CMU 18–746 **Storage Systems**
Spring 2010 **Solutions to Homework 2**

Problem 1 : Write-based Skippy.

(a) From the man page for lseek: "The lseek(*filedes*, *new_offset*, *whence*) function repositions the offset of the file descriptor to the argument offset, according to the directive whence." That is, lseek simply changes the current offset into the file—it does *not* move the hard disk head. The argument *whence* can be either SEEK_SET, SEEK_CUR, or SEEK_END. SEEK_SET sets the file descriptor's offset to *new_offset* bytes. SEEK_CUR sets the file descriptor's offset to the current offset plus *new_offset* bytes. SEEK_END sets the file descriptor's offset to the end of the file.

(b) See Figure 1

(c) If the disk were replaced one that has a higher rotation speed, the following changes would be observed:

- The value observed for rotational latency would decrease
- Since MTM is not determined by the disk hardware, but by the software and firmware layers above, it would not change. However, STM will increase, since the disk is now spinning faster, and so the number of sectors that pass by within the MTM will increase.
- The slope of the graph would decrease
- Since the head switch time and cylinder switch times are properties of the disk head, it is likely that these values will not change.

(d) The hard disk must be more careful about head placement when performing WRITEs as opposed to READs. Hence, it is likely that MTM would decrease if WRITEs were replaced with READs.

Problem 2 : Skippy variant.

(a) The algorithm records the time necessary to write increasingly distant sectors from sector 0—specifically, the pattern written is: $0 \rightarrow 0$, $0 \rightarrow 1$, $0 \rightarrow 2$, etc. For this problem, we will refer to the second sector written in each iteration of the loop as the destination sector and the distance between sector 0 and the destination sector, the hop size. Initially, the hop size smaller than STM, so there are two parts to the observed latency: a rotational latency and the time required for the destination sector to pass under the write head after sector 0 has done so. The first minima occurs at the STM; the corresponding latency here is the MTM. The length of the angled portion of the graphs represent the number of sectors/track. The vertical jumps in the graph occur due to track and cylinder skew and hence represent the head/cylinder switch time. Finally, other minima in the graph occur when the destination sector is on a different track or cylinder than sector 0 and, due to track skw, is offset perfectly, so that it is immediately under the disk head after the head or cylinder switch is performed.

(b) Please see Figure 2

(c) Please see Figure 2

Problem 3 : Block mapping.

(a) 12 blocks $\times$ 4 KB/block = 49,152 bytes = 48 KB.

(b) There are (4 KB / 32 bits) = 1024 block pointers in each indirect block. Therefore the single indirect block adds (1024 blocks $\times$ 4 KB/block) = 4,194,304 bytes = 4 MB, for a total of 4,243,456 bytes (4.047 MB).

(c) The double indirect block adds (1024 $\times$ 1024 $\times$ 4 KB) = 4,294,967,296 bytes = 4 GB, for a total of 4,299,210,752 bytes (4.004 GB).

(d) The triple indirect block adds (1024 $\times$ 1024 $\times$ 1024 $\times$ 4 KB) = 4,398,046,511,104 bytes = 4 TB, for a total of 4,402,345,721,856 bytes (4100 GB).

(e) Now there are (1 KB / 32 bits) = 256 block pointers in each indirect block. So:

- Direct: 12 blocks $\times$ 1 KB/block = 12,288 bytes = 12 KB;
- Single: 256 $\times$ 1 KB = 262,144 bytes = 256 KB;
- Double: 256 $\times$ 256 $\times$ 1 KB = 67,108,864 bytes = 64 MB;
- Triple: 256 $\times$ 256 $\times$ 256 $\times$ 1 KB = 17,179,869,184 bytes = 16 GB;

For a total of 17,247,252,480 bytes (16.06 GB).

(f) The quadruple indirect block adds (1024 $\times$ 1024 $\times$ 1024 $\times$ 1024 $\times$ 4 KB) = 4,503,599,627,370,496 bytes = 4 PB, for a total of 4,508,001,973,092,352 bytes (4,198,404 GB).

(g) Because the block pointers are unsigned 32-bit values, you are limited to addressing only 2^{32} unique blocks. This gives a maximum file size of $2^{32} \times$ 4 KB = 17,592,186,044,416 bytes = 16,384 GB = 16 TB.

(g) Signed 32-bit numbers can address up to 2^{31} = 2,147,483,648 bytes (2 GB). Note that these parameters address byte offsets, not block offsets.

Problem 4 : Extents.

(a) 32 bits for the block addresses means the file system is limited to 2^{32} blocks, or a maximum of 2 TB. This could be addressed with a single extent.

