OS LAB 10

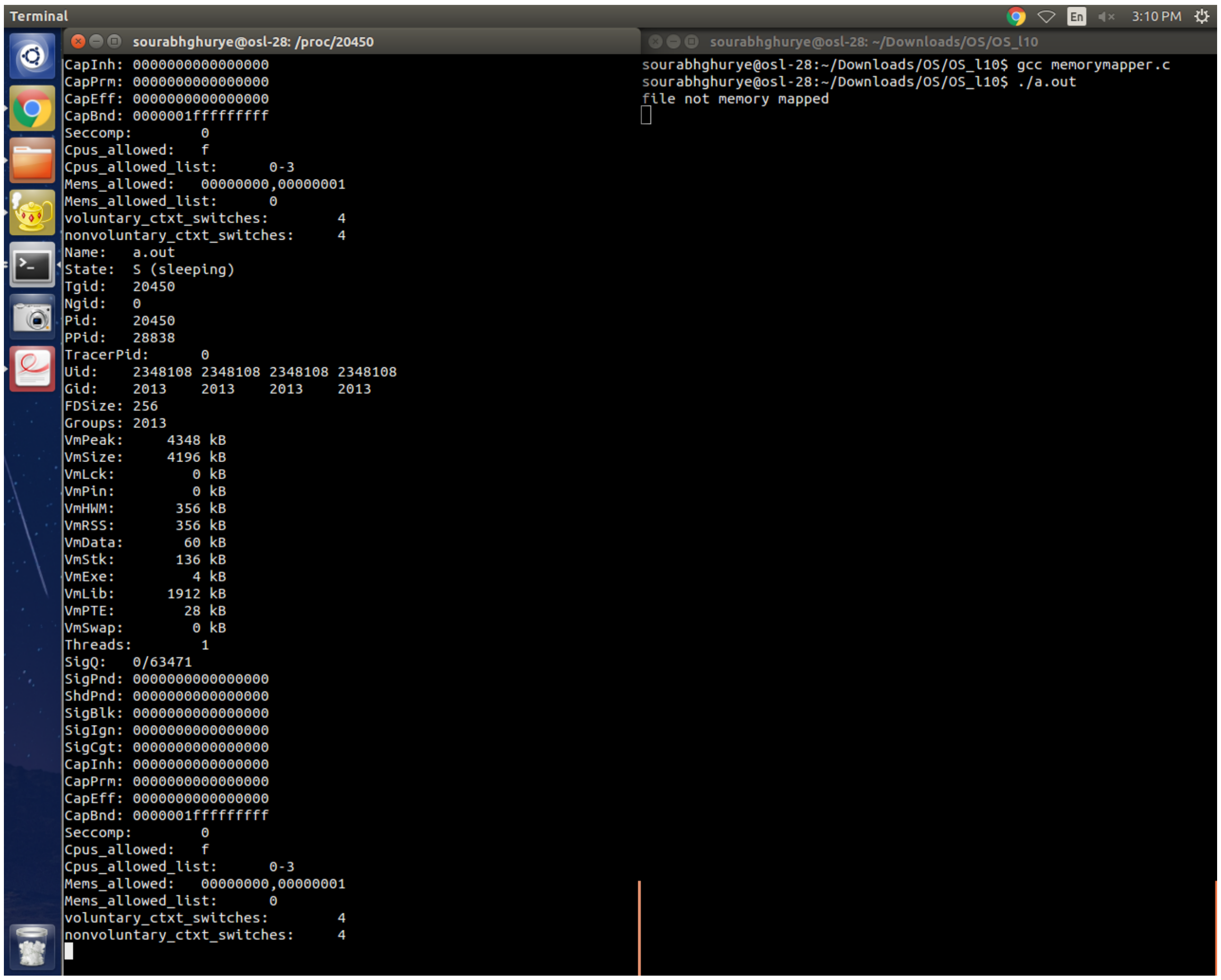
130050001 : Ghurye Sourabh Sunil
130050037 : Utkarsh Mall

Q [A].

