2020, Spring OS Project 2

Synchronous Virtual Device Report

R08922102 侯思岑

R08922133 李永霖

R08922100 曾勁棠

R08922119 蕭景忠

R07922165 賴建中

1. 設計

此次實驗使用 Linux Kernel 4.14.25。實驗中比較 synchronous I/O 及 memory map I/O 在 transmission time 的差異。在 user program 主要修改了 mmap 的部分。以下分兩端說明：

Master 端：

```
while (offset < file_size) {
    if((src = mmap(NULL, PAGE_SIZE, PROT_READ,
                   MAP_SHARED,file_fd, offset)) == (void *) -1) {
        perror("map input file fail");
        return 1;
    }

    do {
        int len = (offset + BUF_SIZE > file_size ? file_size % BUF_SIZE : BUF_SIZE);
        write(dev_fd, src+offset%PAGE_SIZE, len);
        offset = offset + len;
    } while (offset < file_size && offset % PAGE_SIZE != 0);

    ioctl(dev_fd, 0x12345676, (unsigned long)src);

    if(munmap(src, PAGE_SIZE)==-1){
        perror("unmap fail");
        return 1;
    }
}
```

在 master.c 中，首先會讀進並判斷使用者輸入的參數，包含有幾個 files, file 的路徑和傳輸的方法(mmap 或是 fcntl)。接著針對每一個 file 的內容進行傳輸。

在 mmap 的方法中會先將 offset 設定為 0，在每一輪 while 迴圈中，用 mmap 要一塊大小等於 page size 的記憶體，將檔案 offset 之後的資料放於這塊記憶體之中。接下來用另一個 while 迴圈，在每一輪中將這塊記憶體上 buffer size 的資料用 write 的方式寫到 master device 中，並且將 offset 加上一個 buffer 大小。用 write 的 function 會讓 master device 將 buffer 的資料傳到 slave device。這樣下去直到一個 page 的資料已經傳完了或是 offset 達到了 file 的大小。若是前者的話則先清空記憶體，並用新的記憶體開啟 offset 之後的資料，再繼續傳下去；若是後者的話則表示已經傳完整個 file 的資料，釋放記憶體之後就可以結束了。

Slave 端：

```
file_size = 0;
ftruncate(file_fd, mmap_size);
if((dst = mmap(NULL, mmap_size, PROT_READ | PROT_WRITE, MAP_SHARED,
               file_fd, 0)) == (void *) -1) {
    perror("map output file");
    return 1;
}
while ((ret = read(dev_fd, buf, sizeof(buf))) > 0)
{
    memcpy(&dst[file_size%mmap_size], buf, ret);
    file_size += ret;
    if (file_size % mmap_size == 0) {
        ioctl(dev_fd, 0x12345676, (unsigned long)dst);
        munmap(dst, mmap_size);
        ftruncate(file_fd, file_size + mmap_size);
        if((dst = mmap(NULL, mmap_size, PROT_READ | PROT_WRITE, MAP_SHARED,
                       file_fd, file_size)) == (void *) -1) {
                perror("map output file");
            return 1;
        }
    }
}
ftruncate(file_fd, file_size);
ioctl(dev_fd, 0x12345676, (unsigned long)dst);
munmap(dst, mmap_size);
```

一開始先指定 file_fd 的大小為 mmap_size，將 file_fd 用 mmap 映射到 dst。接下的 while loop 中，每次會將 dev_fd 的檔案讀進 buf，再用 memcpy 將 buf copy 到 dst[file_size % mmap_size] 中，然後更新

file_size，一直到 dev_fd 的全部讀完。 while loop 途中如果遇到 dst 配置的 mmap_size 空間都寫過了，會用 munmap 解除映射目前的 dst，增大 file_fd 的空間 mmap_size 的大小，並將新開的空間用 mmap 映射到新的 dst 中，給下一個 while loop 做 memcpy 。 最後更改 file_fd 的大小至正確的 file_size，以刪去尾部多餘的資料，並且用 munmap 歸還記憶體。

