Diskmodel

March 2, 2015

1 Overview

1.1 Introduction

Diskmodel is a library implementing mechanical and layout models of modern magnetic disk drives. Diskmodel models two major aspects of disk operation. The layout module models logical-to-physical mapping of blocks, defect management and also computes angular offsets of blocks. The mechanical model handles seek times, rotational latency and various other aspects of disk mechanics.

The implementations of these modules in the current version of Diskmodel are derived from DiskSim 2.0 [Ganger99]. Disksim 3.0 uses Diskmodel natively. Diskmodel has also been used in a device driver implementation of a shortest positioning time first disk request scheduler.

1.2 Types and Units

All math in diskmodel is performed using integer arithmetic. Angles identified as points on a circle divided into discrete units. Time is represented as multiples of some very small time base. Diskmodel exports the types dm_time_t and dm_angle_t to represent these quantities. Diskmodel exports functions dm_time_itod, dm_time_dtoi (likewise for angles) for converting between doubles and the native format. The time function converts to and from milliseconds; the angle function converts to and from a fraction of a circle. dm_time_t and dm_angle_t should be regarded as opaque and may change over time. Diskmodel is sector-size agnostic in that it assumes that sectors are some fixed size but does not make any assumption about what that size is.

1.2.1 Three Zero Angles

When considering the angular offset of a sector on a track, there are at least three plausible candidates for a "zero" angle. The first is "absolute" zero which is the same on every track on the disk. For various reasons, this zero may not coincide with a sector boundary on a track. This motivates the second 0 which we will refer to as 0_t (t for "track") which is the angular offset of the first sector boundary past 0 on a track. Because of skews and defects, the lowest lbn on the track may not lie at 0_t . We call the angle of the lowest sector on the track 0_t (1 for "logical" or "lbn").

1.2.2 Two Zero Sectors

Similarly, when numbering the sectors on a track, it is reasonable to call either the sector at 0_t or the one at 0_t "sector 0." 0_t corresponds to directly to the physical location of sectors on a

track whereas 0_l corresponds to logical layout. Diskmodel works in both systems and the following function descriptions identify which numbering a given function uses.

1.2.3 Example

Consider a disk with 100 sectors per track, 2 heads, a head switch skew of 10 sectors and a cylinder switch skew of 20 sectors. (x, y, z) denotes cylinder x, head y and sector z.

LBN	0_l PBN	0_t PBN
0	(0,0,0)	(0,0,0)
	:	
99	(0,0,99)	(0,0,99)
100	(0,1,0)	(0,1,10)
101	(0,1,1)	(0,1,11)
	:	
189	(0,1,89)	(0,1,99)
190	(0,1,90)	(0,1,0)
191	(0,1,91)	(0,1,1)
199	(0,1,99)	(0,1,9)

Note that a sector is 3.6 degrees wide.

Cylinder	Head	0_l angle
0	0	0 degrees
0	1	36 degrees
1	0	72 degrees
1	1	108 degrees
2	0	180 degrees

1.3 API

This section describes the data structures and functions that comprise the Diskmodel API.

The dm_disk_if struct is the "top-level" handle for a disk in diskmodel. It contains a few disk-wide parameters – number of heads/surfaces, cylinders and number of logical blocks exported by device – along with pointers to the mechanics and layout interfaces.

1.3.1 Disk-wide Parameters

The top-level of a disk model is the dm_disk_if struct:

};

All fields of diskmodel API structures are read-only; the behavior of diskmodel after any of them is modified is undefined. layout and mech are pointers to the layout and mechanical module interfaces, respectively. Each is a structure containing a number of pointers to functions which constitute the actual implementation. In the following presentation, we write the functions as declarations rather than as types of function pointers for readability. Many of the methods take one or more result parameters; i.e. pointers whose addresses will be filled in with some result. Unless otherwise specified, passing NULL for result parameters is allowed and the result will not be filled in.

