# Lab 10 Report

**Shouvanik Chakrabarti(130050072)**
**Krishna Harsha(130050076)**

**Part A:**

We use the command pmap -x <pid> to obtain the virtual memory dump for the running process.

1.
```
Address           Kbytes     RSS   Dirty Mode  Mapping
0000000000400000       4       4       4 r-x-- a.out
0000000000600000       4       4       4 r---- a.out
0000000000601000       4       4       4 rw--- a.out
00007f66b0cfe000    1772     224       0 r-x-- libc-2.19.so
00007f66b0eb9000    2044       0       0 ----- libc-2.19.so
00007f66b10b8000      16      16      16 r---- libc-2.19.so
00007f66b10bc000       8       8       8 rw--- libc-2.19.so
00007f66b10be000      20      12      12 rw---   [ anon ]
00007f66b10c3000     140     112       0 r-x-- ld-2.19.so
00007f66b12bb000      12      12      12 rw---   [ anon ]
00007f66b12e2000      12       8       8 rw---   [ anon ]
00007f66b12e5000       4       4       4 r---- ld-2.19.so
00007f66b12e6000       4       4       4 rw--- ld-2.19.so
00007f66b12e7000       4       4       4 rw---   [ anon ]
00007ffec5fd3000     132      12      12 rw---   [ stack ]
00007ffec5ff9000       8       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 r-x--   [ anon ]
---------------- ------- ------- -------
total kB            4192     432      92
```

VM_Alloc = 4192 kB
VM_RSS = 432 kB

Here the file is not yet mapped into the virtual memory. The process has just started.

2.
```
Address           Kbytes     RSS   Dirty Mode  Mapping
0000000000400000       4       4       0 r-x-- a.out
0000000000600000       4       4       4 r---- a.out
0000000000601000       4       4       4 rw--- a.out
00007f66b02fe000   10240       0       0 r---- myfile
00007f66b0cfe000    1772     224       0 r-x-- libc-2.19.so
00007f66b0eb9000    2044       0       0 ----- libc-2.19.so
00007f66b10b8000      16      16      16 r---- libc-2.19.so
00007f66b10bc000       8       8       8 rw--- libc-2.19.so
00007f66b10be000      20      12      12 rw---   [ anon ]
00007f66b10c3000     140     112       0 r-x-- ld-2.19.so
00007f66b12bb000      12      12      12 rw---   [ anon ]
00007f66b12e2000      12      12      12 rw---   [ anon ]
00007f66b12e5000       4       4       4 r---- ld-2.19.so
00007f66b12e6000       4       4       4 rw--- ld-2.19.so
00007f66b12e7000       4       4       4 rw---   [ anon ]
00007ffec5fd3000     132      12      12 rw---   [ stack ]
00007ffec5ff9000       8       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 r-x--   [ anon ]
---------------- ------- ------- -------
total kB           14432     436      92
```

VM-Alloc = 14432kB

VM-RSS = 356kB

The file (10 MB) has newly been allocated in virtual memory leading to a corresponding increase in the VM_Alloc. VM_RSS also increases by 4 Kb but this corresponds to an object called [anon] and not to the memory mapped file.

3.

```
Address           Kbytes     RSS   Dirty Mode  Mapping
0000000000400000       4       4       0 r-x-- a.out
0000000000600000       4       4       4 r---- a.out
0000000000601000       4       4       4 rw--- a.out
00007f66b02fe000   10240       4       0 r---- myfile
00007f66b0cfe000    1772     252       0 r-x-- libc-2.19.so
00007f66b0eb9000    2044       0       0 ----- libc-2.19.so
00007f66b10b8000      16      16      16 r---- libc-2.19.so
00007f66b10bc000       8       8       8 rw--- libc-2.19.so
00007f66b10be000      20      12      12 rw---   [ anon ]
00007f66b10c3000     140     112       0 r-x-- ld-2.19.so
00007f66b12bb000      12      12      12 rw---   [ anon ]
00007f66b12e1000      16      16      16 rw---   [ anon ]
00007f66b12e5000       4       4       4 r---- ld-2.19.so
00007f66b12e6000       4       4       4 rw--- ld-2.19.so
00007f66b12e7000       4       4       4 rw---   [ anon ]
00007ffec5fd3000     132      12      12 rw---   [ stack ]
00007ffec5ff9000       8       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 r-x--   [ anon ]
---------------- ------- ------- -------
total kB           14436     472      96
```

VM-Alloc = 14436kB
VM-RSS =  472kB

Here the VM_Alloc is unchanged. VM_RSS increases by 36 Kb but the RSS corresponding to the memory mapped file only increases by 4 Kb. This corresponds to a page being loaded into the physical memory. When we ask to wriite into the first position the first memory mapped page of the file in Virtual Memory must be loaded into physical memory.

