Wei Fei (2538810)
EECS678 Lab12 Report
Prasanth Vivekanandan
04/26/15

1. 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.

Since when the buffer sizes are small, it takes more cycles of read and write, in order to complete the whole process of copying file, which means running the higher overheads for read() and write() functions. In addition, both of two methods needs to copy data from physical disk and copy to a kernel buffer and memory location in userspace, and then copy back in order to write back to the disk. The time required for memmap method is much less because it doesn't actually copy the original data. In other words, the memmap function avoids additional data copy by directly mapping the kernel-space buffers and map the source address to the destination address.

2. 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 read_write code has a little problem is because it does not judge the condition of buffer size. If the input buffer size is smaller than actual size of original file, the system may have overhead when it doing copy process. When the loop goes to the last step, if the remaining size is not enough to sufficient input buffer size, the buffer size may over-copy from the beginning for the file, which means the size of copy file could greater than the original file.

To revise the problem, we may change the destinition file size equal to the source file size. Or we can add a temporary variable named "tempRemainSize", and then add an if-else statement to judge the condition of buffer size and remainning size.

The suducode should like this (in loop):

{

tempRemainSize = tempRemainSize – bufferSize;

```
        If (tempRemainSize > 2 * bufferSize)
        {
                continue;
        else
        {
         bufferSize = tempRemainSize;
        }
}
```