Statistical analysis of damaged data storage media read timings.

Corvus Corax

2009-07-28

Abstract

Data storage devices and drivers show characteristic behavior when operating on damaged media, especially concerning read timings. A read attempt on a damaged sector can take several magnitudes more time than a regular read attempt, which has implications on the performance of data recovery software. This article attempts to shed some light on the behavior of a couple of example hardware and its characteristics and tries to deduct consequences for the performance of recovery software and implied requirements for program behavior and development. The goal is to gather information for a "simulated bad media" environment which can be used to benchmark different data recovery tools under reproducible conditions which mimic real world bad media behavior.

Contents

1 Statistical data generation **2**
- 1.1 Options for data generation . . . 2
- 1.2 Data preparation . . . 3
- 1.3 Shortcomings . . . 3

2 Examples of bad media **4**
- 2.1 The KNOPPIX CD . . . 4
  - 2.1.1 Dell laptop DVD drive . . . 5
  - 2.1.2 LiteOn CDRW drive . . . 6
  - 2.1.3 Samsung DVD-ROM . . . 7
- 2.2 $5\frac{1}{4}$" floppy . . . 7
- 2.3 $2\frac{1}{2}$" IDE hard disk . . . 8

3 Implications on the efficiency of data rescue **9**
- 3.1 Implication on safecopy development . . . 11
- 3.2 Implications on other tools . . . 11

4 Benchmark creation **11**
- 4.1 Approximated timing behavior. . . . 12
- 4.2 Simulation requirements. . . . 12
- 4.3 Shortcomings . . . 13

5 Creation of test cases **13**
- 5.1 CDROM example . . . 13
- 5.2 Other CDROM drive . . . 14
- 5.3 Floppy example . . . 14

6 Conclusion **15**
6.1 Seek time evaluation on KNOPPIX CD 16

1 Statistical data generation

Statistical data is gathered using the tool *safecopy* (safecopy web page `http://safecopy.sourceforge.net`) version 1.5. It provides the following features:

- Attempt to read every sector of a storage medium.
- Generate and store timing information on each read attempt.
- Ability to identify different categories of sectors, as there are:
  - *readable sectors*
  - *unreadable sectors*
  - sectors that upon multiple read attempts sometimes succeed and sometimes fail. hence called *recoverable sectors*

1.1 Options for data generation

To produce significant statistical data, user definable options for safecopy are set as follows for all statistic gatherings:

–**sync** Safecopy will use synchronous or direct IO, bypassing kernel caching, reducing the imprint of kernel behavior on the read timing.

-L 2 Safecopy will use low level IO where applicable, which in case of CDROM media bypasses kernel driver error correction and reads sectors in so called *raw mode* directly from the drive.

-Z 0 This disables a feature of safecopy that forces head realignments after read errors occurred. Since head realignments induce delays for repositioning the drive head on the following read attempt, this would falsify acquired statistical data and needs to be disabled.

-R 4 Safecopy will try to read at least the first sector of a found bad area 4 times before giving up and skipping to the next sector. This is necessary to successfully detect some *recoverable sectors* and provide timing data on them.

-f 1* Safecopy will try to read every single sector of the drive, even in a concurrent bad area. The default behavior would be to skip over bad areas and only check some sectors to find the end of the bad area fast. Since we want statistical data on the behavior of *unreadable sectors*, we attempt to read on every sector instead.

-b xxx The preferred blocksize reported by the Linux kernel is 1024 byte on x86 systems and 4096 byte on x86_64 systems. However most hardware sector sizes are 512 byte. Force setting this causes safecopy to explicitly read every sector individually which is what we want. However some hardware uses different sector sizes. CDROMs for example have a logical sector size of (usually) 2048 byte.

-T timing.log Safecopy will store sector read timing statistics in this file.

-o badblocks.log Safecopy will list *unrecoverable sectors* in this file.

An example safecopy invocation to read from a CD drive on a Linux machine would be:

> safecopy –sync -L 2 -Z 0 -R 4 -f 1* -T timing.log -o badblocks.log -b 2048 /dev/cdrom test.dat

1.2 Data preparation

The relevant output of safecopy is stored in two files:

timings.log Holds sector numbers and read times in µ-seconds.

badblocks.log Holds a list of *unrecoverable sectors*.

To visualize this data meaningfully, the read timings need to be split into read times for the relevant groups.

unrecoverable sectors are easy to extract, since their sector numbers are listed in badblocks.log.

readable sectors are identifiable since they are not listed in *badblocks.log*, and listed in *timing.log* exactly once.

recoverable sectors are identifiable since they are not listed in *badblocks.log*, but listed in timing.log multiple times, since safecopy attempted to read them more than once. This group can be split further into:

- **unsuccessful read attempts** on recoverable sectors, which are all entries with a result code of -1.
- **successful read attempts** on recoverable sectors, which are entries with a positive result code.

Sector numbers and timings for all of these groups are generated by standard text file processing tools and stored in separate files.

These files can then be read by a graphical visualization tool like *gnu-plot*.

1.3 Shortcomings

Safecopy only tries to repeatedly read the first sector of any set of adjacent sectors that cause read errors. As such, it can not distinguish between *recoverable sectors* and *unrecoverable sectors* within or at the end of such a set. Also the amount of read attempts is limited to a finite amount, which implies to possibility of missed *recoverable sectors*, just because they did not recover during the attempted read calls. Therefore it must be assumed that some of the sectors listed as unrecoverable are hidden *recoverable sectors*.

On the other hand some of the sectors identified as *readable sectors* could be hidden *recoverable sectors* that happened to be successfully read on first attempt. Then again every sector has the potential to eventually fail.

This has implications when using the gathered data as a template for simulated bad media, as discussed below.

2 Examples of bad media

To provide data on real world behavior, statistic data has been gathered on real damaged media under several different conditions.

2.1 The KNOPPIX CD

CDROMs are prime suspects for data recovery statistics since they are very common medias, easily damaged, and common damages to them like scratches are easily reproducible. The particular CD used for data generation is worn from heavy use and additionally had purposely made deep scratches on both the front and backside of the self burned CD, producing both permanently damaged sectors and sectors on which a drive with good error correction has a chance of recovery. This CD is read in two different drives.

Figure 1: CDROM top.

Figure 2: CDROM bottom.

Physical sector size of this medium is 2352 bytes, with 2048 bytes user data and 304 bytes error correction and sector address coded. Safecopy can address it in RAW mode, reading the 2352 byte sectors directly, but stores only 2048 byte user data in the destination file. Overall disk size (user data) is 730040320 byte (696 MB).

