Raaid Ahmad
Storage Systems 600.419

October 1, 2004
Project #1

I. Reading Questions:

A) Both system implementers and system researchers value knowledge of low-level performance characteristics of disk drives. This information can help implementers determine what system policies to use and exclude from file systems. Performance characteristics and drive information can also be used by system researchers to allow more efficient research and data extraction from systems being researched. Obtaining this information accurately and in a timely manner is difficult because manufacturer specifications are often incorrect, not up to date, or available.

Three micro-benchmarks, Skippy, Zoned, and Seeker are used to solve this problem of disk information extraction. Each of these three benchmarks extracts different pieces of data about the hardware being tested. The important aspects of the proposed solutions are the broad goals for micro-benchmarks. Benchmarks should be general, complete, accurate, and fast. The combination of the three micro-benchmarks allows all four conditions to be met.

B) The authors' solutions are excellent. This paper contributes many critical intuitions to the storage systems area. For example, many previous benchmarks had tried to overcome rotational latency to obtain accurate benchmarks. Due to the random nature of rotational latency, these benchmarks did not perform well. SKIPPY is the first benchmark that actually exploits rotational latency to extract more information.

In addition to the SKIPPY algorithm, the paper also contributes a general standard for good micro-benchmarks. The paper claims that ideal benchmarks should be general, complete, accurate and fast. An explanation for each of these warrants is given and sets a standard for future benchmarks.

C) I think this experiment was a good opportunity to explore different benchmarking algorithms but I do think too much was left to interpretation by the experimenter. There is so much data and so many different results to interpret that the multiple variables in each and every experiment make it difficult to understand the details.

I think clarity of the experiment can be improved if for some of the algorithms, expected results are given. If some expected results are given, it allows the experiment to know whether or not they are on the right track. Another benefit of expected results is that understanding the expected results can easily help convey concepts. If no expected results are given at all, experimenters may start out by making incorrect assumptions and may confuse themselves by getting off on the wrong foot.

II. System Questions:

A) False. The lseek() system call does not actually cause a disk drive to seek. The lseek() system call is a way to manipulate a pointer to a specific place on the disk. When the lseek() function is called, the disk doesn't actually do anything, the offset point for the associated device is changed. Seeks are actually performed as a part of the read and write system calls. When a read or write call is executed, the disk seeks to the pointer (set by lseek()) and then performs the specified operation.

This fact was discovered during the programming of the HOPPY algorithm. During testing, the algorithm was as follows:

```
ds_lseek(/* seek to 0 */);
// Begin Time
ds_lseek(/*seek nth sector */);
ds_write(/*write to current sector*/);
// End Time
```

This algorithm for HOPPY yielded a uniform write time equivalent to the rotational latency of the disk. This meant that the disk arm was never actually moved back to 0, but instead was kept at the last sector written to. The solution to the problem is denoted below:

```
ds_lseek(/* seek to 0 */);
ds_write(/*write to sector 0*/);        // Perform Write to Reposition Head @ 0
// Begin Time
ds_lseek(/*seek nth sector */);
ds_write(/*write to current sector*/);
// End Time
```

B) The read() calls should exhibit the same behavior as the write() calls. The only difference is that since reads may take a shorter period of time since no data is being written to the disk. As such, one would expect the read() call to be approximately the same shape as the regular SKIPPY algorithm graph, with a smaller magnitude.