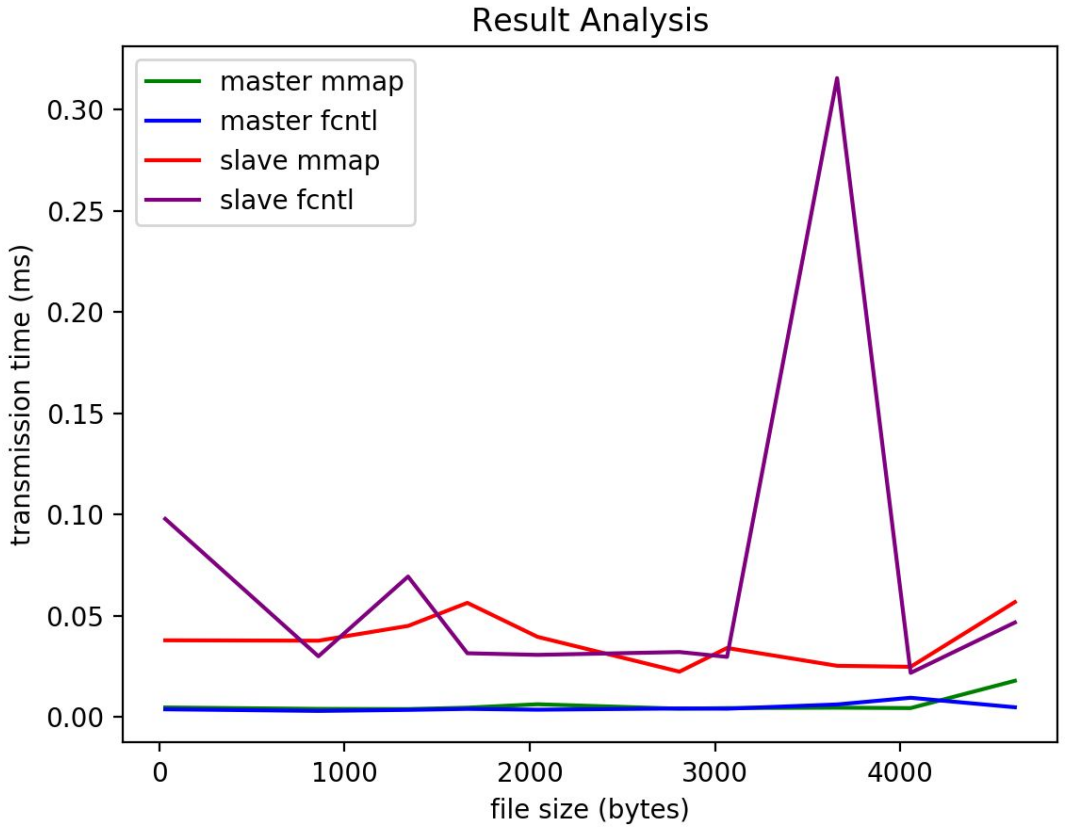
2. 執行結果

包含使用sample input 得到的transmission time結果及自己準備不同大小的測資得到的結果。數據如下：

使用sample input:

file size (bytes)	Master MMAP	Master FCNTL	Slave MMAP	slave FCNTL
32	0.0046	0.0037	0.0378	0.0978
859	0.0039	0.0029	0.0376	0.0299
1343	0.0038	0.0034	0.0449	0.0693
1663	0.0045	0.0039	0.0563	0.0314
2042	0.0062	0.0035	0.0395	0.0306
3065	0.0043	0.0040	0.034	0.0296