(a – Alternate) Assuming you don't need to address blocks other than the first block in an extent, you could use two extents: one of length $2^{32} - 1$ starting at block 0, and the other with length 2^{32} blocks starting at block $2^{32} - 1$, for a total of 4 TB$-$1.

(b) If each extent is only one block long (e.g., a highly fragmented file), the minimum size is (5 $\times$ 512 B) + 1 byte = 2561 bytes.

(c) Yes. As shown in (b), when disks are highly fragmented, the OS may be forced to allocate very short extents. It would then be necessary to use indirect blocks (i.e., pointers to extent lists) for large files.

Problem 5 : Those that do not learn from the mistakes of history are doomed to repeat them.

(a) 1024 cylinders * 256 heads/cylinder * 63 sectors/head = 16,515,072 sectors = 7.875 GB.

(b) Another way of looking at the table is:

Standard	Maximum cylinders	Maximum heads	Maximum sectors
ATA	65,536	**16**	255
Int 13	**1024**	256	**63**
Combined	**1024**	**16**	**63**

1024 cylinders * 16 heads/cylinder * 63 sectors/head = 1,032,192 sectors = 0.492 GB.

(c) Because 28 address bits are available: 2^{28} = 268,435,456 sectors = 128 GB.

(d) Set up the following equations, where t is the number of 18-month periods to reach 128 GB capacity:

$$\begin{aligned} C \cdot 2^0 &= 76.335 \\ C \cdot 2^t &= 128 \end{aligned}$$

$C = 76.335$, so we solve for t using logarithms:

$$\begin{aligned} 76.335 \cdot 2^t &= 128 \\ \log_2\left(2^t\right) &= \log_2\left(\frac{128}{76.335}\right) \\ t &= \frac{\ln 1.6768}{\ln 2} \\ t &= 0.75 \end{aligned}$$

Therefore, the industry will reach the addressable limit (0.75 * 12 months) = 9 months after August 2000: May 2001.

(e) 2^{48} = 281,474,976,710,656 sectors = 134,217,728 GB = 128 PB.

(f) Set up the equations:

$$\begin{aligned} 76.335 \cdot 2^t &= 134,217,728 \\ \log_2\left(2^t\right) &= \log_2\left(\frac{134,217,728}{76.335}\right) \\ t &= \frac{\ln 1,758,272}{\ln 2} \\ t &= 20.75 \end{aligned}$$

Therefore, the industry will surpass the addressible limit of the 48-bit ATA LBA extentsions (20.75 * 12 months) = 249 months after August 2000: May 2021.

(g) 2^{64} = 18,446,744,073,709,551,616 sectors = 8,796,093,022,208 GB = 8 ZB (no kidding).

(h) Set up the equations:

$$
\begin{aligned}
76.335 \cdot 2^t &= 8,796,093,022,208 \\
\log_2\left(2^t\right) &= \log_2\left(\frac{8,796,093,022,208}{76.335}\right) \\
t &= \frac{\ln 1,152,30,143,737}{\ln 2} \\
t &= 36.75
\end{aligned}
$$

Therefore, the industry will surpass the addressible limit of the INT 13 extensions (36.75 * 12 months) = 441 months after August 2000: May 2037.

The disk drive sizes in this problem may seem ludicrous to you, but imagine how ludicrous an 80 GB hard drive must have seemed in 1981. Who knows—in ten years *you* may be sitting on a standards board that's evaluating options to extend the ATA LBA specification...

Problem 6 : I/O System Design.

(a) IOPS for CPU: $\frac{3000\text{MIPS}}{10{,}000} = 300,000$ IOPS

IOPS for Mem: $\frac{\frac{1}{50\text{ns}} \cdot 16}{8\text{KB}} = 39,062$ IOPS

IOPS for I/O bus: $\frac{33\text{MHz} \cdot 32\text{bit}}{8\text{KB}} = 16,113$ IOPS

IOPS for a SCSI controllers: $\frac{1}{.2\text{ms} + \frac{8\text{KB}}{320\text{MB/s}}} = \frac{1}{.2\text{ms}} = 5,000$ IOPS

IOPS for a disk: $\frac{1}{6\text{ms} + \frac{.5}{10{,}000\text{RPM}} + \frac{8\text{KB}}{25\text{MB/s}}} = \frac{1}{6\text{ms}+3\text{ms}+.3\text{ms}} = \frac{1}{9.3\text{ms}} = 108$ IOPS

(b) Fully configured system = 1 CPU, 1 PCI Bus, 8 Controllers with 7 drives each.
7 Drives = $7 \cdot 108$ = 756 IOPS
8 Controllers = (with 7 of current drives) $7 \cdot 756$ = 6048 IOPS, max = $7 \cdot 3,356$ = 23,489 IOPS
So with current drives it seems that the drives are the bottleneck, everything else is underutilized. But it the I/O bus can not handle the controllers at their maximum throughput – so the bus will become a bottleneck.