III. Programming Questions:

A) SKIPPY

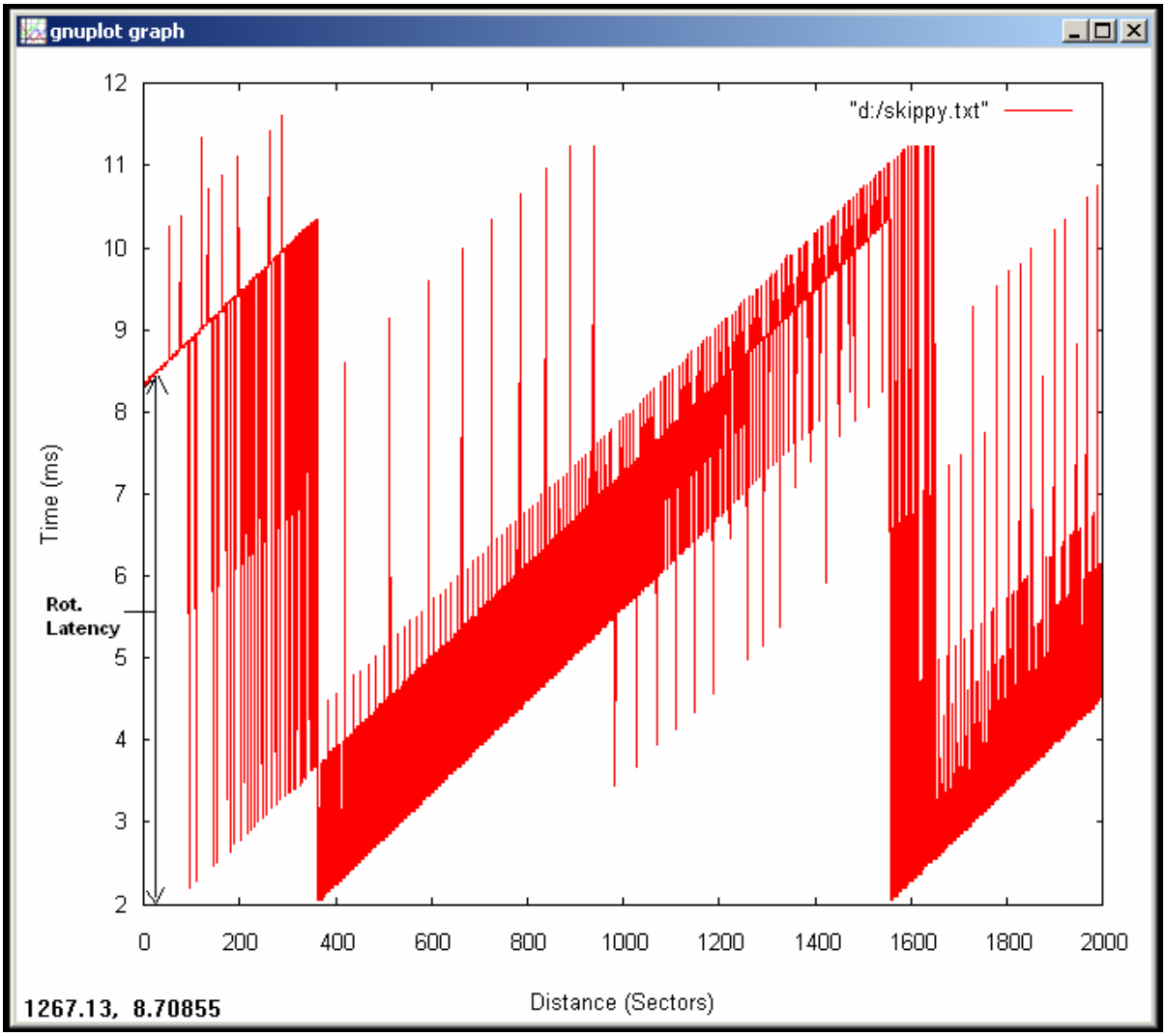

Figure 1

The graph obtained from the DiskSim (Figure 1) simulation is very similar to the graph in section 5, figure 14. The only differences are periodic measurements that read far above or below their expected value. These values may represent slipping or remapping on the disk. The fluctuations seem to be periodic and at a fixed ratio, which would make remapping an unlikely cause of the aberrations. Instead, the behavior exhibited by the graph can be explained by what happens when sectors are "slipped." When a hard drive has a bad sector, the next physical sector is used to represent the space of the previous sector on the logical sector map. In terms of the graphical interpretation, this means that when a sector is being written to by the SKIPPY algorithm, and it is a bad sector, the algorithm attempts to write the sector, fails and then attempts to write the next sector, before succeeding. This slipping is what might cause the large spikes in the graph.