1.3.2 Layout

The layout interface uses the following auxiliary type:

dm_ptol_result_t appears in situations where a client code provides a pbn which may not exist on disk as-described e.g. due to defects. It contains the following values:

DM_SLIPPED
DM_REMAPPED
DM_OK
DM_NX

DM_SLIPPED indicates that the pbn is a slipped defect. DM_REMAPPED indicates that the pbn is a remapped defect. DM_OK indicates that the pbn exists on disk as-is. DM_NX indicates that there is no sector on the device corresponding to the given pbn. When interpreted as integers, these values are all less than zero so they can be unambiguously intermixed with nonnegative integers e.g. lbns.

The layout module exports the following methods:

Translate a logical block number (lbn) to a physical block number (pbn). remapsector is a result parameter which will be set to a non-zero value if the lbn was remapped.

The sector number in the result is relative to the 0_l zero sector.

Same as dm_translate_ltop except that the sector in result is relative to the 0_t sector.

Translate a pbn to an lbn. remapsector is a result parameter which will be set to a non-zero value if the pbn is defective and remapped.

The sector number in the operand is relative to the 0_l zero sector.

Same as dm_translate_ptol except that the sector in the result is relative to the 0_t sector.

Returns the number of sectors on the track containing the given lbn.

Returns the number of physical sectors on the track containing the given pbn. This may not be the same as the number of lbns mapped on this track. If the cylinder is unmapped, the return value will be the number of sectors per track for the nearest (lower) zone.

Computes lbn boundaries for the track containing the given pbn. first_lbn is a result parameter which returns the first lbn on the track containing the given pbn; similarly, last_lbn returns the last lbn on the given track. remapsector returns a non-zero value if the first or last block on the track are remapped. Note that last_lbn - first_lbn + 1 may be greater than the number of LBNs mapped on the track e.g. due to remapped defects.

Computes the seek distance in cylinders that would be incurred for given request. Returns a dm_ptol_result_t since one or both of the LBNs may be slipped or remapped.

This computes the starting offset of a pbn relative to 0. The operand is a pbn relative to 0_l ; the result is an angle relative to 0. This accounts for all skews, slips, etc.

The return value is 0_l for the track identified by the second argument. This is equivalent to calling dm_pbn_skew for sector 0 on the same track.

Finds the pbn of the sector whose leading edge is less than or equal to the given angle. Returns a ptol_result_t since the provided angle could be in slipped space, etc. Both the angle in the second operand and the sector number in the result pbn are relative to 0_l .

Returns the angular width of an extent of num sectors on the given track. Returns 0 if num is greater than the number of sectors on the track.

```
dm_angle_t dm_lbn_offset(struct dm_disk_if *, int lbn1, int lbn2);
```

Computes the angular distance/offset between two logical blocks.

```
int dm_marshalled_len(struct dm_disk_if *);
```

Returns the size of the structure in bytes when marshalled.

```
void *dm_marshall(struct dm_disk_if *, char *);
```

Marshall this layout struct into the provided buffer. The return value is the first address in the buffer not written.

1.3.3 Mechanics

The following diagram shows the breakdown of a zero-latency access in our model, and the corresponding definitions of seek time, positioning time and access time.

Computes the amount of time to seek from the first track to the second track, possibly including a head switch and additional write settling time. This is only track-to-track so the angles in the parameters are ignored. read should be nonzero if the access on the destination track is a read and zero if it is a write; extra write-settle time is included in the result for writes.

From the given inital condition and access, it will return the first block on the track to be read. The access is for len sectors starting at physical sector start on the same track as initial. immed indicates if this is an "immediate" or "zero-latency" access; if immed is zero, the result will always be the same as start.

This computes the rotational latency incurred from accessing up to len blocks from the track starting from angle initial and sector start. This will access to the end of the track but not wrap around; e.g. for a sequential access that starts on the given track and switches to another, after reaching the end of the first. The return value is the initial rotational latency; i.e. how long before the media transfer for the first block to be read starts. addtolatency is a result parameter returning additional rotational latency as defined in the figure above. Note that for non-zero-latency accesses, addtolatency will always be zero. Also note that for zero latency accesses, the latency is the amount of time before the media transfer begins for the first sector i.e. the same sector that would be returned by dm_access_block().