An interesting effect on data recovery is, while the area with missing reflective layer around sector #100000 is just about 4 millimeters in diameter, the affected count of unreadable sectors is often up to 12 sectors, which is more than half a track, sometimes leaving only a few sectors readable on that track.

The likely cause of this behavior, which occur ed on all tested CD-drives, is that the laser optic looses its focus where the reflective layer is missing, and needs some time to refocus on the CD surface. Since the CD is constantly spinning, sectors that "spun through" during refocusing time cannot be read. The focus loss appears on every turn in that area, reproducing the "after scratch out of focus effect" so to speak. Workarounds for that could be to force an extremely low spinning speed, but since that cannot be easily set in software, it requires a firmware patch or special hardware meant for data rescue.

Re-establishing the reflective features of the CD back surface could possibly reduce the out of focus effect, even if they cannot make the directly affected data readable it would make the directly following data on the same track re-accessible.

2.1.1 Dell laptop DVD drive

The mentioned CD is read in a Dell Inspiron 8200 inbuilt DVD-ROM drive, identifying itself as

> HL-DT-STDVD-ROM GDR8081N, ATAPI CD/DVD-ROM drive
> ATAPI 24X DVD-ROM drive, 512kb Cache, DMA

This drive is handling scratches on the front-side of the media badly, unable to recover most of the affected sectors. Interesting to see is the relative difference in absolute reading speed as the drive automatically adjusts its spin speed when it encounters problems.

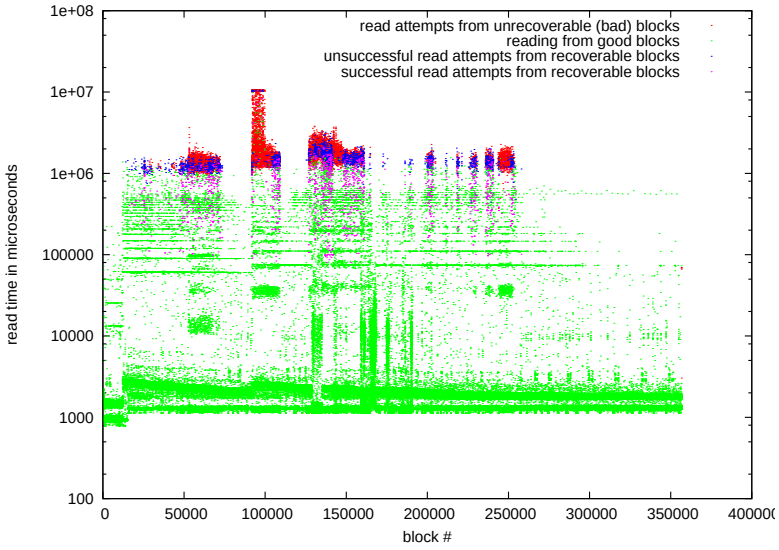

Figure 3: Sector read timings.

As one can see, read attempts on *unrecoverable sectors* are all slower by about 3 orders of magnitude. Reading from the area where the reflective layer has been damaged is even slower, leading to recovery times up to 10 seconds per block.

Unrecoverable sectors are heavily mixed with *readable sectors* and *recoverable sectors* in bad areas, usually with both types on the same track, only a small number of sectors apart, but spanning over a noticeable span of tracks affected by the error cause.

Note: This statistic diagram has been created with an older timing.log creation algorithm, not including a result code for sector read attempts. Therefore its listing some bogus *unsuccessful read attempts* which had been *readable sectors* just past the end of a bad area which have been read twice. Eventually this sheet needs to be re-created to distinguish this group from real recovered sectors.

2.1.2 LiteOn CDRW drive

The same CD is read in a

> LITE-ON LTR-32123S, ATAPI CD/DVD-ROM drive
> ATAPI 40X CD-ROM CD-R/RW drive, 1984kB Cache

This drive performs significantly better in terms of reading from problematic media. It has read most of the data covered by scratches on the underside of the CD. Its hard to belief this was the same source medium.

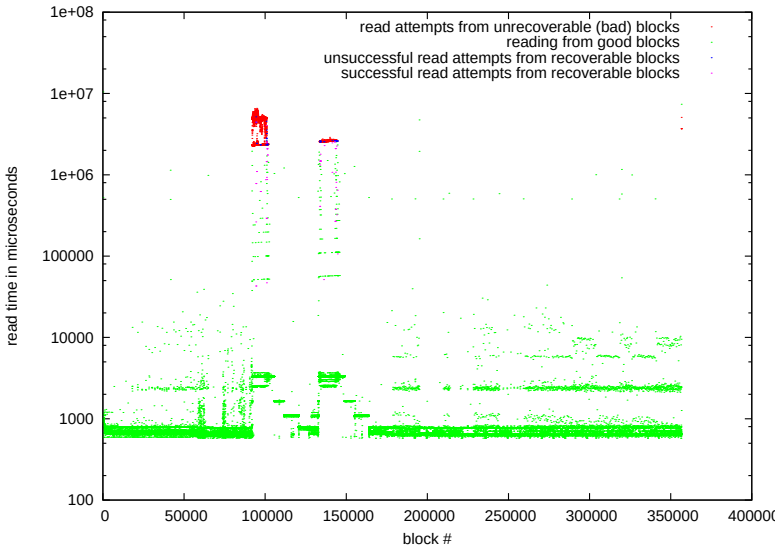

Figure 4: Sector read timings.

Here, too, *unrecoverable sectors* are heavily mixed with *readable sectors* and *recoverable sectors* in the bad area, as described above due to the laser refocusing issue.

2.1.3 Samsung DVD-ROM

SAMSUNG DVD-ROM SD-608, ATAPI CD/DVD-ROM drive
ATAPI 32X DVD-ROM drive, 512kB Cache

This particular drive has not been used for statistics collection, it is just mentioned for reference. Any occurred unreadable sector caused a firmware crash, leading to a 60 second timeout during which the drive would not respond, followed by an ATAPI bus reset and a drive reset, automatically issued by the kernel, rebooting of the drive firmware. Reading an entire damaged CD on this drive would simply have taken far too long.

2.2 $5\frac{1}{4}$" floppy

While hopelessly outdated, any old floppy disks still around are by now bound to have at least some unreadable sectors due to loss of magnetization, dust, long term effect of background radioactivity, etc. Therefore its likely that any important data needing to be recovered from such a disk will be subject to data recovery tools.