The rotational latency of this drive is approximately 8.33 ms. This number is derived from the graph very easily. At the beginning of the skippy algorithm, the offset is very small, which means that consecutive writes incur nearly full rotational latency since the platter must spin an entire time before it can write to consecutive (or extremely close) sectors. For this reason, the first few values recorded by the skippy algorithm represent the rotational latency. From the rotational latency, the speed of rotation can easily be determined. If the platter spins once in 8.33 milliseconds, it revolves approximately 7202 times in 1 minute. **As such, it can be concluded that the hard drive being tested spins at approximately 7200 RPM.**

Additional analysis of the graph reveals the head switch time to be approximately 1.66ms (Figure 2). This can be taken by taking the difference between the normal time returned and a small jump followed by a return to approximately the same value before the jump. Head switches represent the time taken to switch from one recording surface to another. This head switch time does not, however, include a cylinder switch. Table 1 shows a head switch from sector 481 to 482.

Distance (Sector Offset)	Time (ms)
481	2.688000
482	***4.346000***
483	2.689000

Table 1

After a few head switches, a larger jump can be identified on the graphs. These larger jumps represent cylinder switches in the hard drive. A cylinder switch includes a head switch. On this particular hard drive, there are 4 peaks representing head switches, in between cylinder switch peaks. **This denotes that there are 5 heads on this hard drive because the final cylinder switch after the 4 head switches is a cylinder switch (that encompasses a head switch.)** A cylinder switch time is measured in the same way as a head switch. **The approximate cylinder switch time for the hard drive being tested is 2.3ms (Table 2).**

Distance (Sector Offset)	Time (ms)	Distance (Sector Offset)	Time (ms)
434	2.419	445	2.509
435	2.419	446	2.509
436	***4.794 (C)***	**447 (H)**	**4.166**
437	2.42	448	2.509
438	2.464	449	2.509
439	2.464	**450 (H)**	**4.167**
440	**4.122 (H)**	451	2.509
441	2.464	452	2.554
442	2.464	***453 (C)***	***4.838***
443	**4.167 (H)**	454	2.554
444	2.464	455	2.554

Table 2

* - (C) denotes Cylinder switch and (H) denotes a head switch.