 ${\tt dm_pos_time}$ and ${\tt dm_acctime}$ optionally return broken-down components of the result via the following struct:

```
struct dm_mech_acctimes {
   dm_time_t seektime;
   dm_time_t initial_latency;
   dm_time_t initial_xfer;
   dm_time_t addl_latency;
   dm_time_t addl_xfer;
};
```

For a zero-latency access, the last two fields will always be zero. dm_pos_time only fills in the first two fields; dm_acctime fills in all 5.

```
dm_time_t dm_pos_time(struct dm_disk_if *,
```

```
struct dm_mech_state *initial,
struct dm_pbn *start,
int len,
int rw,
int immed);
```

Compute the amount of time before the media transfer for len sectors starting at start begins starting with the disk mechanics in state initial. 0 for rw indicates a write, any other value indicates a read. A non-zero value for immed indicates a "zero-latency" access. Positioning time is the same as seek time (including head-switch time and any extra write-settle time) plus initial rotational latency.

len must be at least 1.

Estimate how long it will take to access len sectors starting with pbn start with the disk initially in state initial. 0 for rw indicates a write; any other value indicates a read. A non-zero value for immed indicates a "zero-latency" access. result_state is a result parameter which returns the mechanical state of the disk when the access completes.

len must be at least 1.

Access time consists of positioning time (above), transfer time and any additional rotational latency not included in the positioning time, e.g. in the middle of a zero-latency access transfer.

dm_acctime ignores defects so it yields a smaller-than-correct result when computing access times on tracks with defective sectors. This is deliberate as the handling of defects is a high-level controller function which varies widely.

Compute how long it will take the disk to rotate from the angle in the first position to that in the second position.

Computes the amount of time to transfer len sectors to or from the track designated by the second argument. This is computed in terms of dm_get_sector_width() and dm_rottime() in the obvious way.

Returns the amount of time to swith from using the first head to the second.

Returns the angle of the media after time has elapsed assuming the media started at angle 0.

```
dm_time_t dm_period(struct dm_disk_if *);
```

Returns the rotational period of the media.

```
int dm_marshalled_len(struct dm_disk_if *);
```

Returns the marshalled size of the structure.

```
void *dm_marshall(struct dm_disk_if *, char *);
```

Marshalls the structure into the given buffer. The return value is the first address in the buffer not written.

1.4 Model Configuration

Diskmodel uses libparam to input the following blocks of parameter data:

```
dm_disk
dm_layout_g1
dm_layout_g1_zone
dm_mech_g1
dm_layout_g2
dm_layout_g2_zone
dm_layout_g4
```

$1.4.1 \, dm_{-}disk$

The outer dm_disk block contains the top-level parameters which are used to fill in the dm_disk_if structure. The only valid value for "Layout Model" is a dm_layout_g1 block and for "Mechanical Model," a dm_mech_g1 block.

dm_disk	Block count	int	required
This specifies 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 parameter	s. A warning is reported if the value	ies differ.	

dm_disk	Number of data surfaces	int	required
This specifies the number	of magnetic media surfaces (not p	latters!) or	which data are recorded.
Dedicated servo surfaces should not be counted for this parameter.			

dm_disk	Number of cylinder	`s	int	required
This specifies the number	of physical cylinders.	All cylinders	s that imp	act the logical to physical
mappings should be included	led.			

dm_disk	Mechanical Model		block	optional
This block defines the dis	k's mechanical model.	Currently,	the only	available implementation is
dm_mech_g1.				

dm_disk	Layout Model	block	required
This block defines the disk	s's layout model.		