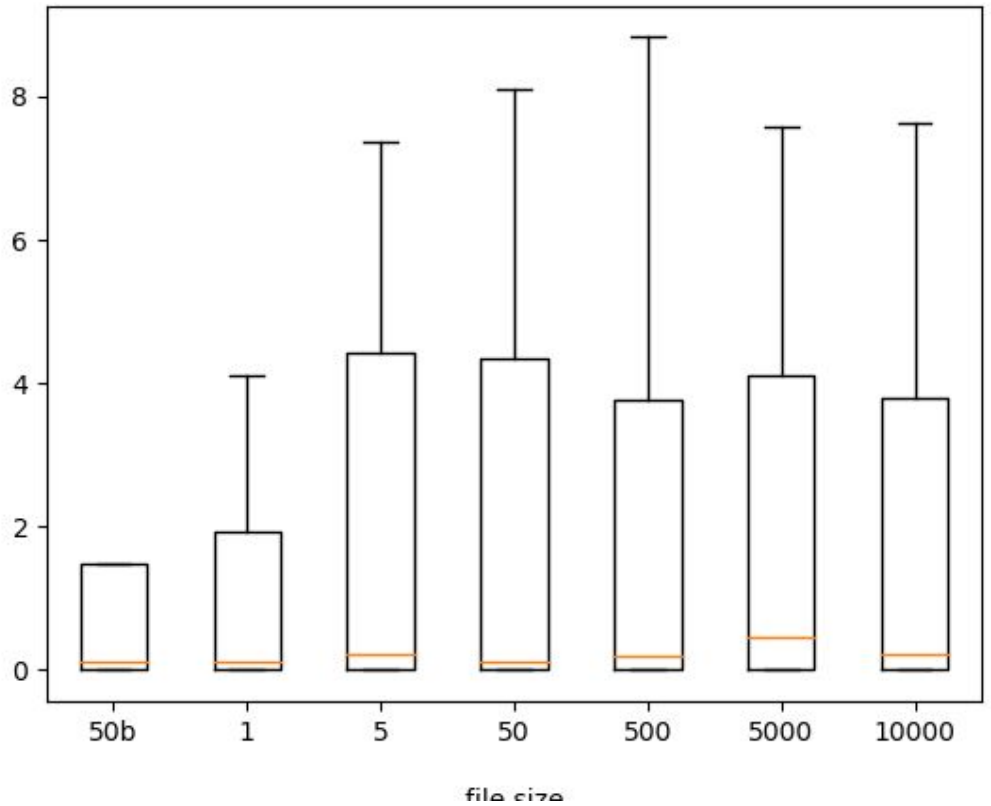
2808	0.0040	0.0041	0.0223	0.0320
4056	0.0043	0.0094	0.0247	0.0217
3659	0.0045	0.0061	0.0252	0.3156
4619	0.0178	0.0047	0.0567	0.0466



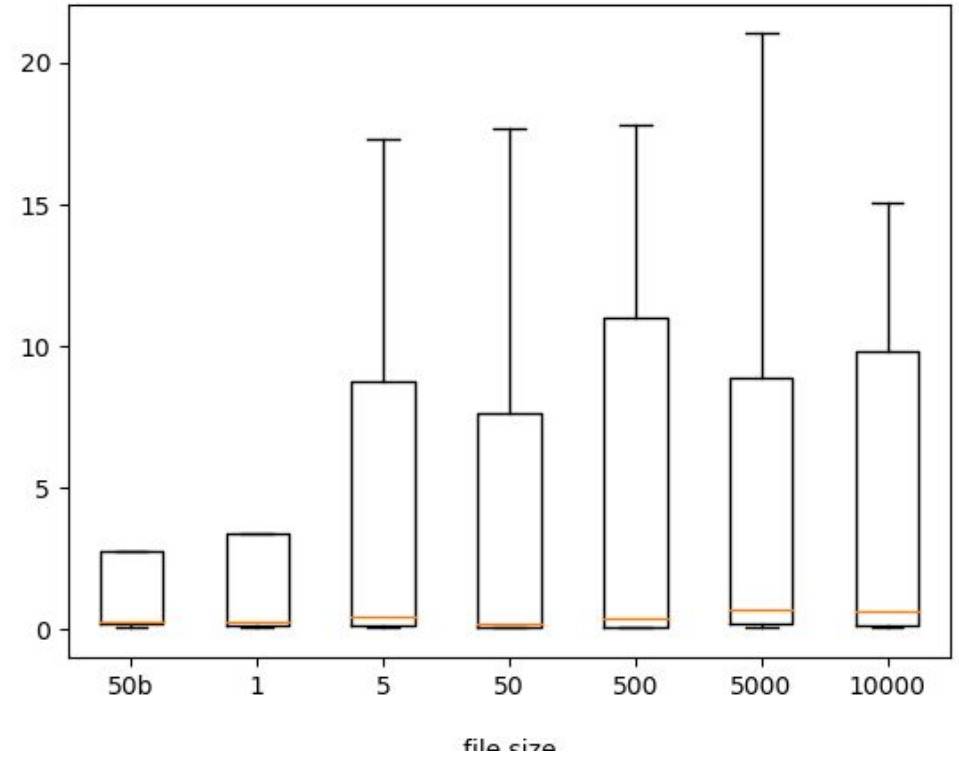
由上圖視覺化的實驗結果得知，在不考慮一些不合理極值的情形下，master端的transmission time在mmap及fcntl 並無顯著差異，相較之下，在slave端可以看出使用mmap大致上表現得比fcntl佳。