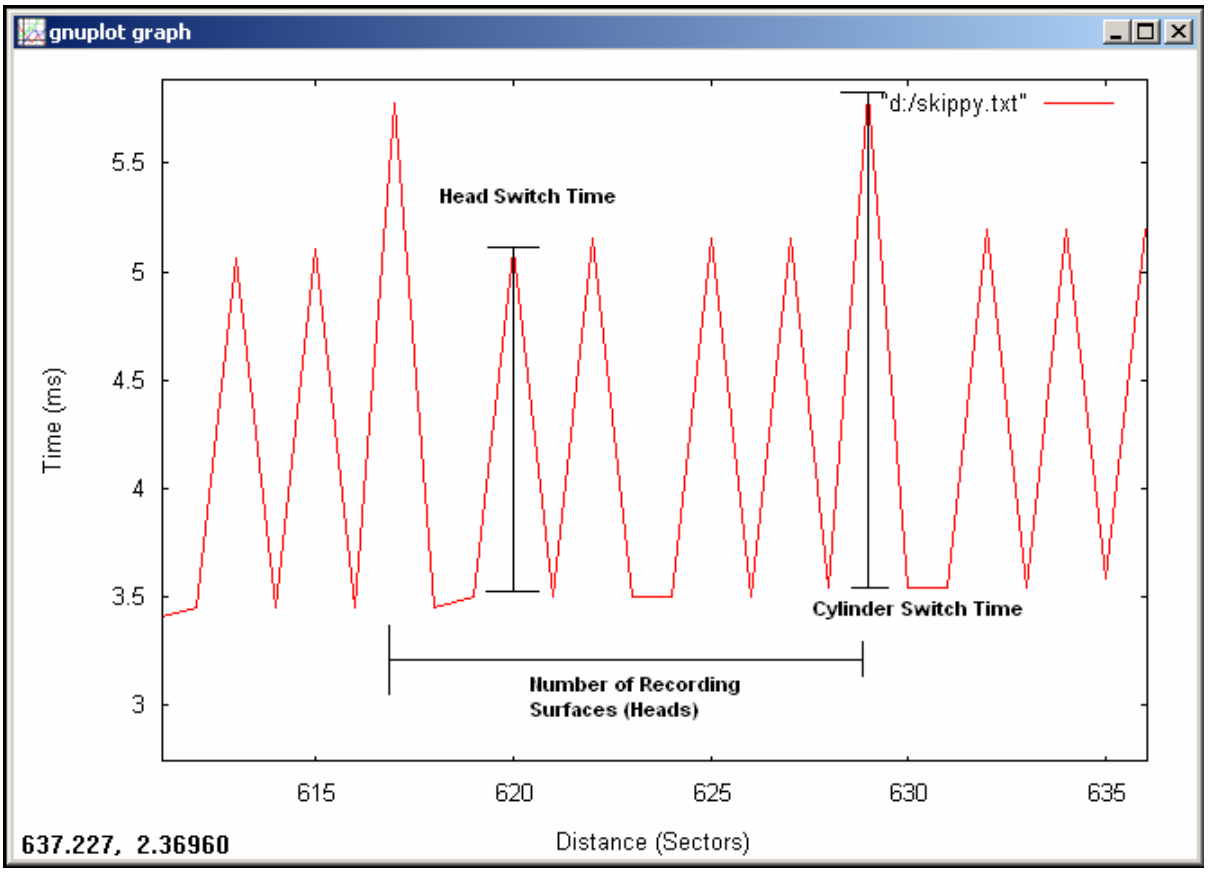

Figure 2

The final metric of the hard drive that can be determine from examining the SKIPPY algorithm data is the minimum time to media plus transfer time. This calculation measures the fastest possible time the hard drive can seek a sector and write to it. **The minimum time to media plus transfer time for this hard drive is approximately 2.1ms.**

2) HOPPY

Head switches and Cylinder switches are *not* readily apparent in the HOPPY algorithm graph. Unfortunately, HOPPY covers far fewer sectors than SKIPPY does. Over 2000 iterations, SKIPPY covers approximately 2 million sectors while in the same number of iterations, HOPPY only covers 2000 sectors. As such, HOPPY is iterated through 40,000 times to show head and cylinder switches.