Figure 5: $5\frac{1}{4}$" floppy disk.

Sector size of this medium is 512 byte, stored in IBM AT 1.2 MB 80 track low level format. Safecopy reads this medium through the Linux kernel driver from a standard (though old and dusty) $5\frac{1}{4}$" floppy drive using O_DIRECT for reading from /dev/fd1, seeing only the 512 byte user data per block. This particular disk has 3 unreadable sectors on tracks 0 and 1 for unknown reasons (probably loss of magnetization).

A significant feature of these drives is the high head positioning time, resulting in high read times for the first sector of each track. This delay is is in the same order of magnitude as the delay imposed by read errors. However the exact timings for sectors in the different groups are all significantly distinct.

Bad sectors are more than 4 orders of magnitudes slower to respond than good ones.

A feature of this particular disk is the low amount of *unrecoverable sectors*. Never the less, the existing bad sectors occur physically and logically close to each other, grouped in a *bad area*.

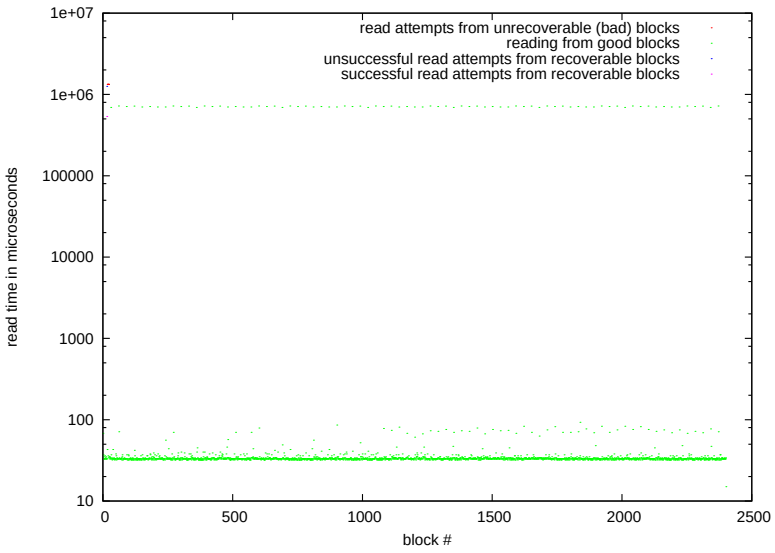

Figure 6: Timing behavior of this disk.

2.3 $2\frac{1}{2}$" IDE hard disk

SATA and PATA hard disks of different capacity are among the most common hard disks found, and as such often subject to data recovery. According to Murphy's law the probability of fatal hard disk failure increases exponentially with the time since the last backup. Recovery performance on these disks is heavily influenced by intelligent drive firmware, sometimes even for the better.

Figure 7: IDE disk drive.

This is a 6007357440 byte (6 so called GB) IDE drive with a logical sector size of 512 byte, identifying as:

> TOSHIBA MK6015MAP, ATA DISK drive
> Model=TOSHIBA MK6015MAP, FwRev=U8.12 A, Serial No=51K92837T

This drive had been removed from a laptop after sector read errors occurred.

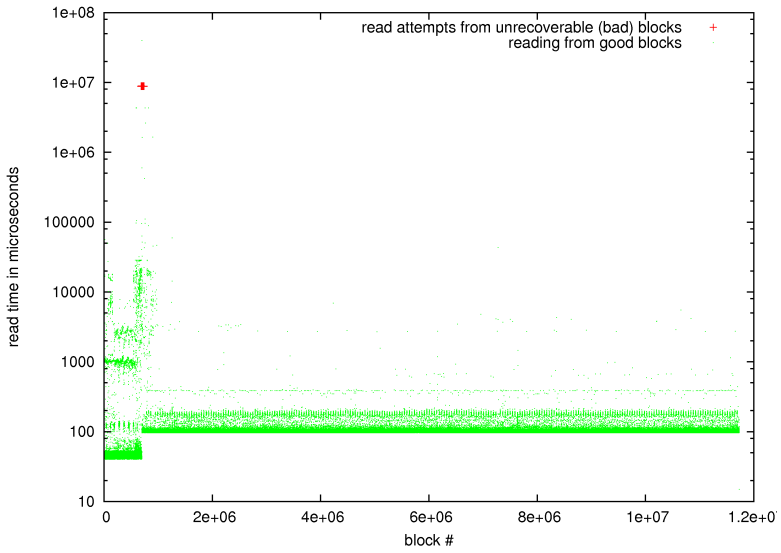

Figure 8: Sector read timings.

As one can see, similar to the behavior of CD-ROMS, the drive firmware reduces the absolute drive speed after it encounters errors. However the drive does not spin up again until a subsequent power down. On hard disks, the difference in read timings between successfully read data and errors is extreme, ranging over 5 orders of magnitude! (100 SEC versus 10 seconds per sector)

Note: No recoverable sectors have been found on this particular drive. However earlier field tests of safecopy on damaged IDE drives encountered several cases where sectors could be recovered. Unfortunately no timing data is available on those.

3 Implications on the efficiency of data rescue

The examined data confirmed the assumption that accessing *unrecoverable sectors* or even *recoverable sectors* is exceedingly slow to accessing *readable sectors*. It also shows a typical distribution of unrecoverable sectors, which are clustered in *bad areas*, caused by a shared error cause (for example a scratch, or a head crash). This bad areas can span from a few tracks up to a significant part of the storage medium, depending again on the error cause and severity.

It needs to be kept in mind that on some media (for example a HDD with a head crash) accessing *unrecoverable sectors*, or even *bad areas* can potentially further damage the drive, rendering more sectors unreadable and increasing data loss.

Therefore, when recovering data from a bad medium, accessing *unrecoverable sectors* should be avoided completely and accessing *recoverable sectors* should be minimized and take place at the end of a recovery process. Accessing data outside of *bad areas* should take place at the beginning.

