CS 162 Section 7

True/False

1. With SSD's, writing data is straightforward and fast, whereas reading data is complex and slow.

2. User applications have to deal with the notion of file blocks, whereas operating systems deal with the finer grained notion of disk sectors.

3. Directories in UNIX are basically files with pointers to inodes in them.

Short Answer

1. What are the major components of disk latency? Explain each one.

2. Indirect blocks look a lot like another data structure we looked at earlier this semester. What is the other data structure? Can you compare the tradeoffs you make in either case?

3. What are the main challenges of distributed systems (specifically, distributed hash tables)?

4. What are their advantages and disadvantages of iterative vs. recursive KV-store queries in terms of performance?

Longer Answer

1. How many times is the disk accessed when you type “ls /home/cs162”?

   Assuming the total response time to read an inode or block from disk is 5ms, and all inodes and directories consume one block, how long does this take?

   Now say the following is true::
   - The root directory listing is cached in memory.
   - The disk controller is using a track buffer.
   - I-nodes are always stored on the same cylinder as the blocks they refer to.

   Assume that data in the track buffer or in memory is “free” (e.g. 0ms) to access. Now much time would it take to read the contents of “/home/cs162”?

2. In class we said that the operating system deals with bad or corrupted sectors. Some disk controllers magically hide failing sectors and re-map to “back-up” locations on disk when a sector fails.

   If you had to choose where to lay out these “back-up” sectors on disk - where would you put them? Why?

   How do you think that the disk controller can check whether a sector has gone bad?

   Can you think of any drawbacks of hiding errors like this from the operating system?