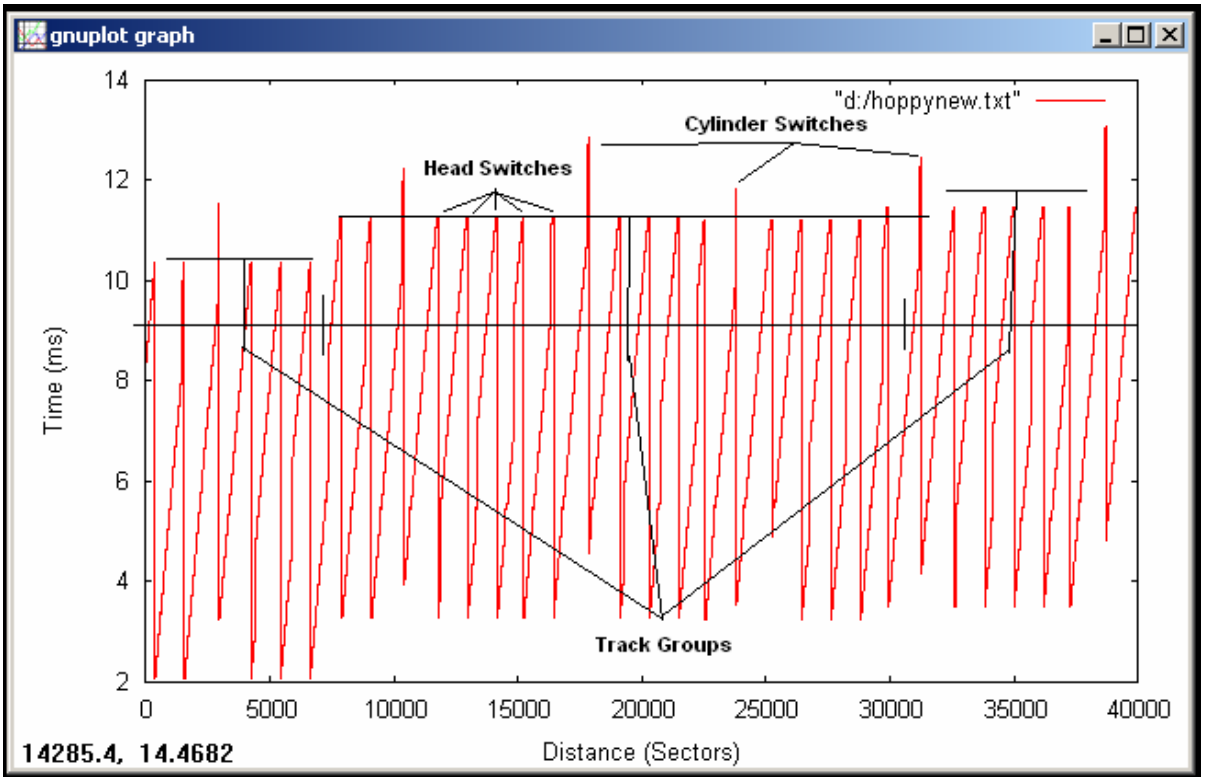

Figure 3

Figure 3 shows labeled head switches and cylinder switches that are inconsistent with the conclusions drawn in SKIPPY. The figure shows 5 head switch peaks between cylinder switches, which is inconsistent with SKIPPY conclusions. The reason for this is that HOPPY incurs variable rotational latency and it makes head switch and cylinder switch delineations somewhat difficult.

However, track groups can be vaguely identified. Figure 3 shows 3 different track groups that are differentiated by different time values. The first track group has a lower set of time values than the 2nd and 3rd track groups. Given a traversal of more sectors, more track groups can be found but creating output files for these take exorbitant amounts of time.

