Michael Salib 6.033 4/23/02

We can reliably detect, but not necessarily correct, disk seek errors on writes, by reading recently written sectors after they have been written. By rereading these sectors, we can recompute the corresponding sector checksum and verify that it matches the checksum on disk as well as the checksum we had intended to write. These two tests insure that the data on disk is consistent and that the data on disk is what we intended to write in the location we intended to write in.

The performance implications of this scheme depend on whether it is synchronous or asynchronous; if we require that errors be detected immediately after the write completes (synchronous error detection), then disk write performance will be substantially reduced since the disk must wait after every single write for the recently written track to spin around for verification. On the other hand, if we can tolerate some delay in detecting disk seek errors on writes, we can design the Operating System to schedule a data read for recently written sectors shortly after the corresponding sectors are written to. Because the OS can schedule disk requests with an elevator algorithm and can interleave write sector verification reads with other disk activity, we should see only a slight reduction in throughput.

The down side of this approach is that it forces disk seek error detection to be asynchronous; in other words, the OS won't become aware of disk seek errors for some time after the failed write operation has completed. For disk writes issued on behalf of the file system, this may not matter very much since the file system is going to persist throughout the lifetime of the system and can retry the operation repeatedly until it succeeds. Asynchronous error detection is more problematic for userland applications that are issuing raw disk writes. These applications are transient, and may well have terminated between the time when their write call successfully returns and when the kernel informs them that the write failed. Even worse, there is no clear API with which the kernel can inform applications of these failures. We can remedy this problem by having the OS refuse to return from raw disk write calls until the data has been verified, but this will dramatically increase write latency from the application's perspective.