1.4.2 G1 Layout

The dm_layout_g1 block provides parameters for a first generation (g1) layout model.

dm_layout_g1	LBN-to-PBN mapping scheme	int	required			
This specifies the type of I	This specifies the type of LBN-to-PBN mapping used by the disk. 0 indicates that the conventional					
mapping scheme is used:	LBNs advance along the 0th track	of the 0th	cylinder, then along the			
1st track of the 0th cylin	der, thru the end of the 0th cylind	ler, then to	the 0th track of the 1st			
cylinder, and so forth. 1 is	ndicates that the conventional map	oing scheme	e is modified slightly, such			
that cylinder switches do r	that cylinder switches do not involve head switches. Thus, after LBNs are assigned to the last track					
of the 0th cylinder, they are assigned to the last track of the 1st cylinder, the next-to-last track of						
the 1st cylinder, thru the 0th track of the 1st cylinder. LBNs are then assigned to the 0th track of						
the 2nd cylinder, and so on ("first cylinder is normal"). 2 is like 1 except that the serpentine pattern						
does not reset at the beginning of each zone; rather, even cylinders are always ascending and odd						
cylinders are always descer	nding.					

dm_layout_g1	Sparing scheme used	int	required	
This specifies the type of	sparing used by the disk. Later par	ameters de	termine where spare space	
is allocated. 0 indicates the	hat no spare sectors are allocated. 1	l indicates	that entire tracks of spare	
sectors are allocated at the	ne "end" of some or all zones (sets	of cylinder	s). 2 indicates that spare	
sectors are allocated at the	ne "end" of each cylinder. 3 indicat	es that spa	re sectors are allocated at	
the "end" of each track.	4 indicates that spare sectors are al	located at	the "end" of each cylinder	
and that slipped sectors of	do not utilize these spares (more sp	ares are lo	cated at the "end" of the	
disk). 5 indicates that spa	are sectors are allocated at the "fron	nt" of each	cylinder. 6 indicates that	
spare sectors are allocated	spare sectors are allocated at the "front" of each cylinder and that slipped sectors do not utilize			
these spares (more spares are located at the "end" of the disk). 7 indicates that spare sectors are				
allocated at the "end" of the disk. 8 indicates that spare sectors are allocated at the "end" of each				
range of cylinders. 9 indicates that spare sectors are allocated at the "end" of each zone. 10 indicates				
that spare sectors are allocated at the "end" of each zone and that slipped sectors do not use these				
spares (more spares are lo	cated at the "end" of the disk).			

dm_layout_g1	Rangesize for sparing	int	required	
This specifies the range (e.g., of cylinders) over which spares are allocated and maintained. Currently,				
this value is relevant only	for disks that use "sectors per cylin	nder range"	sparing schemes.	

dm_layout_g1	Skew units	string	optional
This sets the units with w	hich units are input: revolutions o	r sectors.	The "disk-wide" value set
here may be overridden pe	er-zone. The default unit is sectors	5.	

dm_layout_g1	Zones	list	required
This is a list of zone blo	ck values describing the zones/bands	of the disk.	

The Zones parameter is a list of zone blocks each of which contains the following fields:

dm_layout_g1_zone	First cylinder number	int	required
This specifies the first physical cylinder in the zone.			

dm_layout_g1_zone	Last cylinder number	int	required
This specifies the last phy	sical cylinder in the zone.		

dm_layout_g1_zone	Blocks per track	int	required
This specifies the number	of sectors (independent of logical-to	p-physical n	nappings) on each physical
track in the zone.			

$dm_layout_g1_zone$	Offset of first block	float	required
This specifies the physical	offset of the first logical sector in	the zone.	Physical sector 0 of every
track is assumed to begin a	at the same angle of rotation. This	may be in e	ither sectors or revolutions
according to the "Skew un	its" parameter.		

dm_layout_g1_zone	Skew units	string	optional
Default is sectors. This	value overrides any set in the surrou	inding layo	ut block.