3) PLOPPY

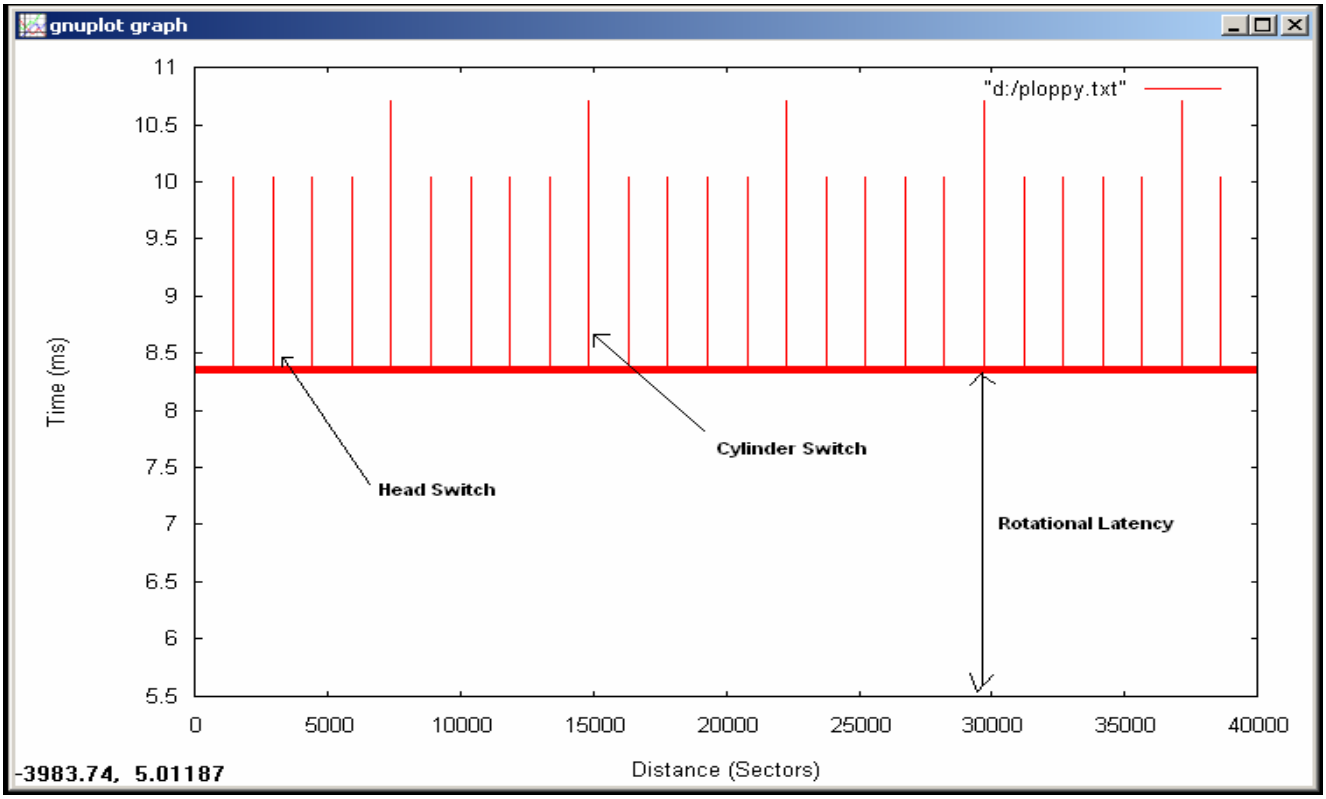

Figure 4

The PLOPPY algorithm allows very easy identification of head switches, cylinder switches and rotational latency, as shown in Figure 4. The reason this algorithm is much more effective than HOPPY in identifying these disk geometries is the starting point. HOPPY uses a 0 based system that always seeks sectors starting from 0. PLOPPY, on the other hand, seeks from the previously written sector. This allows a clean delineation in seek/write times between adjacent sectors on the same track/cylinders and adjacent sectors on different tracks/cylinders.

Since consecutive sectors are being written, the rotational latency remains largely the same. In addition to being relatively constant, the rotational latency is very high since each and every write incurs almost full rotational latency. Specific calculations can be made for both head switches and cylinder switches. The points identified on the graph can be subtracted from the constant rotational latency (~8.33 ms) to obtain the head switch time (~1.66ms) and cylinder switch time (~2.3ms).

