EECS 678: Lab 9 - Memory Mapped I/O

Gehrig Keane
2727430

Thursday 21th April, 2016

The time required to copy the file using read_write varies with the size of the buffer specified. Smaller buffer sizes take longer. The time required for memmap varies much less regardless of how you perform the copy. Discuss why this is, and in your discussion consider the influence of: (1) the overhead of the read() and write() system calls and (2) the number of times the data is copied under the two methods. This question is worth 20 points.

The data-paths for the ***read*** and ***write*** system calls involve moving data from the hardware storage media (magnetic disk or solid state memory) to a cloaked kernel buffer, then to a main memory location accessible by the user, then back to a cloaked kernel buffer, and finally the data is returned back to the storage media. This to-and-fro process operates continually, but for small buffer sizes or storage intensive programmatic applications this kernel intermediary contributes towards undo redundancy. The use of ***memmap*** removes much of the incurred overhead in that the data-path from the kernel buffer to disk and back operates much more efficiently. We can essentially send data direclty to and from the disk and the kernel managed buffers.

When you use the read_write command as supplied, the size of the copied file varies with the size of the buffer specified. When you use the memmap command implemented the size of the source and destination files will match exactly. This is because there is a mistake in the read_write code. What is the mistake, and how can it be corrected?

The initial implementation of the ***read_write*** command allocates memory based on the assumption that teh file of choice is some multiple of the buffer size, ergo when this assumption fails memory is wastefully allocated. In lieu of this issue, if we use the ***read*** command to monitor the number of bytes read we can write exactly that many in return eliminating the previously wasteful behavior. This solves the filesize problem explicitly.