${\tt dm_layout_g1_zone}$	Empty space at zone	e front in	nt required	
This specifies the size of the	he "management area"	allocated at the	e 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 mappin	ıg.			

dm_layout_g1_zone	Skew for track switch	float	optional		
This specifies the number	This specifies 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	d by the manufacturer to		
optimize sequential access	s. This may be in either sectors o	r revolution	as according to the "Skew		
units" parameter.					

dm_layout_g1_zone Skew for cylinder switch float optional

This specifies 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. This may be in either sectors or revolutions according to the "Skew units" parameter.

dm_layout_g1_zone Number of spares int required

This specifies the number of spare storage locations – sectors or tracks, depending on the sparing scheme chosen – allocated per region of coverage which may be a track, cylinder, or zone, depending on the sparing scheme. 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.

dm_layout_g1_zone | slips | list | required

This is a list of lbns for 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. Each integer in the list indicates the slipped (defective) location.

dm_layout_g1_zone defects list required

This list describes previously detected defective media locations – sectors or tracks, depending upon the sparing scheme chosen – that have been remapped to alternate physical locations. The elements of the list are interpreted as pairs wherein the first number is the original (defective) location and the second number indicates the replacement location. Note that these locations will both be either a physical sector number or a physical track number, depending on the sparing scheme chosen.

1.4.3 G1 Mechanics

The dm_mech_g1 block provides parameters for a first generation (g1) mechanical model.

dm_mech_g1	Access time type	string	required		
This specifies the method	d for computing mechanical delays	. Legal va	lues are constant which		
indicates a fixed per-rec	indicates a fixed per-request access time (i.e., actual mechanical activity is not modeled),				
averageRotation which indicates that seek activity should be modeled but rotational latency is					
assumed to be equal to one half of a rotation (the statistical mean for random disk access) and					
trackSwitchPlusRotation which indicates that both seek and rotational activity should be mod-					
eled.					

dm_mech_g1	Constant access time	float	optional
Provides the constant acce	ess time to be used if the access time	e type is se	t to constant.

dm_mech_g1	Seek type	string	required
------------	-----------	--------	----------

This specifies the method for computing seek delays. Legal values are the following: linear indicates that the single-cylinder seek time, the average seek time, and the full-strobe seek time parameters should be used to compute the seek time via linear interpolation. curve indicates that the same three parameters should be used with the seek equation described in [Lee93] (see Section 1.5.1). constant indicates a fixed per-request seek time. The Constant seek time parameter must be provided. hpl indicates that the six-value HPL seek equation values parameter (see below) should be used with the seek equation described in [Ruemmler94] (see below). hplplus10 indicates that the six-value HPL seek equation values 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 ten seek times parameter (see below) as in [Worthington94]. extracted 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. If extracted layout is used, the parameter Full seek curve (below) must be provided.

dm_mech_g1	Average seek time	float	optional
The mean time necessary	to perform a random seek		

dm_mech_g1	Constant seek time	float	optional
For the "constant" seek ty	rpe (above).		

dm_mech_g1	Single cylinder seek time	float	optional	
This specifies the time necessary to seek to an adjacent cylinder.				

dm_mech_g1	Full strobe seek time	float	optional
This specifies the full-stro	be seek time (i.e., the time to seek	from the	innermost cylinder to the
outermost cylinder).			

dm_mech_g1	Full seek curve	string	optional
The name of the input file	containing the seek curve data. T	The format of	this file is described below.

dm_mech_g1			settling	J	float	required		
This specifies the additional	al time	required	d to precise	ly settle th	ne read/wri	te head for	writing (after a	,
seek or head switch). As the	is para	meter ir	nplies, the	seek times	computed	using the ab	oove parameter	
values are for read access.								

dm_mech_g1	Head switch time	float	required	
This specifies the time required for a head switch (i.e., activating a different read/write head in orde				
to access a different media surface).				

dm_mech_g1	Rotation speed (in rpms)	int	required	
This specifies the rotation speed of the disk platters in rpms.				