Within a *bad area* itself there are two likely sector distributions:

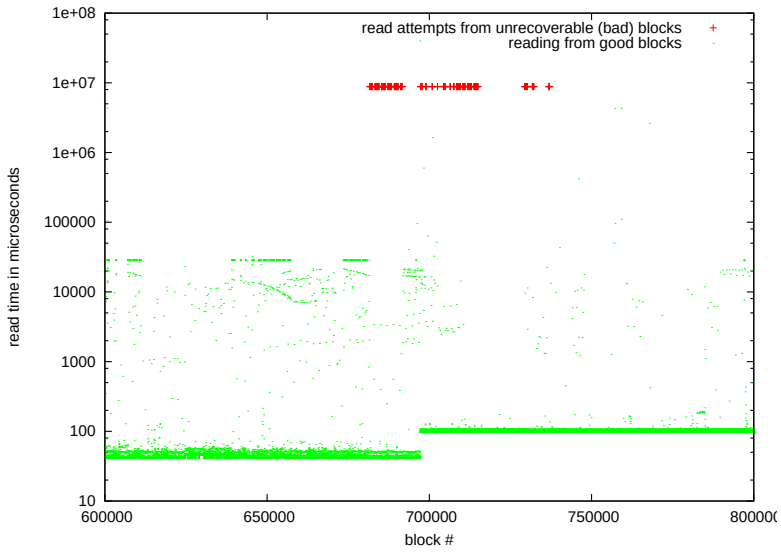

Figure 9: A closeup of the bad area.

1. All sectors in the bad area are *unrecoverable sectors*, optionally with some *recoverable sectors* at the rim.

2. Parts of the affected tracks are readable, resulting in an approximately periodic oscillation of *readable sectors* and *unrecoverable sectors* optionally with some *recoverable sectors* at the borderline. All of the analyzed media for this article showed this type of distribution.

The first, straightforward error distribution can be dealt with efficiently by establishing its begin and end sector with a small amount of read attempts and reading the rest of the disk, by algorithms current data rescue tools (safecopy, ddrescue, etc) already apply.

For the second (and apparently more common) distribution however, a satisfying solution is difficult to find. Even finding the end of the *bad area* with few read attempts is tricky, since there is a chance that such a read attempt matches a *readable sector* within the *bad area*, suggesting a premature end of the *bad area*. This could be coped with heuristic approaches, for example by analyzing the periodicity of the disk and guessing the likeliness of an *unreadable sector* based on its track position modulo relative to the first encountered bad sector. For this however the disk geometry must be known, since sector counts per track vary from disk to disk and sometimes even from track to track. In such a case the likeliness of guessing correctly would be increasingly low the further a guessed sector is apart from the reference sector.

The other problem with the second distribution is potentially recoverable data within a *bad area*. Especially on hard disks with multiple physical disks inside, the amount of data on a singly cylinder (spanning all disk surfaces) and as such between two sets of *unrecoverable sectors* can be significantly high, and potentially span entire files that need to be recovered.

It is thinkable that a heuristic approach could rescue some of this data fast,

if making use of the periodicity and the Gaussian-like guessable size of the bad spot within each track, avoiding those areas. However this still won't rescue readable and recoverable sectors closer to the unrecoverable sectors on the same track, requiring a more thorough access later.

3.1 Implication on safecopy development

The bad area skipping algorithm of safecopy currently works quite efficiently at avoiding *unrecoverable sectors* within a *set of unrecoverable sectors* on a single track. However skipping over entire *bad areas* in a first rescue attempt is done via a relatively dump approach (–stage 1 skips 10% of the disk regardless of the size of the erroneous area). As mentioned above, finding the end of an oscillating *bad area* efficiently is a non trivial process if the disk geometry is not known. A heuristic algorithm that efficiently determines those could therefore increase the data rescue effectiveness of safecopy.

The same is true for rescuing within a *bad area*. A probabilistic approach could proactively avoid accessing sectors if they have a high likeliness of being unreadable, and treat them like *unreadable sectors*, to be accessed in a later safecopy run.

For both of these the periodicity of errors needs to be learned on the fly for each particular medium. Since safecopy can distinguish some hardware by means of its low level driver, for some media a assumed geometry can be provided by the low level back end.

The best implementation would probably be an optionally activatable heuristic module, which would dynamically adjust seek resolution and failure skip size, and proactively assume sectors to be bad based on probabilism. Since the geometry is subject to change, the probabilism should decrease with the "distance" to the last occurred "real" error.

3.2 Implications on other tools

After creating a benchmark to create reproducible test result of the behavior of a recovery tool under realistic conditions, this can be used to benchmark other tools to show their strengths and shortcomings in comparison, and lead to conclusions on the effectiveness of different approaches.

4 Benchmark creation

A benchmark for data recovery should mimic the behavior of a set of representative test cases of real bad media. For the creation of this benchmark suite both the qualitative effect of a sector read attempt and the time consumption of such should be realistic.

A good starting point for such a benchmark is the debug test library coming with safecopy, since it sets up a test environment transparently for a to be tested program unaware of this virtual environment. The test library of safecopy 1.4 already provides the ability to produce simulated recoverable and unrecoverable sectors, so all it needs in addition to this is the ability to simulate timing behavior, and a reference set of to be simulated realistic test media behavior.

4.1 Approximated timing behavior.

As the analysis of the timing statistics of real media above suggests, the timing behavior of tested media can be approximated with a simplifying timing behavior, as long as the spanning of read timing over orders of magnitude is kept in a way it would still mimic the characteristic of real media. Non exponential variations on the other side can probably be ignored since they are subject to uncontrollable dynamic causes (CPU and IO load caused by other processes, etc). That leads to the following observations.

- The majority of *readable sectors* are read within a certain constant time frame$T1 \pm E$.
- Reading from a limited amount of *readable sectors* s takes longer than $T1 \pm E$ by approximately factor 2^{X_s}.
- Attempting to read from an *unrecoverable sectors* b takes time $(T1 \pm E) * 2^{X_b}$, which is usually several orders of magnitude bigger than T1.
- Reading from *recoverable sectors* either fails, in which case the read attempt takes about as long to respond as an *unrecoverable sector*, or it succeeds within the time bounds of a *readable sector*.

Therefore a *simplified media timing characteristics* specification can be given for any media by specifying the following:

- A base time T1 used for the majority of *readable sectors*.
- A sector timing exponent $x_s \in \{Xs\}$ for each *readable sector* s that takes significantly long to read from.
- A sector timing exponent $x_b \in \{Xb\}$ for each *unrecoverable sector* b.
- Two sector timing exponents $x_r \in \{Xr\}$ and $Y_r \in \{Yr\}$ for each recoverable sector r, storing the reading time in both the readable and unreadable case.