Part (1) : cat /proc/<pid>/status

VMalloc = 4196 kB
VMRSS = 356 kB

Explanation : The VMalloc is low as we have not mapped the file yet.


Terminal
3:10 PM
sourabhghurye@osl-28: /proc/20450
sourabhghurye@osl-28: ~/Downloads/OS/OS_l10

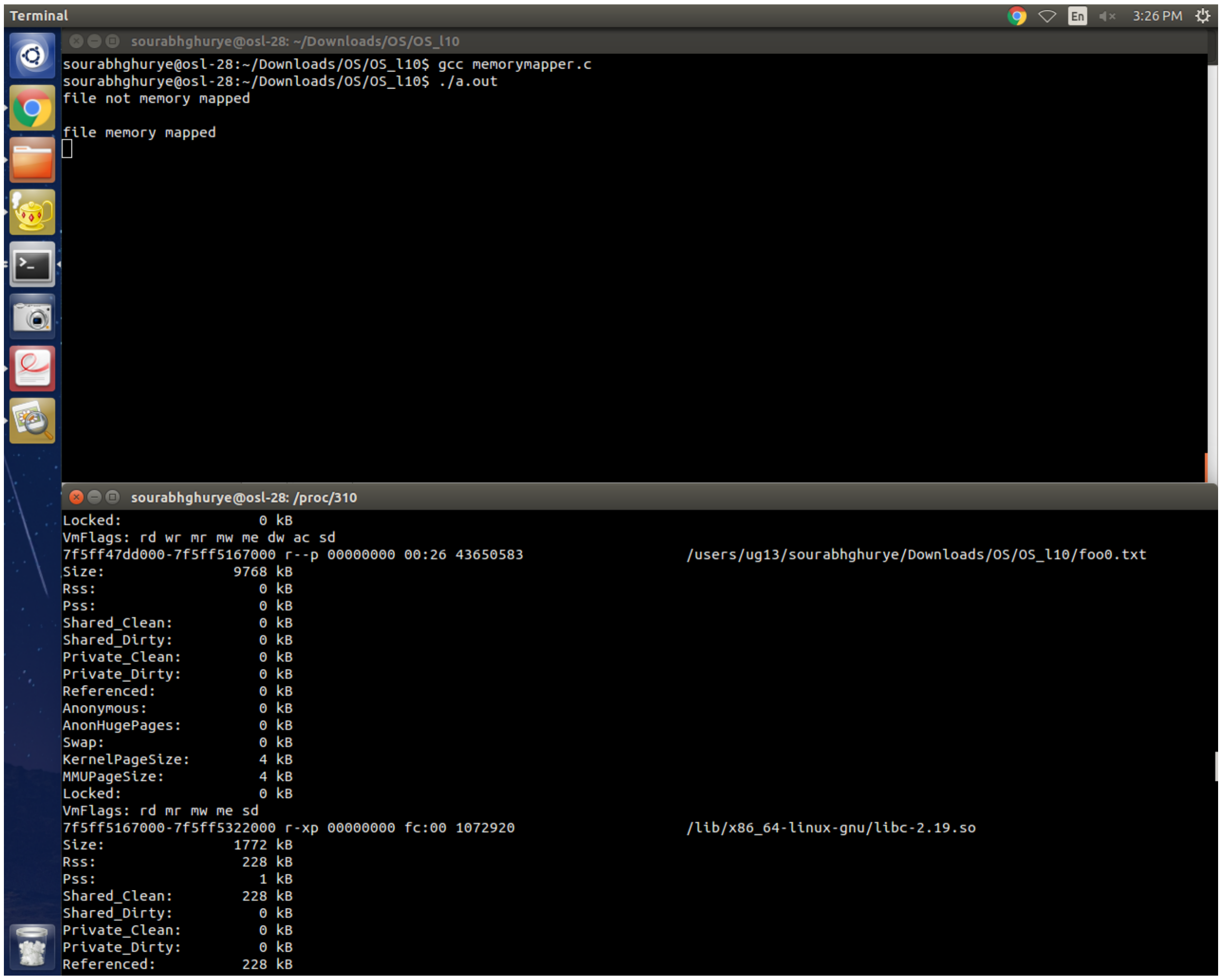
```
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000001fffffffff
Seccomp:        0
Cpus_allowed:   f
Cpus_allowed_list:      0-3
Mems_allowed:   00000000,00000001
Mems_allowed_list:      0
voluntary_ctxt_switches:        4
nonvoluntary_ctxt_switches:     4
Name:   a.out
State:  S (sleeping)
Tgid:   20450
Ngid:   0
Pid:    20450
PPid:   28838
TracerPid:      0
Uid:    2348108 2348108 2348108 2348108
Gid:    2013    2013    2013    2013
FDSize: 256
Groups: 2013
VmPeak:     4348 kB
VmSize:     4196 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:       356 kB
VmRSS:       356 kB
VmData:       60 kB
VmStk:       136 kB
VmExe:         4 kB
VmLib:      1912 kB
VmPTE:        28 kB
VmSwap:        0 kB
Threads:        1
SigQ:   0/63471
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000000000
SigCgt: 0000000000000000
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: 0000001fffffffff
Seccomp:        0
Cpus_allowed:   f
Cpus_allowed_list:      0-3
Mems_allowed:   00000000,00000001
Mems_allowed_list:      0
voluntary_ctxt_switches:        4
nonvoluntary_ctxt_switches:     4
```

```
sourabhghurye@osl-28:~/Downloads/OS/OS_l10$ gcc memorymapper.c
sourabhghurye@osl-28:~/Downloads/OS/OS_l10$ ./a.out
file not memory mapped
```