(c) To maximize performance we need as many disks as possible, since they are the major bottleneck. So if we had 56 disks (8 controllers * 7 disks) we would have a total of $56 \cdot 108$ IOPS$= 6,048$ IOPS. We know that a single controller can easily handle 7 drives and since $6,048 < 6,113$ (the max of the PCI bus), our system can handle this load.
The (somewhat unreasonable) cost of this system is:
$8 \cdot \$350 + 56 \cdot (36 \cdot \$4) = \$2,800.00 + 56 \cdot \$144 = \$10,864.00$.

(d) IOPS for new disks: $\frac{1}{4\text{ms} + \frac{.5}{15{,}000\text{RPM}} + \frac{8\text{KB}}{30\text{MB/s}}} = \frac{1}{4\text{ms}+2\text{ms}+.26\text{ms}} = \frac{1}{6.26\text{ms}} = 160$ IOPS

IOPS for new bus: $\frac{66\text{MHz} \cdot 64\text{bit}}{8\text{KB}} = 64,453$ IOPS

IOPS for new CPU: (I messed up and used 1000MIPS instead of 10,000MIPS, there's no way a slower processor would be a good idea unless it might be cheaper)

For the system in (c) the new disks would be most beneficial.

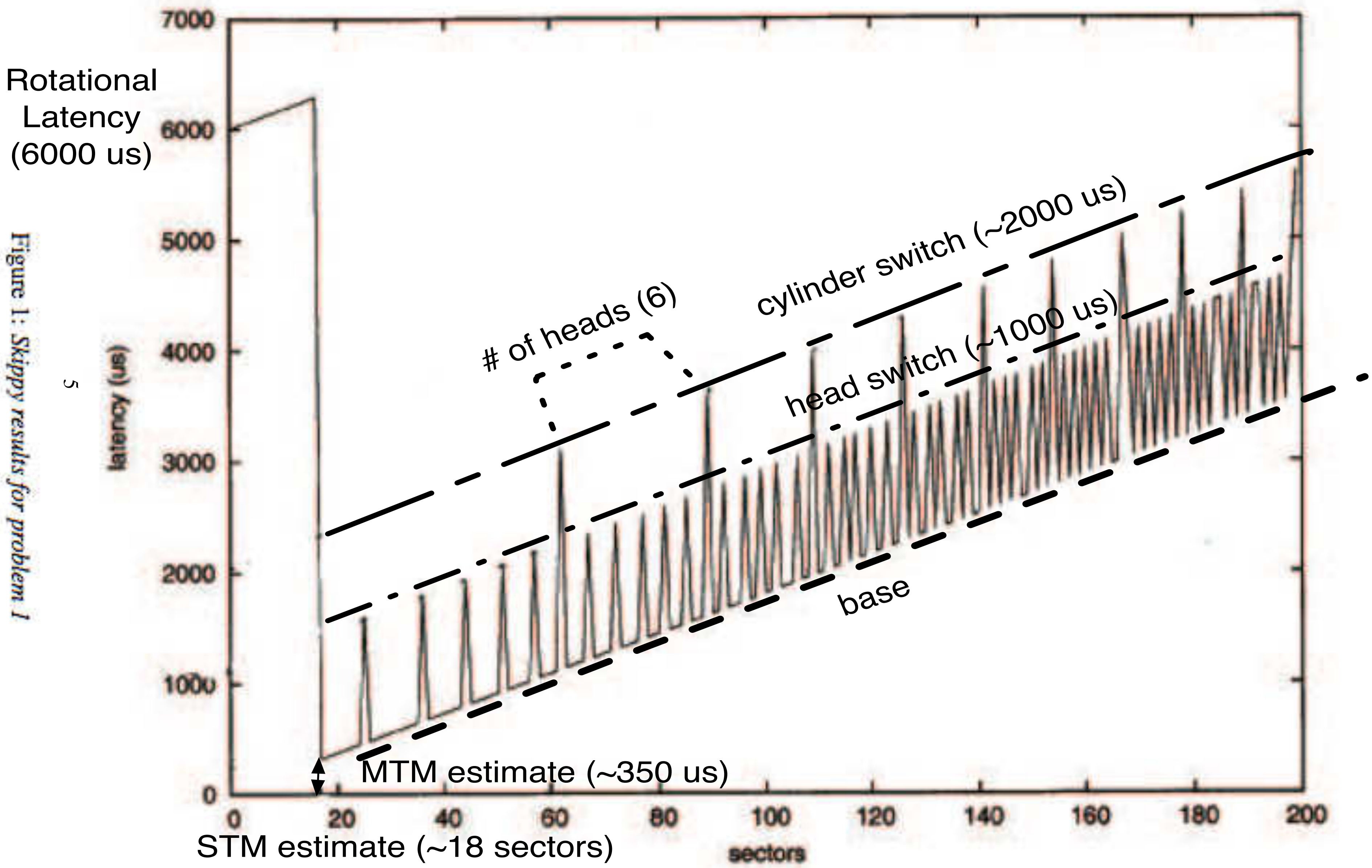
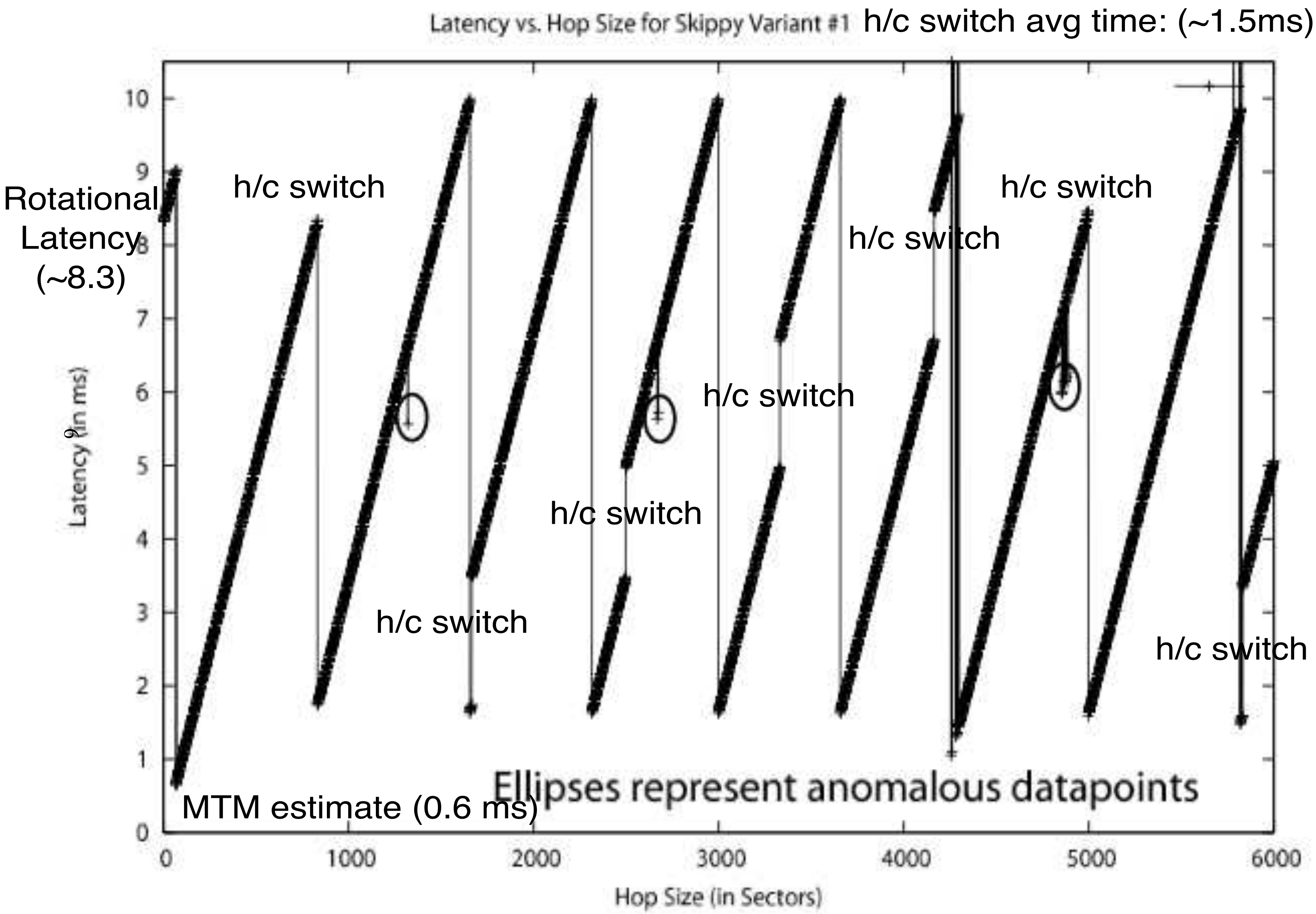


Figure 1: *Skippy results for problem 1*



Latency vs. Hop Size for Skippy Variant #1
h/c switch avg time: (~1.5ms)
Rotational Latency (~8.3)
h/c switch
h/c switch
h/c switch
h/c switch
h/c switch
h/c switch
h/c switch
h/c switch
Latency (in ms)
Hop Size (in Sectors)
Ellipses represent anomalous datapoints
MTM estimate (0.6 ms)
9