller cache statistics, a set of per-controller aggregate queue statistics are generated by the request queue module. That is, queue statistics are reported for each individual controller across all storage devices attached to that controller. The specific statistics reported are the same as the "IOdriver ..." statistics described in section 5.2.5, except that they apply to each controller's back-end, per-device request queues (and are denoted accordingly). The "Print ... stats?" parameters for the queue statistics are the same as for the corresponding driver parameters with the word, "driver", replaced by "controller".

If there are multiple devices attached to a controller, then the corresponding per-device queue statistics are also reported for each device (i.e., in addition to the aggregate statistics described above). The per-device statistics will not be reported if "Print controller per-device stats?" is set to false (0).

Total controller bus wait time: the total amount of time spent by all controllers waiting for access to a bus (i.e., arbitration delay).

5.2.8 Bus statistics

No aggregate statistics (across sets of buses) are reported.

Bus #X Total utilization time: the amount of time (in milliseconds) that the bus was not idle during the simulation run. Utilization as a fraction of total simulation time is also reported on this line.

The following set of statistics are not reported if "Print bus idle stats?" is set to false (0).

Bus #X Idle period length stats: aggregate statistics for the lengths of idle periods (i.e., periods during which the bus was unused) observed for bus X.

The remaining statistics are not reported if "Print bus arbwait stats?" is set to false (0).

Bus #X Number of arbitrations: the number of arbitration decisions made for bus X, including those that involved only a single requester.

Bus #X Arbitration wait time stats: aggregate statistics for bus X acquisition delays experienced by attached components. Such delays include both the bus arbitration overhead and any wait time experienced while other components finish their bus transfers.

Parameter	HP C2247A	Seagate ST41601N	DEC RZ26	HP C2490A	HP C3323A
Formatted Capacity	1.05 GB	1.37 GB	1.03 GB	2.13 GB	1.05 GB
RPM	5400	5400	5400	6400	5400
Diameter	$3\,{}^1\!/\!_2''$	$5\,{}^1\!/\!_4''$	$3\,{}^1\!/\!_2''$	$3\,{}^1\!/\!_2''$	$3\,{}^1\!/\!_2''$
Height	$1.63''$	$3.25''$	$1.63''$	$1.63''$	$1.00''$
Data Surfaces	13	17	14	18	7
Cylinders	2051	2098	2570	2582	2910
Zones	8	14	1	11	8
Sectors/Track	56–96	61–85	58	68–108	72-120

Table 1: Basic disk drive parameters.

6 Validation

The disk module of the storage subsystem simulator has been validated by exercising five disk drives (representing three different disk manufacturers; see table 1) and capturing traces of the resulting I/O activity. Using the observed inter-request delays, each traced request stream was also run through the simulator, which was configured to emulate the corresponding real subsystem. For each disk, this process was repeated for several synthetic workloads with varying read/write ratios, arrival rates, request sizes and degrees of sequentiality and locality. The measured and simulated response time averages match to within 0.8% for all validation runs. (The bus, controller and device driver modules have also been validated as part of a more comprehensive, system-level simulation environment [Ganger95a].)

Greater insight into the validity of a storage subsystem model can be gained by comparing measured and simulated response time distributions [Ruemmler94]. Figures 1 and 2 show distributions of measured and simulated response times for a sample validation workload of 10,000 requests. 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 figure for each of the curves is given in the corresponding caption. The worst-case demerit figure observed over all validation runs was only 2.0% of the corresponding average response time. To our knowledge, no previous disk drive simulator has achieved this level of accuracy.

To accurately mimic the performance behavior of a disk drive, the parameter values used to configure the simulator must accurately reflect the behavior of the actual device. The extremely close match shown in figure 1 was realized by measuring parameter values directly with a logic analyzer attached to the SCSI bus. The configuration values for the other four disks were obtained with an automatic (software) extraction tool (described in [Worthington95]). While still accurate and much less time-consuming, these values are not quite as precise as those obtained with the logic analyzer. We have further observed that using the few, combined values generally present in disk drive specifications yields much larger discrepancies between simulated and observed performance.