使用七種不同大小的測資（大小依序為50bytes, 1kB, 5kB, 50kB, 500kB, 5000kB, 10000kB）測試20次得到的transmission time（已刪除極值情形）：

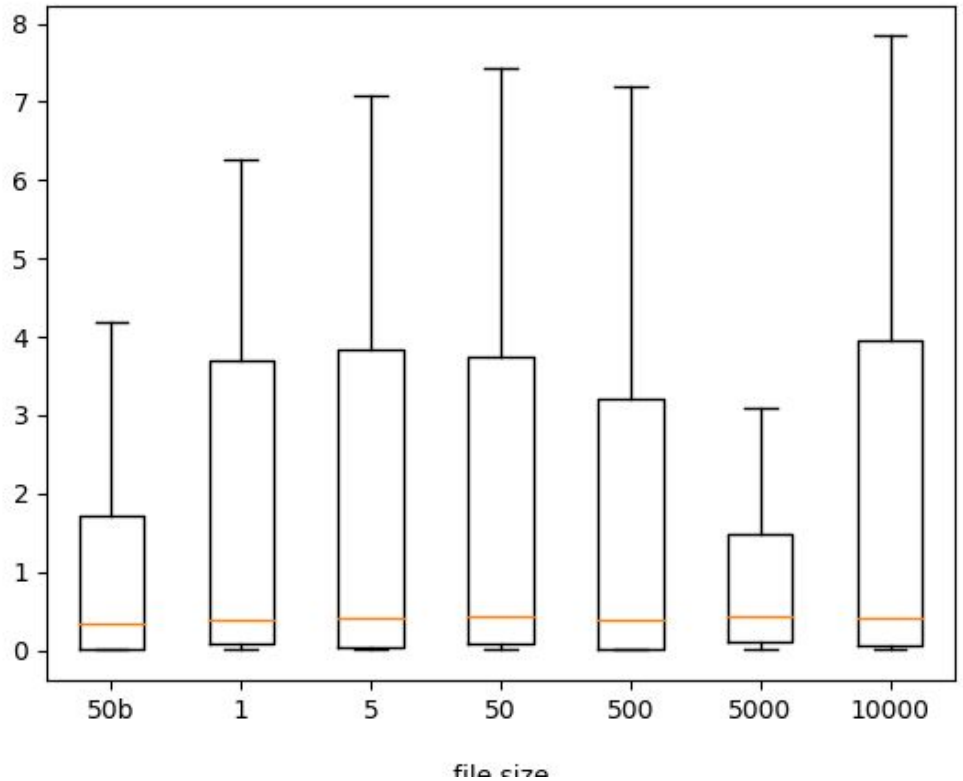
fcntl to fcntl :
(master)