4.2 Simulation requirements.

A benchmark tool that wants to simulate a damaged medium according to the *simplified media timing characteristics* deducted above must simulate the following:

- *readable sectors* that respond in time T1.
- a limited defined set of *readable sectors* that respond in time $T1 * 2^{X_s}$.
- a limited defined set of unrecoverable sectors that respond with failure in time $T1 * 2^{X_b}$,
- a limited defined set of recoverable sectors r that respond either
  - with failure in time $T1 * 2^{X_r}$ or
  - with success in time $T1 * 2^{Y_r}$.

Where T1, {Xs}, {Xb}, {Xr}, {Yr} and the sectors in each set are the *simplified media timing characteristics* of the test medium.

These requirements are met by the modified test debug library as shipped with safecopy-1.5.

4.3 Shortcomings

Some characteristic behavior of media can not be described by the *simplified media timing characteristics*. This model simulates sector read times for each sector individually. On real world media however, the read time also depends on the state of the drive.

One important feature not modelled is the seek time a drive needs to address a certain sector. Instead, as can be seen on the floppy disk data, these times are simulated as a feature of a single sector (for example the first sector of a track). However this doesn't apply to real world media, as this delay depends on the sequence and order of sectors being read. For example reading a media backwards would induce this delay on the last sector of a track instead.

A possible workaround would be to measure seeking times on a device and include them into the simulation. However the seek time is a complex function depending on both head position, turning speed, turning position and several other drive dependant parameters, which can not easily be measured in the generic case. A linear delay function depending on the block address distance of current position and seeking destination could probably create better simulation results if the reading software does a lot of seeking, however the accuracy of simulation would still be lacking.

5 Creation of test cases

To create and demonstrate a benchmark test case, the statistical data of a real media test must be analyzed and the parameters of the *simplified media timing characteristics* need to be determined. This test case should then be validated by comparing the statistics generated on the test case to the base data set of the real media and confirm that the test case behaves "reasonably similar" to the original. "Reasonably similar" hereby means, closer to each other than to the typical damaged media that is not the reference source, while also making recovery tools behave similarly, when comparing success rate and time consumption.

5.1 CDROM example

Data source for this example is the above mentioned KNOPPIX CD read on the Dell.

Calculating the exponent multiplicands based on T1 leads to the following *simplified media timing characteristics*:

Simulating this disk, using the safecopy test debug library, measuring sector read timings with safecopy-1.5, the following sector read timing statistics were gathered:

As expected, there are random derivations of actual sector read timings from the simulated times, but they stay within the estimated error band E

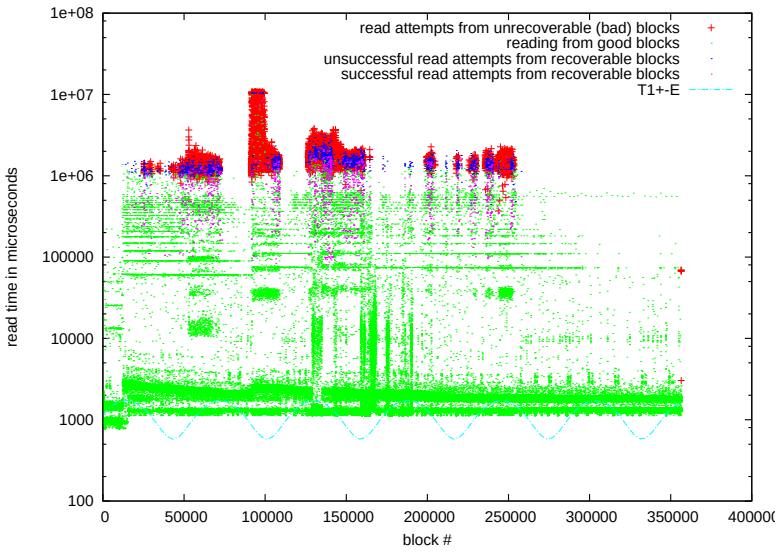

Figure 10: Original read timings.

where $t = (T1 \pm E) * 2^{X_s}$. Therefore the simplification of the simulation results would reproduce the original *simplified media characteristics* given by T1,{Xs},{Xb},{Xr},{Yr}.

During simulation, the same sectors were recovered as in the original safecopy run on the real disk and read attempts were undertaken on the same sectors. Intuitively the above requirements of "reasonable similarity" seem to be met. To give this claim foundation it is necessary to calculate the accumulated performance difference between original run and simulation based on the gathered statistics:

The simulation by *simplified media characteristics* caps the reading time error at ±200% for each single sector, while the overall reading time for the whole disk is determined by the reading time error accumulated through the most dominant sectors. In the case of this CD, those clearly are damaged sectors.

The similarity between the two curves for overall reading time is clearly visible in this diagram.

5.2 Other CDROM drive

The same CD in a LightOn drive:

5.3 Floppy example

Data source is the above analyzed 5¼" floppy disk.

Most of the simulated *readable sectors* take a bit too long to read. However the overall simulation is too short by about 20%, since the significantly long read times of the dominant first sector of each track (originally caused by head seeking) are simulated about that percentage too fast. This errors are within the expected error margin of the *simplified media characteristics*.

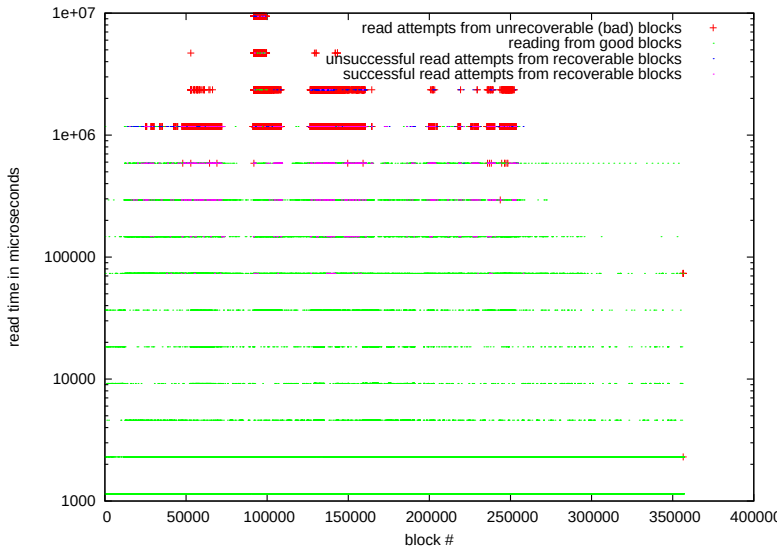

Figure 11: *Simplified media characteristics.*

6 Conclusion

It seems to be possible to undertake a reasonably accurate simulation of the behavior of damaged media and use this for benchmarking recovery tools based on *simplified media characteristics*.

