CMU 18-746 **Storage Systems** **Assigned: 31st Aug. 2006**
Fall 2006 **Project 1** **Due: 3d Oct. 2006**

Overview: In this project you will probe the ordinary OS-level block storage device interface in order to determine the low-level behavioral characteristics of an attached disk drive.

To prepare for this project, read *Microbenchmark-based Extraction of Local and Global Disk Characteristics* by Nisha Talagala, Remzi H. Arpaci-Dusseau and David Patterson (Technical Report CSD-99-1063, University of California, Berkeley, 1999). Using the methods described in the paper, you will verify a subset of their experimental results and perform several additional experiments not previously published.

We recommend that you run your experiments on a real disk or disk partition. This will require that you have administrative access to a Unix-like system (Linux, FreeBSD, Solaris, etc.) with at least one available disk partition to be overwritten (for example, a swap partition that can be disabled for the duration of your experiments & reformatted at the conclusion). This disk may use any interconnect: SCSI, ATA, Fibre Channel, etc. Your friendly T.A. is available if you have questions about setting up your experimental environment.

If you do not have access to such a system (or you have trouble getting acceptable experimental results) a disk emulator package specifically created for this project is available from the course Web pages. The emulator is built using the DiskSim Simulation Environment and models the Seagate Barracuda ST32171W, which is similar to one of the experimental drives from Talagala *et al.* There is a README file in the emulator package that will help get you started. The library included in the package was compiled under Solaris; to compile your own progam and link against the library you'll need to have access to a Solaris-based machine (for example, `sun4.andrew.cmu.edu`).

This is an individual project. You are not permitted to collaborate with anybody, and all work and coding must be your own. All questions should be directed to the T.A. *Exception:* you may ask anyone for help setting up your experimental environment, verifying that you are not putting any data at risk, etc. You may also share an experimental machine and disk partition with a classmate, as long as you do not discuss the project or share code.

Deliverables: All deliverables are due at the beginning of class on the assigned date. Late submissions will not be accepted, and no extensions will be granted. You will submit a report containing (1) a description of the problem and your experimental environment, (2) a description of each experiment and your results, including any graphs you've generated, and (3) a conclusion describing what you've learned. This report must be typed and neatly organized, and graphs must be clearly labeled. In addition to the report, you will email a copy of your code to the T.A.

Experiments:

(a) Write-based SKIPPY: Reimplement the write-based SKIPPY and obtain results similar to those presented in (the upper part of) the figures in Section 5 of Talagala *et al.* Create a graph with your results & a table to report values for rotational latency, MTM, sectors per track, number of heads, head switch time, and cylinder switch time. Look up the manufacturer's published specifications for the disk you're using (not all values will be available) and report any percent error in your experiment. Explain what you believe to be the source of this error.

(b) Write-based SKIPPY variant #1: Modify your code above as follows. Choose a starting sector. Perform a small write to this sector, then incrementally hop to subsequent sectors and perform a small write. For each data point, begin with a write to the original starting sector.

Algorithm: $\text{hop}(i) = 0 \rightarrow i \mid i \in \{1, 2, 3, \ldots, n\}$

Example: $0 \rightarrow 1, 0 \rightarrow 2, 0 \rightarrow 3, \ldots, 0 \rightarrow n$

Create a graph with your results. Perform a similar analysis to the SKIPPY graphs and attempt to abstract the physical characteristics of the disk drive: where are the head switches, cylinder switches, and other artifacts of disk geometry?

(c) Write-based SKIPPY variant #2: Modify your code above as follows. Choose a starting sector. From this sector, measure the time to write the subsequent sector. Repeat this for the sector you've just written; do not return to the original sector.

Algorithm: $\text{hop}(i) = (i-1) \rightarrow i \mid i \in \{1, 2, 3, \ldots, n\}$

Example: $0 \rightarrow 1, 1 \rightarrow 2, 2 \rightarrow 3, \ldots, (n-1) \rightarrow n$

Create a graph with your results. Perform a similar analysis to the SKIPPY graphs and attempt to abstract the physical characteristics of the disk drive: where are the head switches, cylinder switches, and other artifacts of disk geometry?

(d) Read-based SKIPPY. Repeat the first SKIPPY experiment but use small reads instead of small writes. Create a graph with your results. Are you able to extract the same information as from the write-based SKIPPY? To what do you attribute any differences?

(e) ZONED: Reimplement the read-based ZONED and obtain results similar to those presented in Figure 10 of Talagala *et al.* (Note: you should perform this test over the entire disk, not just the disk partition you used above). Create a graph with your results. Compare your results for maximum and minimum bandwidth to the manufacturer's published specifications. To what do you attribute any differences?

Suggestions: Here is an example of how you might set up the experimental environment on a Linux system. Note: your disk labels and partition numbers may vary; if you're not sure what you're doing, ask somebody before you risk irreparable loss of data!

- su to root

  ```
  su
  ```

- Have an empty (and I mean EMPTY!) partition.

  ```
  for the sake of argument say /dev/hda6
  ```

- Create a character device driver that links to that partition.

  ```
  raw /dev/raw/raw1 /dev/hda6
  ```

- Turn off write-caching and read optimizations on this disk

  ```
  hdparm -W 0 /dev/hda6
  hdparm -A 0 /dev/hda6
  ```

The remaining steps must be taken inside your program.

- Create a buffer out of which to do I/O

  ```
  char buf [1023];
  ```

- Open the raw device

  ```
  rawdevh = open ( "/dev/raw/raw1", O_RDWR);
  ```

- Find the byte offset in this buffer that is memory aligned, i.e., offset 0, 512, 1024, 1536, etc. in a page.

  ```
  for ( i=0; i<512; i++ ) {
    rc = read ( rawdevh, buf+i, 512 );
    if ( rc != -1 ) {
      rawoffset = i;
      break;
    }
  }
  ```

- Do all subsequent I/O aligned to that offset.

  ```
  rc = write ( rawdevh, (void *) buf + rawoffset, 512 );
  ```