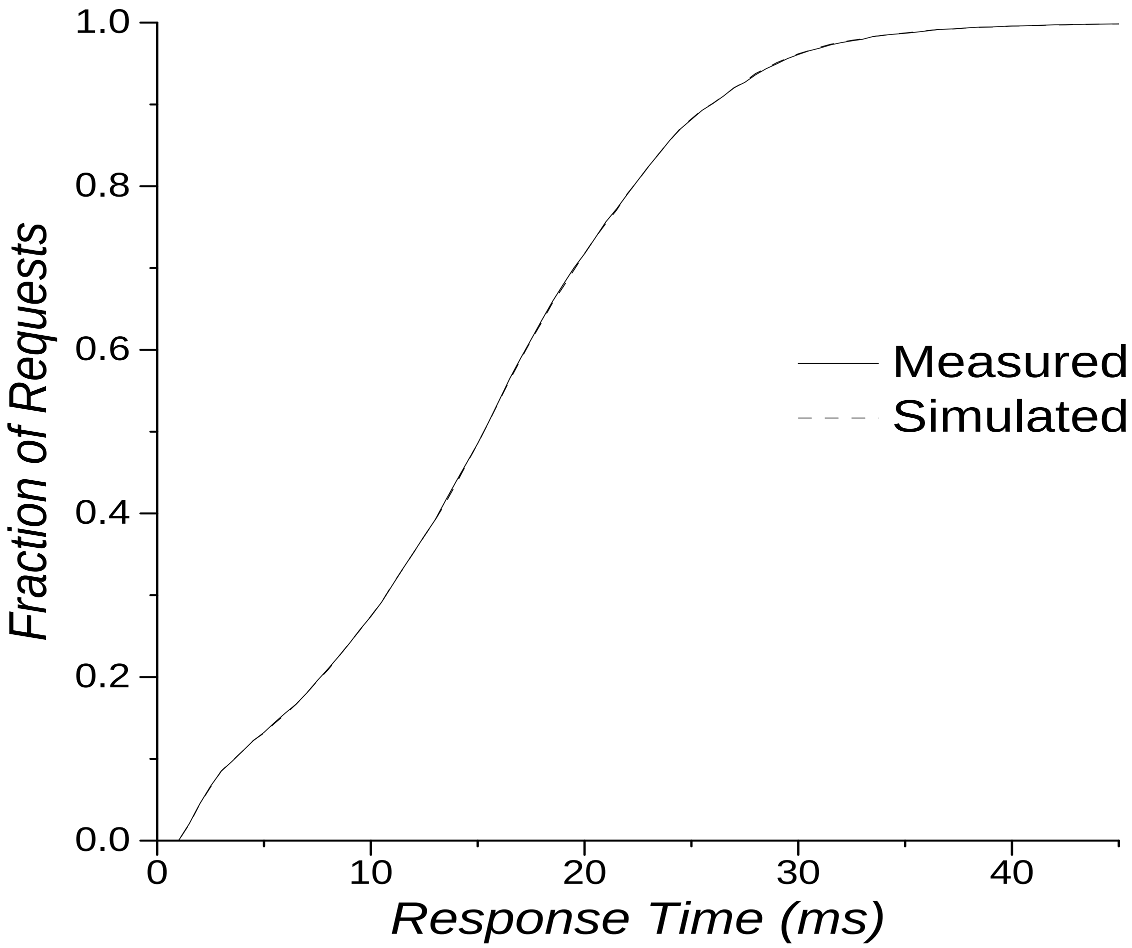
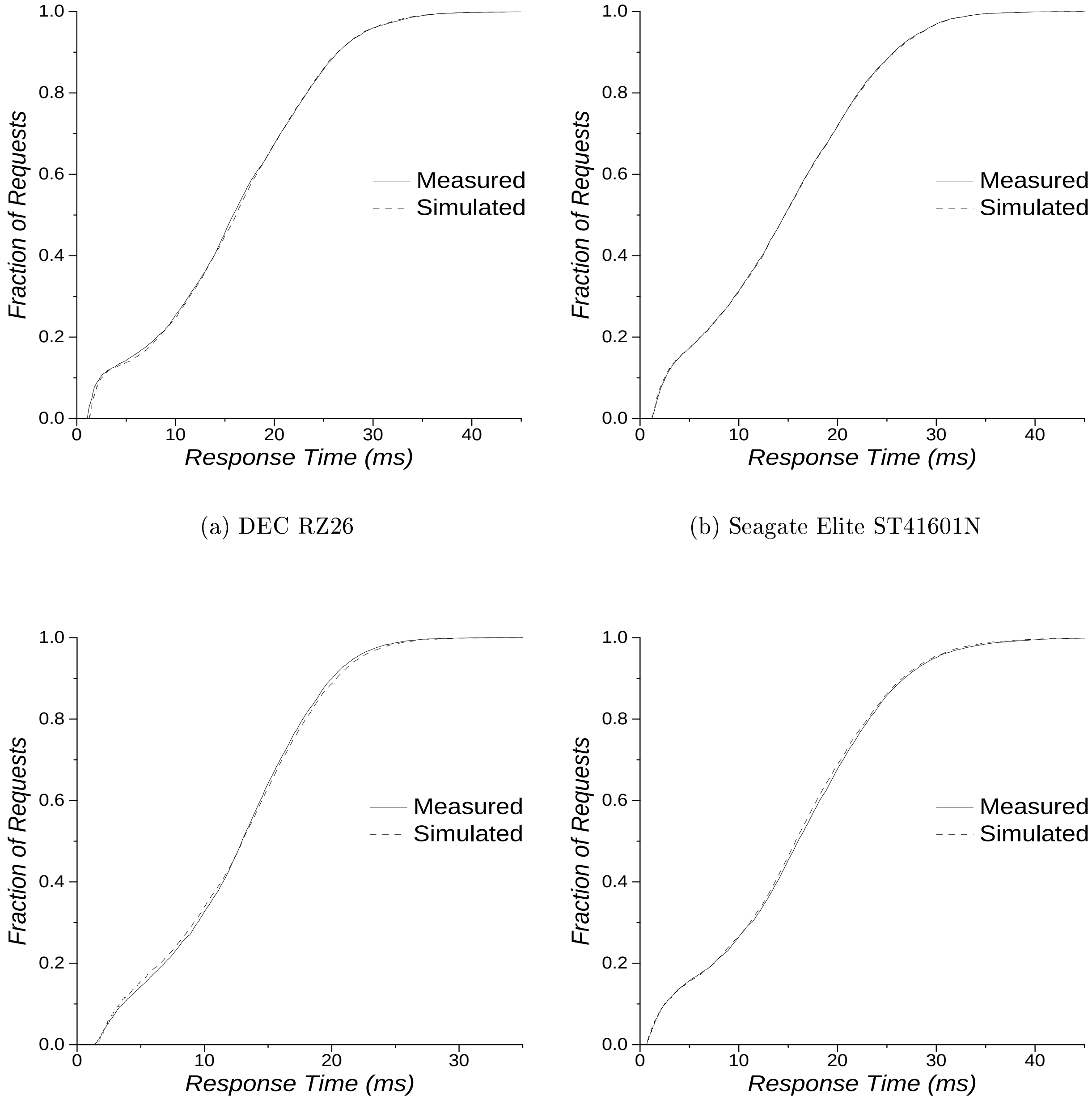