4.

```
Address           Kbytes     RSS   Dirty Mode  Mapping
0000000000400000       4       4       0 r-x-- a.out
0000000000600000       4       4       4 r---- a.out
0000000000601000       4       4       4 rw--- a.out
00007f66b02fe000   10240       8       0 r---- myfile
00007f66b0cfe000    1772     252       0 r-x-- libc-2.19.so
00007f66b0eb9000    2044       0       0 ----- libc-2.19.so
00007f66b10b8000      16      16      16 r---- libc-2.19.so
00007f66b10bc000       8       8       8 rw--- libc-2.19.so
00007f66b10be000      20      12      12 rw---   [ anon ]
00007f66b10c3000     140     112       0 r-x-- ld-2.19.so
00007f66b12bb000      12      12      12 rw---   [ anon ]
00007f66b12e1000      16      16      16 rw---   [ anon ]
00007f66b12e5000       4       4       4 r---- ld-2.19.so
00007f66b12e6000       4       4       4 rw--- ld-2.19.so
00007f66b12e7000       4       4       4 rw---   [ anon ]
00007ffec5fd3000     132      12      12 rw---   [ stack ]
00007ffec5ff9000       8       4       0 r-x--   [ anon ]
ffffffffff600000       4       0       0 r-x--   [ anon ]
---------------- ------- ------- -------
total kB           14436     476      96
```

VM-Alloc = 14436 kB
VM-RSS = 476kB

Here the VM_Alloc is unchanged. VM_RSS increases by 4 Kb and the RSS corresponding to the memory mapped file only increases by 4 Kb. This corresponds to a page being loaded into the physical memory. When we ask to wriite into the 10000th position the first memory mapped page of the file in Virtual Memory does not contain this location and a different page of the file must now be loaded into physical memory.

**Part B:--------------------------------------------------------------**

1->
Throughput: 216.804356 MBps

2->
Throughput: 211.978794 MBps

3->
We do not see a large difference in the reading throughput whether we use memory mapped files or direct file reads, with memory mapping being slightly better because there is no copying of pages from kernel space memory to user space memory as happens in direct reads. However this is a minor factor in comparison to the time taken to read files form the disk onto the memory which happens similarly in both cases. Thus this small advantage of memory mapping cannot yield a large benefit in this case.

4->

Part 1:

Before Experiment:

On running the free -m command we get:

```
             total       used       free     shared    buffers     cached
Mem:          7885       1456       6429        205          3        616
-/+ buffers/cache:        836       7049
Swap:            0          0          0
```

Thus 616 MB is the disk buffer cache size.

After experiment:

```
             total       used       free     shared    buffers     cached
Mem:          7885       1696       6189        205          3        856
-/+ buffers/cache:        835       7050
Swap:            0          0          0
```

Thus 856 MB is the disk buffer cache size.

Since every block of the file is read in this Part all/many of these blocks end up in the disk buffer cache. We see a 250 MB increase in the cache size corresponding to all the files being cached.

Part 2:

Before:

```
             total       used       free     shared    buffers     cached
Mem:          7885       1468       6417        205          5        619
-/+ buffers/cache:        842       7043
Swap:            0          0          0
```

619 MB disk buffer cache size

After:

```
             total       used       free     shared    buffers     cached
```

| | | | | | | |
|---|---|---|---|---|---|---|
| Mem: | 7885 | 1740 | 6145 | 206 | 9 | 887 |
| -/+ buffers/cache: | | 843 | 7042 | | | |
| Swap: | 0 | 0 | 0 | | | |

887 MB disk buffer cache size.

Since every block of the file is read in this Part all/many of these blocks end up in the disk buffer cache. We see a 268 MB increase in the cache size corresponding to all the files being cached. There may be some further cacheing of libraries or other disk blocks leading to the further increase in cache size.

5->
Throughput: 212.955531 MBps

6->
Throughput: 23.585518 MBps

The write throughut in this case using ordinary writes is much less than that obtained using memory mapping. The primary reason for this is that in the case of memory mapped file writes the writes happen into memory. Once all the writes have completed the mapped memory is flushed to disk. However in the case of ordinary writes, every write occors to disk and requires the immediate flushing of a page to disk. When a large amount of data is written this leads to heavy overhead as many pages are repeated written to disk. Thus the throughput obtained is much less than in the case of memory mapped writes.

7->
Frm the exercises above it is clear that using memory mapped files gives significant performance benefits over regular files, when there are a large number of writes and the file contents are not in the disk buffer cache, as it allows us to perform writes largely to memory instead of repeatedly writing to disk. It also has the advantage of not requiring pages to be copied for the kernel space in memory to the user space while performing file reads.

8->
Throughput with ordinary files: 282.166031 MBps
Throughput with memory mapped files: 298.719743 MBps

We see that we do not obtain a significant advantage from memory mapped files as opposed to ordinary files in this case. The number of writes is smaller and each write takes less time due to cacheing, leading to high throughput in both cases and no major advantage for memory mapping.