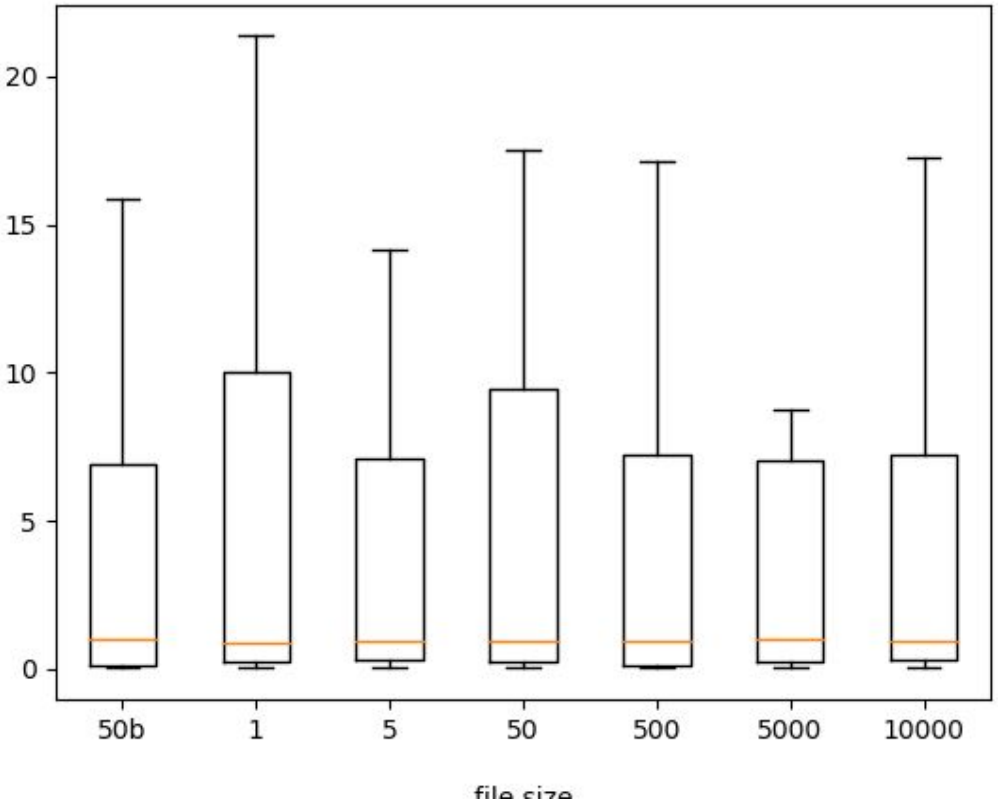
(slave)



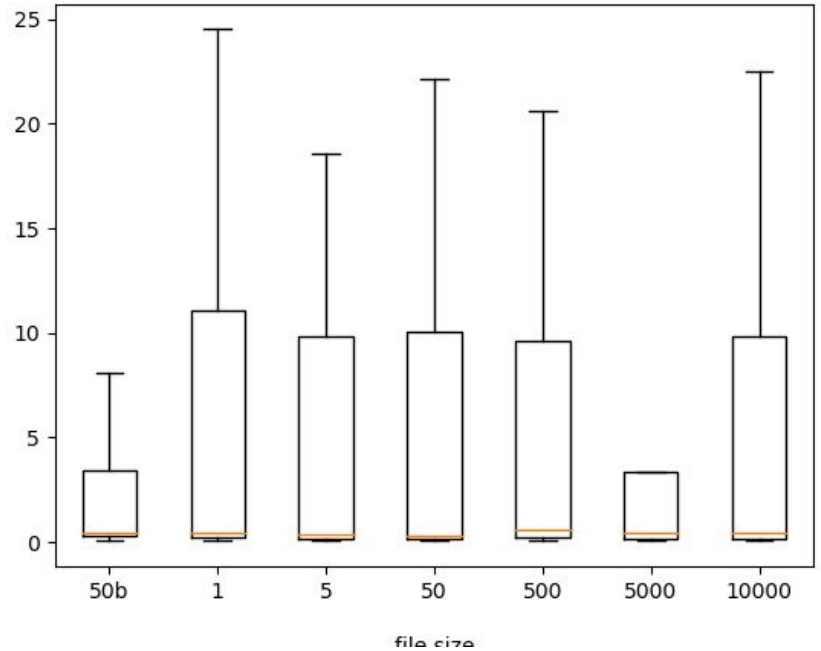
mmap to fcntl :
(master)