Figure 1: Measured and Simulated Response Time Distributions for an HP C2247A Disk Drive. The demerit figure for this validation run is 0.07 ms, or 0.5% of the corresponding mean response time. Characteristics of the HP C2247A can be found in table 1 and in [HP92, Worthington94]. The validation workload parameters are 50% reads, 30% sequential, 30% local [normal with 10000 sector variance], 8KB mean request size [exponential], interarrival time [uniform 0–22 ms].



(a) DEC RZ26

(b) Seagate Elite ST41601N

(c) HP C2490A

(d) HP C3323A

Figure 2: Measured and Simulated Response Time Distributions for 4 Disk Drives. The demerit figures for these validation runs are 0.19 ms, 0.075 ms, 0.26 ms and 0.32 ms, respectively (or 1.2%, 0.5%, 2.0% and 1.9% of the corresponding mean response times). drives can be found in table 1 and in [Seagate92, Seagate92a, HP93, HP94, Worthington96]. The validation workload parameters are 50% reads, 30% sequential, 30% local [normal with 10000 sector variance], 8KB mean request size [exponential], interarrival time [uniform 0–22 ms].

A Copyright notice for DiskSim

References

[Ganger93] G. Ganger, Y. Patt, "The Process-Flow Model: Examining I/O Performance from the System's Point of View", *ACM SIGMETRICS Conference*, May 1993, pp. 86–97.

[Ganger93a] G. Ganger, B. Worthington, R. Hou, Y. Patt, "Disk Subsystem Load Balancing: Disk Striping vs. Conventional Data Placement", *Hawaii International Conference on System Sciences*, January 1993, pp. 40–49.

[Ganger94] G. Ganger, B. Worthington, R. Hou, Y. Patt, "Disk Arrays: High Performance, High Reliability Storage Subsystems", *IEEE Computer*, Vol. 27, No. 3, March 1994, pp. 30–36.

[Ganger95] G. Ganger, "System-Oriented Evaluation of Storage Subsystem Performance", Ph.D. Dissertation, CSE-TR-243-95, University of Michigan, Ann Arbor, June 1995.