The most important aspect not covered is, that a benchmark simulation would need to simulate the time consumption of seeking. Otherwise data recovery tools that do a lot of seeking on the source media had the potential to behave significantly better on the benchmark than on real damaged media. However a linear delay, based on sector address difference should be sufficient to penalize excessive seeking, even if it does not simulate real world seek times accurately in all cases.

Special consideration needs to be taken about media where seeking times are high compared to read times of damaged sectors even during a linear read, like the seen $5\frac{1}{4}$" floppy disk. If the seek time is simulated by the above separate algorithm, significant delays inflicted on single sectors by seeking during statistics gathering of the source must be compensated by taking those sectors out of the {Xs} set.

Luckily most media types have insignificant seek times when reading linearly, so no compensation needs to be done.

To estimate the seek time, we compare the sector read times of two readable sectors that are physically far apart and are not in {Xs} (meaning their read time when reading linearly is within $t = (T1 \pm E) * 2^0 = T1 \pm E$) when read directly after each other. The seek time to seek a distance of one sector would then be calculated as $t_{seek} = \frac{t_2 - T1}{n}$ where t2 is the read time of the second sector and n is the numbers of sectors the two sectors are apart.

This delay therefore is measured on some media covered by this article, and the results are bundled into a benchmark suite, to be published when safecopy-1.5 is, together with this article, data files and scripts.

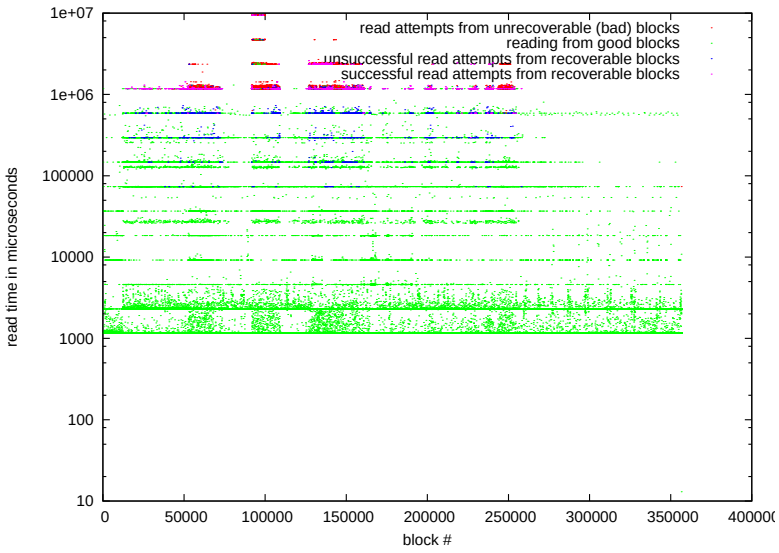

Figure 12: The simulation result.

6.1 Seek time evaluation on KNOPPIX CD

Doing a forced seek over the whole disk (on the Dell drive) with a sparse include file using safecopy, resulted in the following read times:

After setting up the simulator to delay to 649 nanoseconds per seeked sector, the simulation of the same sequence lead to these very accurate timings:

The difference in seek and read times when reading media "backwards" however are still not simulated. My assumption would be that seeking backwards on a CD might be slower, while on a hard disk there should be no measurable difference. However since safecopy does not offer a "read backwards" mode, measuring these is currently just too much effort.

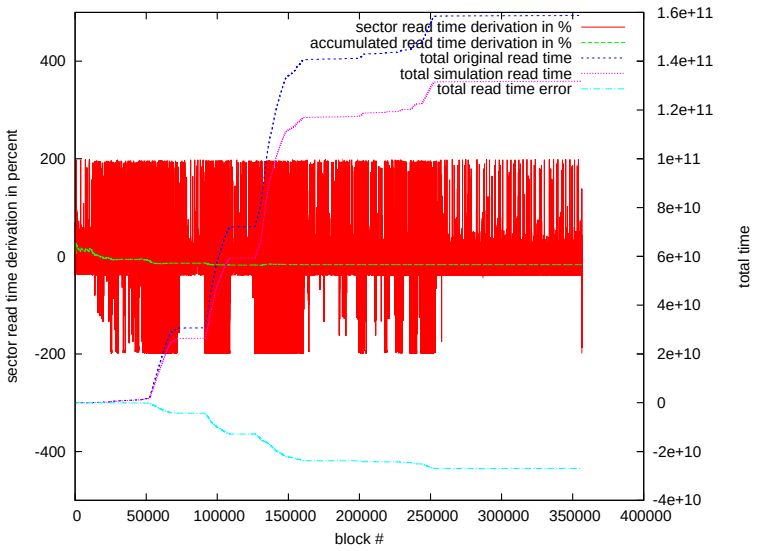

Figure 13: The error between reading the media and the simulation.

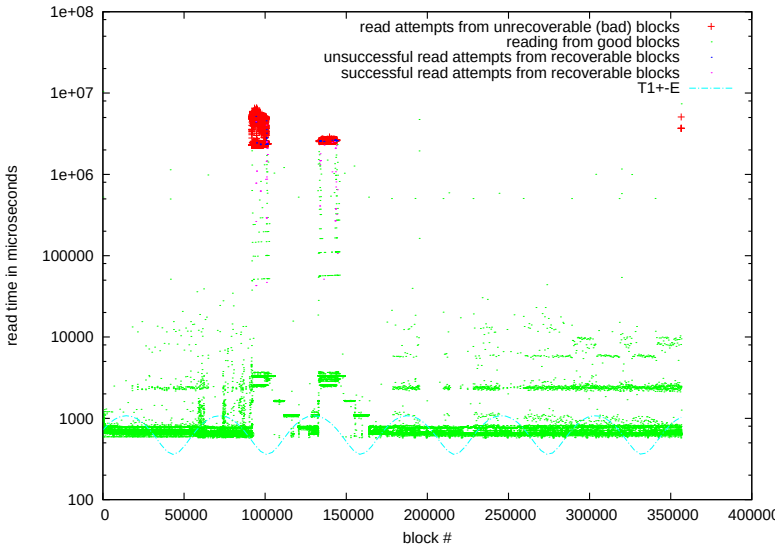

Figure 14: Original.