4) SLOPPY

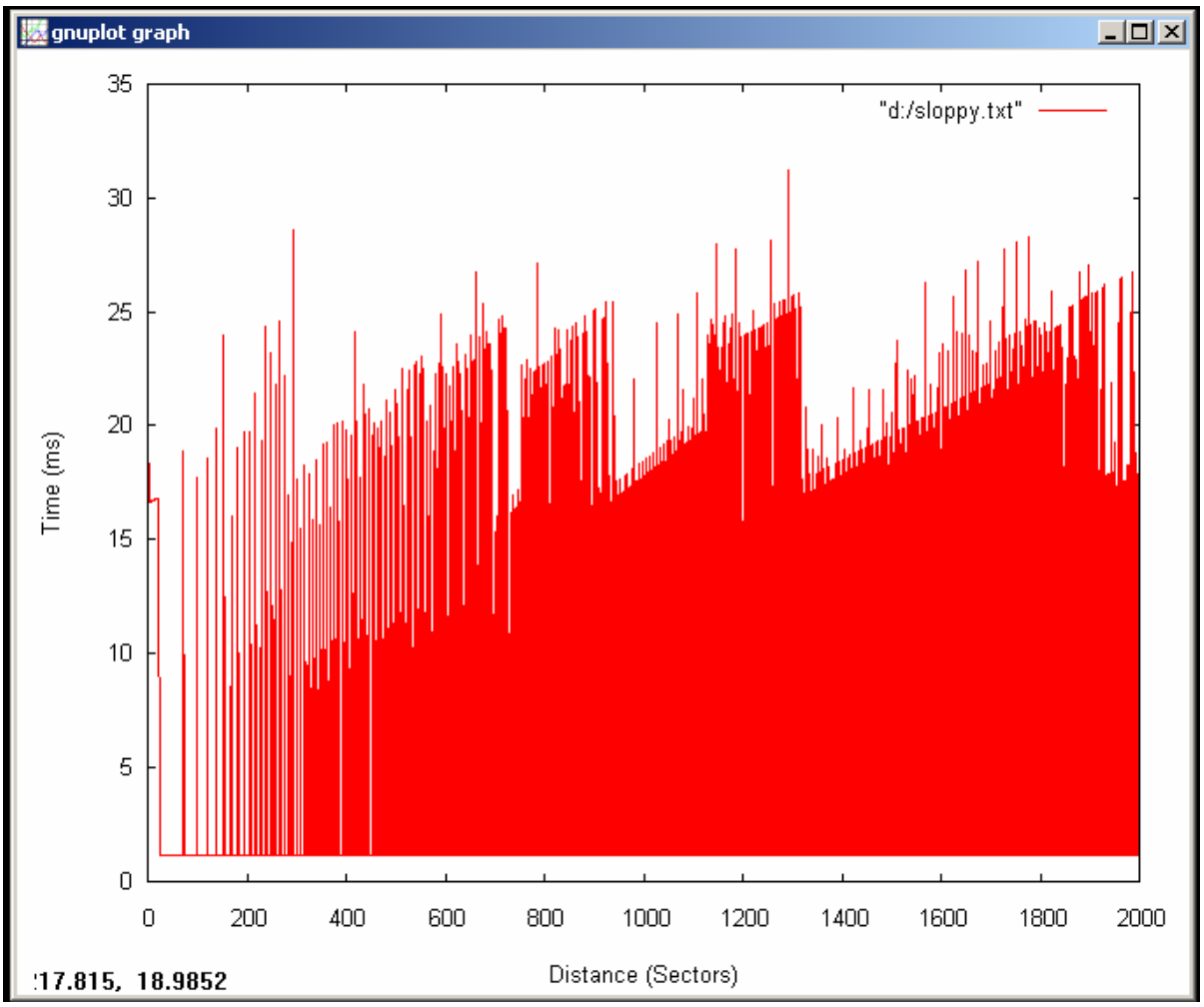

Figure 5

The results for this algorithm were quite unexpected. The results are not at all similar to the original write-based SKIPPY. The graph is composed a series of nearly instant (1.15 ms) reads followed by a long read. As the algorithm progresses, the density of the 1.15ms reads decrease drastically. At the end of the disk, all 1.15ms reads are gone and the entire array of reads is ~20 ms.

The general trend is an increase in time of seeks and writes, as the algorithm gets closer and closer to the center of the hard disk. Track zones are vaguely viewable and can be identified by sharp increases and decreases in time.

It was difficult to find any specific trends in this graph. However, after about sitting here for 6 hours staring at graphs and tables, a slight correlation was noted. Every time there is cylinder and head switch on SKIPPY, within three or so sectors, there is also a spike in the read time on the SLOPPY graph. The significance of this correlation is not totally known, but this does indicate that read times do jump in the vicinity of head or cylinder switches.