[Ganger95a] G. Ganger, "Generating Representative Synthetic Workloads An Unsolved Problem", *Computer Measurement Group (CMG) Conference*, Decemeber 1995, pp. 1263–1269.

[Holland92] M. Holland, G. Gibson, "Parity Declustering for Continuous Operation in Redundant Disk Arrays", *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, October 1992, pp. 23–35.

[HP91] Hewlett-Packard Company, "HP C2247 3.5-inch SCSI-2 Disk Drive – Technical Reference Manual", Edition 1, Draft, December 1991.

[HP92] Hewlett-Packard Company, "HP C2244/45/46/47 3.5-inch SCSI-2 Disk Drive Technical Reference Manual", Part Number 5960-8346, Edition 3, September 1992.

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

[HP94] Hewlett-Packard Company, "HP C3323A 3.5-inch SCSI-2 Disk Drives, Technical Reference Manual", Part Number 5962-6452, Edition 2, April 1994.

[Karedla94] R. Karedla, J. S. Love, B. Wherry, "Caching Strategies to Improve Disk System Performance", *IEEE Computer*, Vol. 27, No. 3, March 1994, pp. 38–46.

[Lee91] E. Lee, R. Katz, "Peformance Consequences of Parity Placement in Disk Arrays", *ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 1991, pp. 190–199.

[Lee93] E. Lee, R. Katz, "An Analytic Performance Model of Disk Arrays", *ACM Sigmetrics Conference*, May 1993, pp. 98-109.

[NCR89] NCR Corporation, "NCR 53C700 SCSI I/O Processor Programmer's Guide", 1989.

[NCR90] NCR Corporation, "Using the 53C700 SCSI I/O Processor", SCSI Engineering Notes, No. 822, Rev. 2.5, Part No. 609-3400634, February 1990.

[NCR91] NCR Corporation, "Class 3433 and 3434 Technical Reference", Document No. D2-0344-A, May 1991.

[Otoole94] J. O'Toole, L. Shrira, "Opportunistic Log: Efficient Installation Reads in a Reliable Storage Server", *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, November 1994, pp. 39–48.

[Ousterhout85] J. Ousterhout, H. Da Costa, D. Harrison, J. Kunze, M. Kupfer, J. Thompson, "A Trace-Driven Analysis of the UNIX 4.2 BSD File System", *ACM Symposium on Operating System Principles*, 1985, pp. 15–24.

[Rosenblum95] M. Rosenblum, S. Herrod, E. Witchel, A. Gupta, "Complete Computer Simulation: The SimOS Approach", *IEEE Journal of Parallel and Distributed Technology*, Winter 1995, pp. 34-43.

[Ruemmler93] C. Ruemmler, J. Wilkes, "UNIX Disk Access Patterns", *Winter USENIX Conference*, January 1993, pp. 405–420.

[Ruemmler94] C. Ruemmler, J. Wilkes, "An Introduction to Disk Drive Modeling", *IEEE Computer*, Vol. 27, No. 3, March 1994, pp. 17–28.

[Satya86] M. Satyanarayanan, *Modeling Storage Systems*, UMI Research Press, Ann Arbor, MI, 1986.

[Seagate92] Seagate Technology, Inc., "SCSI Interface Specification, Small Computer System Interface (SCSI), Elite Product Family", Document Number 64721702, Revision D, March 1992.

[Seagate92a] eagate Technology, Inc., "Seagate Product Specification, ST41600N and ST41601N Elite Disc Drive, SCSI Interface", Document Number 64403103, Revision G,

[Thekkath94] C. Thekkath, J. Wilkes, E. Lazowska, "Techniques for File System Simulation", *Software – Practice and Experience*, Vol. 24, No. 11, November 1994, pp. 981–999.

[Worthington94] B. Worthington, G. Ganger, Y. Patt, "Scheduling Algorithms for Modern Disk Drives", *ACM SIGMETRICS Conference*, May 1994, pp. 241–251.

[Worthington95] B. Worthington, G. Ganger, Y. Patt, J. Wilkes, "On-Line Extraction of SCSI Disk Drive Parameters", *ACM SIGMETRICS Conference*, May 1995, pp. 146–156.

[Worthington95a] B. Worthington, "Aggressive Centralized and Distributed Scheduling of Disk Requests", Ph.D. Dissertation, CSE-TR-244-95, University of Michigan, Ann Arbor, June 1995.

[Worthington96] B. Worthington, G. Ganger, Y. Patt, J. Wilkes, "On-Line Extraction of SCSI Disk Drive Parameters", Technical Report, University of Michigan, Ann Arbor, 1996, in progress.