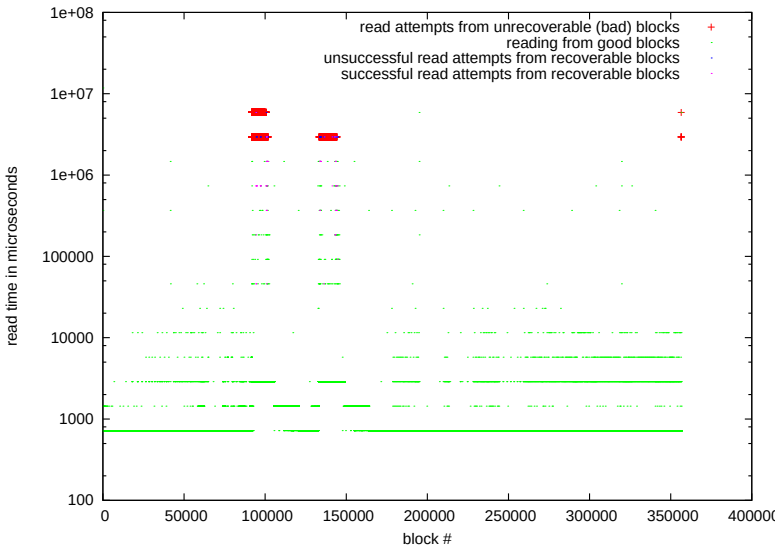

Figure 15: *Simplified media timing characteristics.*

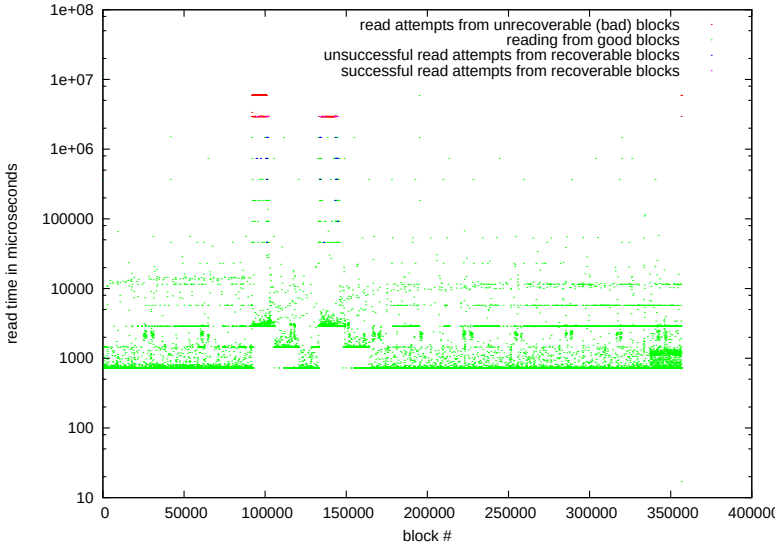

Figure 16: Simulation.

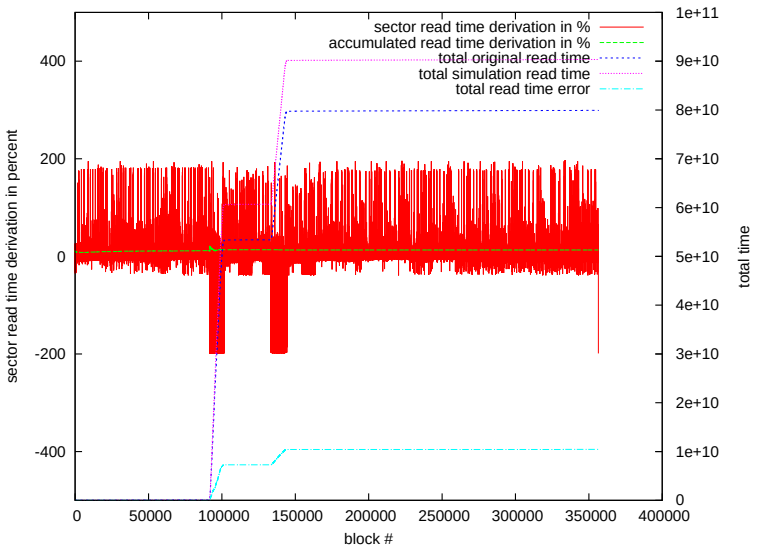

Figure 17: The error between reading the media and the simulation.

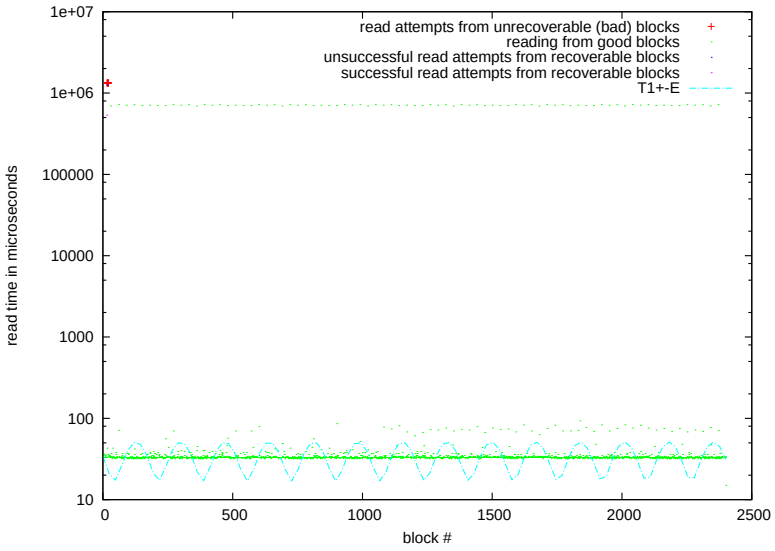

Figure 18: Read timings on the original disk.

