Preparation for Recitation 17

Read *The Design and Implementation of a Log-Structured File System*.

In today's reading, the authors depart radically from UNIX File System (presented in the UNIX paper and in Section 2.5 of the textbook) and propose, instead, a file system in which all of the data is stored in *an append-only log*, that is, a flat file that can be modified only by having data added to the end of it. In Chapter 9, we also hear about logs, specifically how they help achieve *reliability*. For today's reading, the purpose of the log is to achieve good *performance*.

To understand why an append-only log might lead to high performance, we quickly review the basics of disks (which you should recall from class). Data on a disk is stored on a constantly spinning platter and read by a head that floats above the platter. The platter is broken down into concentric rings called tracks; each track is divided into sectors. The time required to read or write data to a disk can be broken down into three components: moving the head to the correct track, waiting for the correct sector to rotate under the head, and transferring the data off the disk. The first two operations (known collectively as a "seek") are very expensive (and aren't getting cheaper very fast): together, they can take 10 or more milliseconds. Once the disk head is at the correct location, it can transfer data relatively quickly (at around 40 MB/s on today's high-end disks or 10 MB/s on lower-quality disks).

Now, given this description, the goal of a log-structured file system is to minimize seeks by treating the disk as an infinite append-only log. For example, the file system software simply appends new files to the end of the log. For this strategy to translate into higher performance, the file system software must:

- Write in large batches to amortize the cost of a seek (if any).
- Always start writing from the last place it stopped writing.
- Avoid reading; reads are likely to cause a seek.

As you read the paper, consider how the authors achieve these three goals. Keep in mind that the "infinite log" is actually on a finite disk and ask how that constraint affects their goals. Also, ask yourself whether certain file access patterns by applications might make it hard for a log-structured file system to avoid seeks.