dm_mech_g1	Percent error in rpms	float	required

This specifies the maximum deviation in the rotation speed specified above. During initialization, the rotation speed for each disk is randomly chosen from a uniform distribution of the specified rotation speed \pm the maximum allowed error. This feature may be deprecated and should be avoided.

dm_mech_g1	list	optional	
This is a list of ten floating	k time for s	seek distances of 1 through	
10 arrlindans			

dm_mech_g1	HPL seek equation values	list	optional
This is a list containing s	ix numbers specifying the variable	s V_1 through	v_6 of the seek equation
described in [Ruemmler94	(see below).		

1.4.4 G2 Layout

The dm_layout_g2 block provides parameters for a second generation (g2) layout model.

dm_layout_g2	Layout Map File	string	required
dm_layout_g2	Zones	list	required

The Zones parameter is a list of zone blocks each of which contains the following fields:

dm_layout_g2_zone	First cylinder number	int	required
This specifies the first physical cylinder in the zone.			

	$dm_layout_g2_zone$	Last cylinder number	int	required
This specifies the last physical cylinder in the zone.				

dm_layout_g2_zone	First LBN	int	required
The first LBN in this zone.			

dm_layout_g2_zone	Last LBN	int	required
The first LBN in this zone	<u>)</u>		

dm_layout_g2_zone	Blocks per track	int	required
This specifies the number	of sectors (independent of logical-to	p-physical n	nappings) on each physical
track in the zone			

dm_layout_g2_zone	Zone Skew	float	optional
This specifies 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. This may be in either sectors or revolutions			
according to the "Skew ur	according to the "Skew units" parameter.		

dm_layout_g2_zone	Skew units	string	optional
Default is sectors. This	value overrides any set in the surrou	ınding layo	ut block.

dm_layout_g2_zone	Skew for track switch	float	optional
This specifies the number	of physical sectors that are skip	pped when assign	ning logical block numbers
to physical sectors at a t	rack crossing point. Track sl	kew is computed	d by the manufacturer to
optimize sequential access	s. This may be in either sector	ors or revolution	s according to the "Skew
unita" peremeter			

	$dm_layout_g2_zone$	Skew for cylinder switch	float	optional
	This specifies the number	of physical sectors that are skippe	d when assig	ning logical block numbers
	to physical sectors at a c	ylinder crossing point. Cylinder s	kew is comp	outed by the manufacturer
	to optimize sequential acc	ess. This may be in either sectors	or revolutio	ons according to the "Skew
ı	units" parameter.			

1.4.5 G4 Layout

The dm_layout_g4 block provides parameters for a fourth generation (g4) layout model.

dm_layout_g4	TP	list	required
s0, – lowest sector number on the track sn, – highest sector number on the track spt – physical spt			
on the track Indicates a range of lbns [0,n] map to sectors [s0,sn] on a track. This will almost always			
be $[0,n-1,n]$ where n is the	be [0,n-1,n] where n is the physical SPT.		

$\mathtt{dm_layout_g4}$	IDX	list	required		
IDX contains a list of g4	IDX contains a list of g4 layout index nodes which are themselves represented as lists of integers.				
The inner lists contain th	e index entries which have 8 fields	as follows:	lbn, – the lbn of the first		
instance, relative to the pa	instance, relative to the parent cyl, – the lowest cylinder in the first instance, relative to the parent				
runlen, – the number of lbns this entry covers cylrunlen, – the number of cylinders for this entry len,					
– the number of lbns in one instance of the child cyllen, – the number of cylinders covered by one					
instance childtype, – whet	her the child is an index node (IDX) or track p	oattern (TP) child – index		
of the child node in IDX or TP The entries are given inline so the inner lists will have a multiple of					
8 entries. The last index node (last entry in the outer list) is the "root" index node that covers all					
of the LBN and cylinder space.					

dm_layout_g4	Slips	list	required
A list of slip locations. Each slip location is described by 2 integers, the first being the logical offset			
(lbn) and the second being the number of LBNs are slipped.			