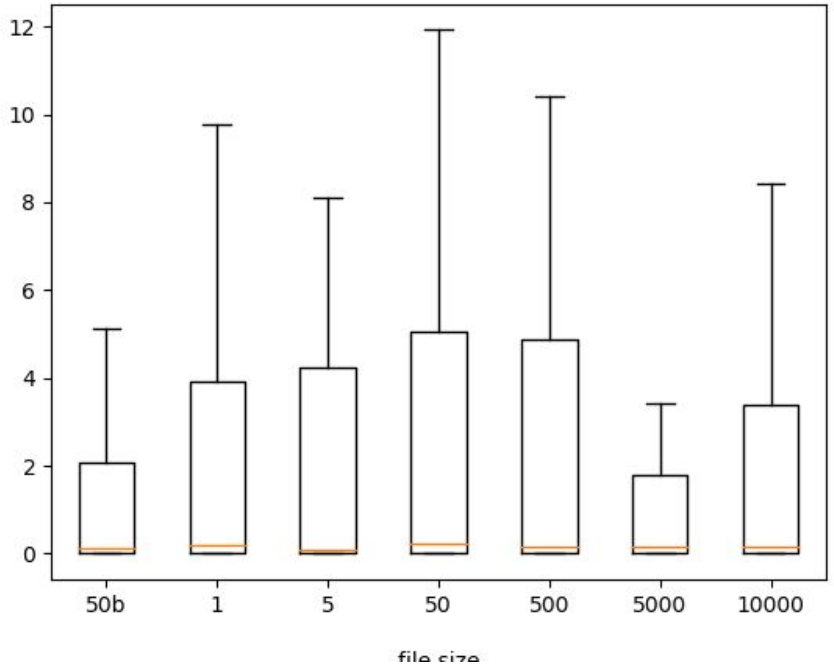
(slave)



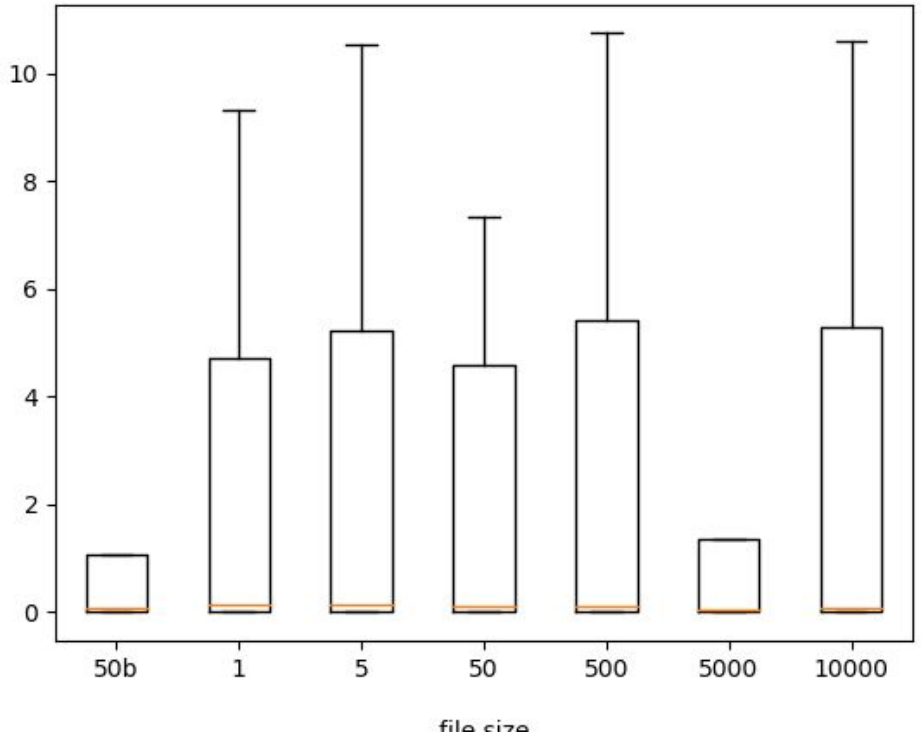
fcntl to mmap :
(master)