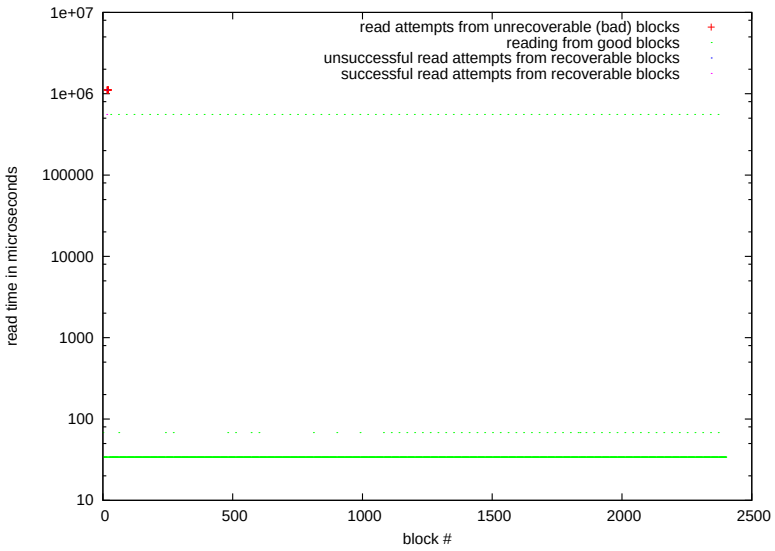

Figure 19: *Simplified media timing characteristics.*

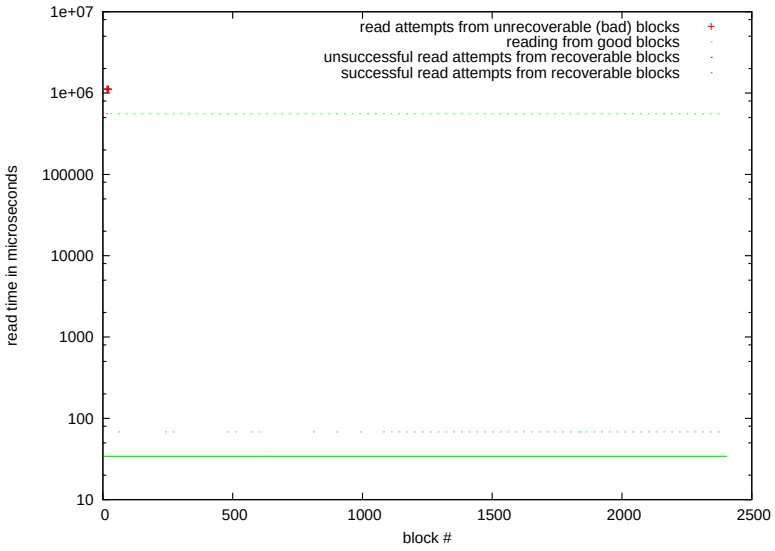

Figure 20: Simulation.

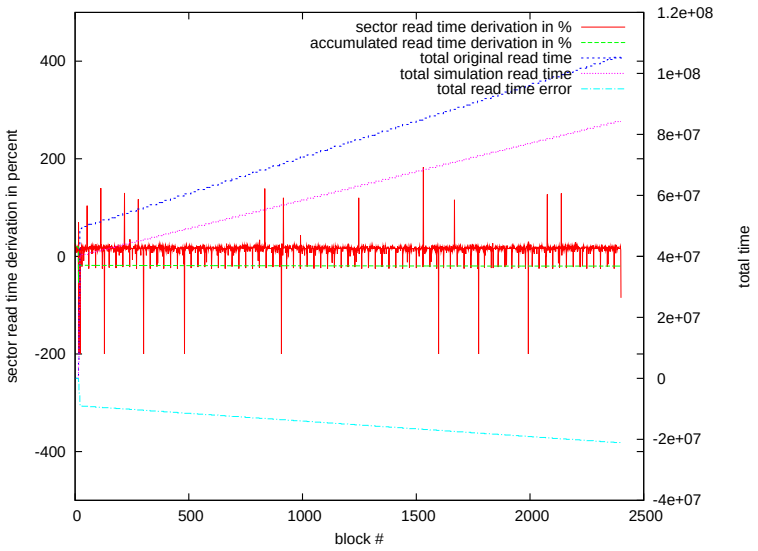

Figure 21: The error between reading the media and the simulation.

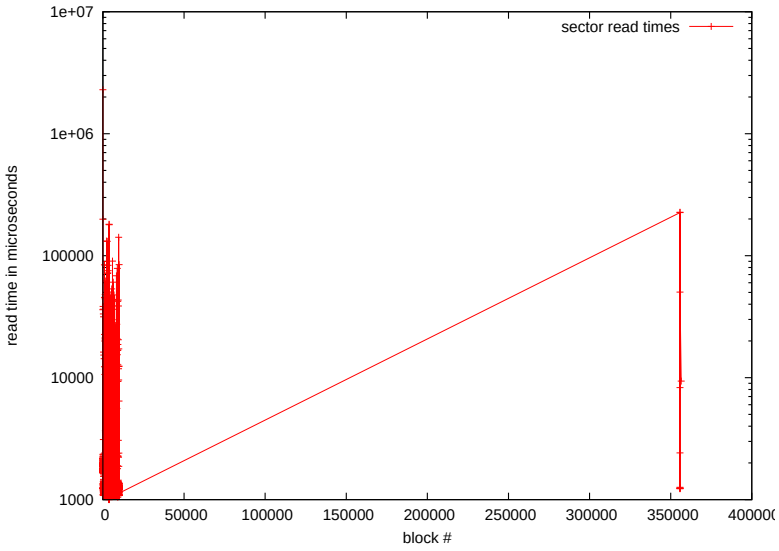

Figure 22: Sector seek time on original disk.

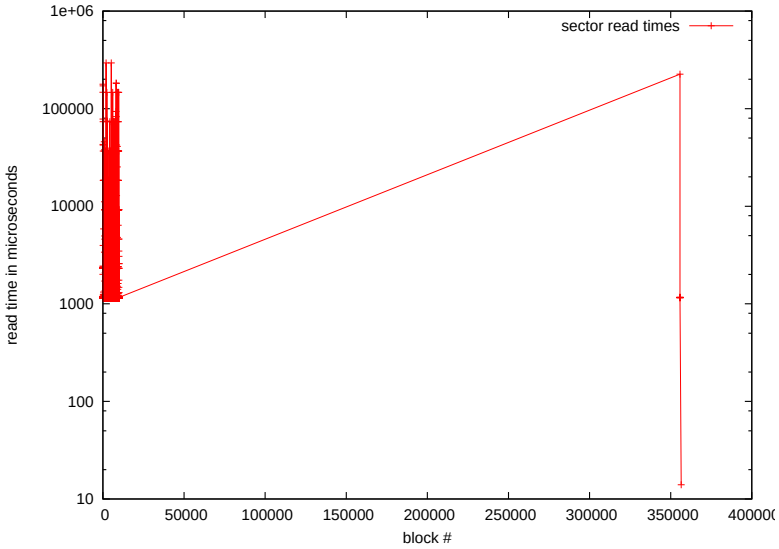

Figure 23: Simulation of sector seek time.