ı	$\mathtt{dm_layout_g4}$	Remaps	list	required
I	A list of remapped defects.	Each remap is described by 6 integrated	ers: LBN, -	logical offset of remapped
ı	location len, – number of	contiguous sectors remapped cylind	er, head, se	ector, – remap destination
ı	spt – physical SPT on des	tination track		

1.5 Seek Equation Definitions

1.5.1 Lee's Seek Equation

$$seekTime(x) = \left\{ \begin{array}{rcl} 0 & : & if x = 0 \\ a\sqrt{x-1} + b(x-1) + c & : & if x > 0 \end{array} \right., \text{where}$$

x is the seek distance in cylinders,

 $a = (-10minSeek + 15avgSeek - 5maxSeek)/(3\sqrt{numCyl}),$

b = (7minSeek - 15avgSeek + 8maxSeek)/(3numCyl), and

c = minSeek.

1.5.2 The HPL Seek Equation

Seek distance	Seek time	
1 cylinder	V_6	where distincts and distance in culinders
$\langle V_1 \text{ cylinders} \rangle$	$V_2 + V_3 * \sqrt{dist}$, where $dist$ is the seek distance in cylinders.
$>=V_1$ cylinders	$V_4 + V_5 * \text{dist}$	

If $V_6 == -1$, single-cylinder seeks are computed using the second equation. V_1 is specified in cylinders, and V_2 through V_6 are specified in milliseconds.

 V_1 must be a non-negative integer, $V_2 \dots V_5$ must be non-negative floats and V_6 must be either a non-negative float or -1.

Format of an extracted seek curve

An extracted seek file contains a number of (seek-time,seek-distance) data points. The format of such a file is very simple: the first line is

Seek distances measured: <n>

where <n> is the number of seek distances provided in the curve. This line is followed by <n> lines of the form <distance>, <time> where <distance> is the seek distance measured in cylinders, and <time> is the amount of time the seek took in milliseconds. e.g.

Seek distances measured: 4

1, 1.2

2, 1.5

5, 5

10, 9.2

2 Installation

To Build Diskmodel:

- 1. build libparam and libtrace
- 2. edit .paths in the diskmodel source directory to reflect where you built libparam and libtrace
- 3. 'make' in the diskmodel directory

3 Typical use with libparam

```
'make all' sets up include and lib subdirectories such that you may use
-I$(DISKMODEL_PREFIX)/include
with the preprocessor and
#include<diskmodel/dm.h>
etc. Similarly,
-L$(DISKMODEL_PREFIX)/lib -ldiskmodel
with the linker where DISKMODEL_PREFIX is the top-level source directory where you built diskmodel.
   1. register diskmodel libraram modules with libraram. e.g.
#include <diskmodel/modules/modules.h>
for(i = 0; i <= DM_MAX_MODULE; i++) {</pre>
  lp_register_module(dm_mods[i]);
   2. use lp_loadfile() to load a model file
   3. use lp_instantiate() to instantiate a model from the input file. The result of the instanti-
ation is a struct dm_disk_if *
   e.g. struct dm_disk_if *disk = lp_instantiate(...);
   4. Access methods through d. e.g. dm_time_t seektime = d->mech->dm_seek_time(...)
```

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, December 1995, pp. 1263–1269.
- [Ganger98] G. Ganger, B. Worthington, Y. Patt, "The DiskSim Simulation Environment Version 1.0 Reference Manual", Technical Report CSE-TR-358-98, University of Michigan, Ann Arbor, February 1998.

- [Ganger99] G. Ganger, B. Worthington, Y. Patt, "The DiskSim Simulation Environment Version 2.0 Reference Manual", December 1999.
- [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.
- [Schindler99] J. Schindler, G. Ganger, "Automated Disk Drive Characterization", Technical Report CMU-CS-99-176, Carnegie Mellon University, December 1999.
- [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.