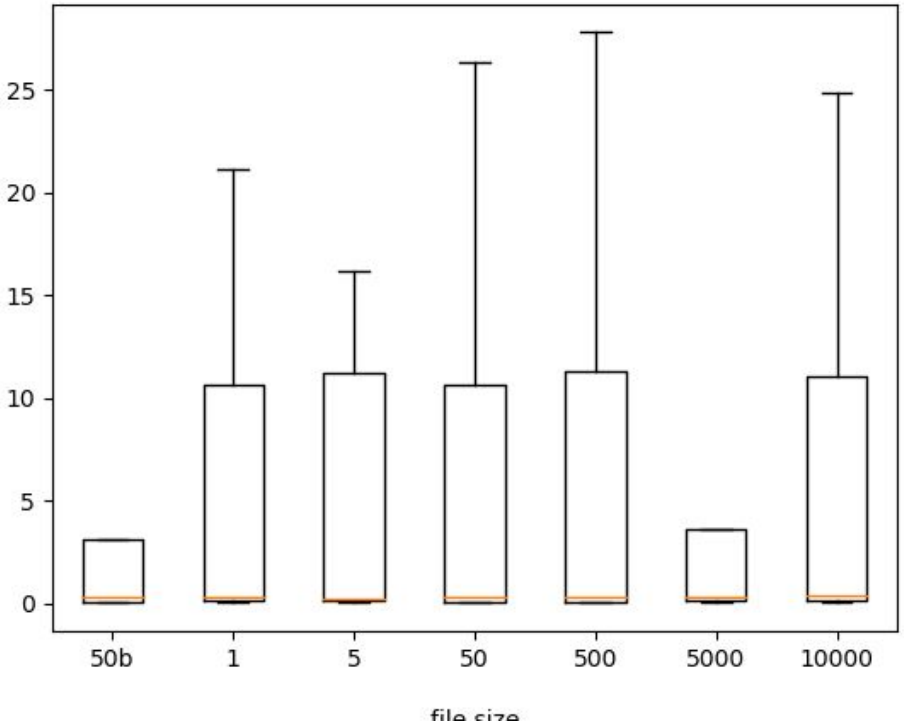
(slave)



mmap to mmap:
(master)



(slave)



3. 分析及比較不同設定下的結果

由實驗的結果大致可歸納出以下結論：當檔案大小沒有很大的時候，使用mmap及fcntl兩種方式得到的效能差異並不會太大，一方面也是因為將檔案對應到記憶體時也有其overhead，因為我們設計mmap的部分是每當一個 page size (4096 bytes)的記憶體內容完成作業之後，會歸還記憶體並再 mmap file接下來的部分，因此每當重新操作memory時會有建立page table的一些附加的成本，甚至有可能需要處理到 page fault的問題，另外我們每次都只從device中拿取buffer size(512 bytes)的大小拿出或放入記憶體，每次動作也會有overhead。
而用 fcntl 的方法當檔案大小增加時，也會增加disk IO存取次數並大幅增加overhead，所以在實驗結果上比較不出兩者差異。但當檔案大小逐漸增大時，可看出slave端得到的結果會較不平均，且與master端差異會越大。

4. 分工

R07922165 賴建中	撰寫 master user design
R08922102 侯思岑	統計實驗結果與呈現、分析資料，撰寫報告
R08922133 李永霖	撰寫 slave user program
R08922100 曾勁棠	寫 master mmap 的部分，並產出實驗數據
R08922119 蕭景忠	撰寫 slave design, 測試並修改程式碼

5. 參考

Sample code (https://drive.google.com/file/d/1rpJnDB9VRxM0vm1dGkeyyWFSko_Mf0Ek/view)

File Reading and Writing (https://www.geeksforgeeks.org/input-output-system-calls-c-create-open-close-read-write/)
mmap (https://man7.org/linux/man-pages/man2/mmap.2.html)
munmap (https://linux.die.net/man/2/munmap)
ioctl (https://www.man7.org/linux/man-pages/man2/ioctl.2.html)
ksocket (https://github.com/hbagdi/ksocket)