Part(2) : cat /proc/<pid>/status

VMalloc = 13964 kB
VMRSS = 356 kB
VMRSS for mapped file section = **0 kB**   cat /proc/<pid>/smaps

Explanation : Virtual memory is allocated to the mapped file. Therefore VMAlloc increases by approx 10MB. There is no change in VMRSS as the file is only mapped for now and is not in the physical memory.



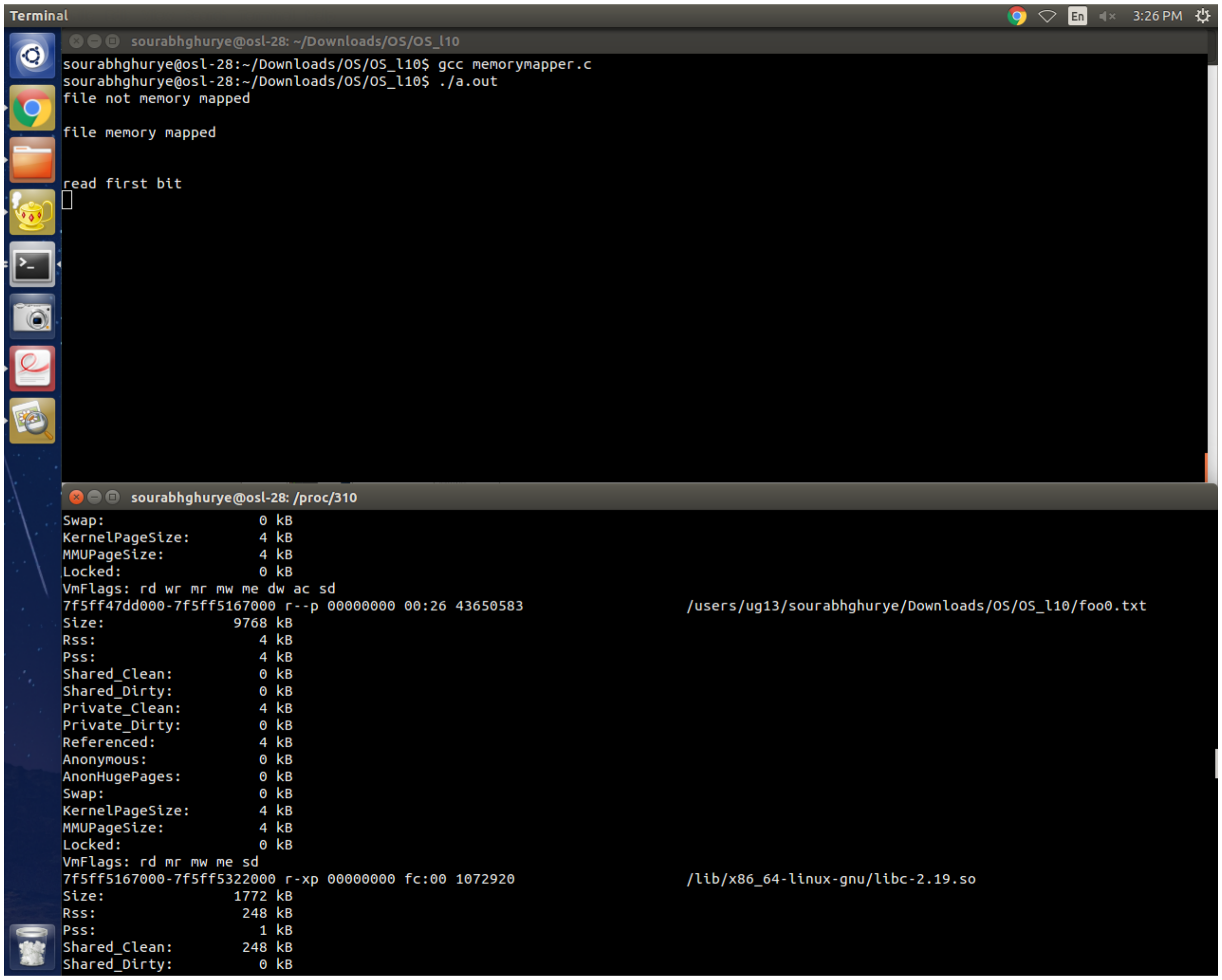
```
sourabhghurye@osl-28: ~/Downloads/OS/OS_l10
sourabhghurye@osl-28:~/Downloads/OS/OS_l10$ gcc memorymapper.c
sourabhghurye@osl-28:~/Downloads/OS/OS_l10$ ./a.out
file not memory mapped

file memory mapped
```

```
sourabhghurye@osl-28: /proc/310
Locked:                0 kB
VmFlags: rd wr mr mw me dw ac sd
7f5ff47dd000-7f5ff5167000 r--p 00000000 00:26 43650583                    /users/ug13/sourabhghurye/Downloads/OS/OS_l10/foo0.txt
Size:               9768 kB
Rss:                   0 kB
Pss:                   0 kB
Shared_Clean:          0 kB
Shared_Dirty:          0 kB
Private_Clean:         0 kB
Private_Dirty:         0 kB
Referenced:            0 kB
Anonymous:             0 kB
AnonHugePages:         0 kB
Swap:                  0 kB
KernelPageSize:        4 kB
MMUPageSize:           4 kB
Locked:                0 kB
VmFlags: rd mr mw me sd
7f5ff5167000-7f5ff5322000 r-xp 00000000 fc:00 1072920                     /lib/x86_64-linux-gnu/libc-2.19.so
Size:               1772 kB
Rss:                 228 kB
Pss:                   1 kB
Shared_Clean:        228 kB
Shared_Dirty:          0 kB
Private_Clean:         0 kB
Private_Dirty:         0 kB
Referenced:          228 kB
```

Part(3) : cat /proc/<pid>/status

VMalloc = 13964 kB
VMRSS = 356 kB
VMRSS for mapped file section = **4 kB** cat /proc/<pid>/smaps

Explanation : status file shows the VMRSS up to the granularity of ~350kB. Hence there is no change. VMRSS for the mapped file section is now 4kB. We can see that the 4kB of file data has been brought into the RAM to serve the first file access.


Terminal
3:26 PM
sourabhghurye@osl-28: ~/Downloads/OS/OS_l10
sourabhghurye@osl-28: /proc/310

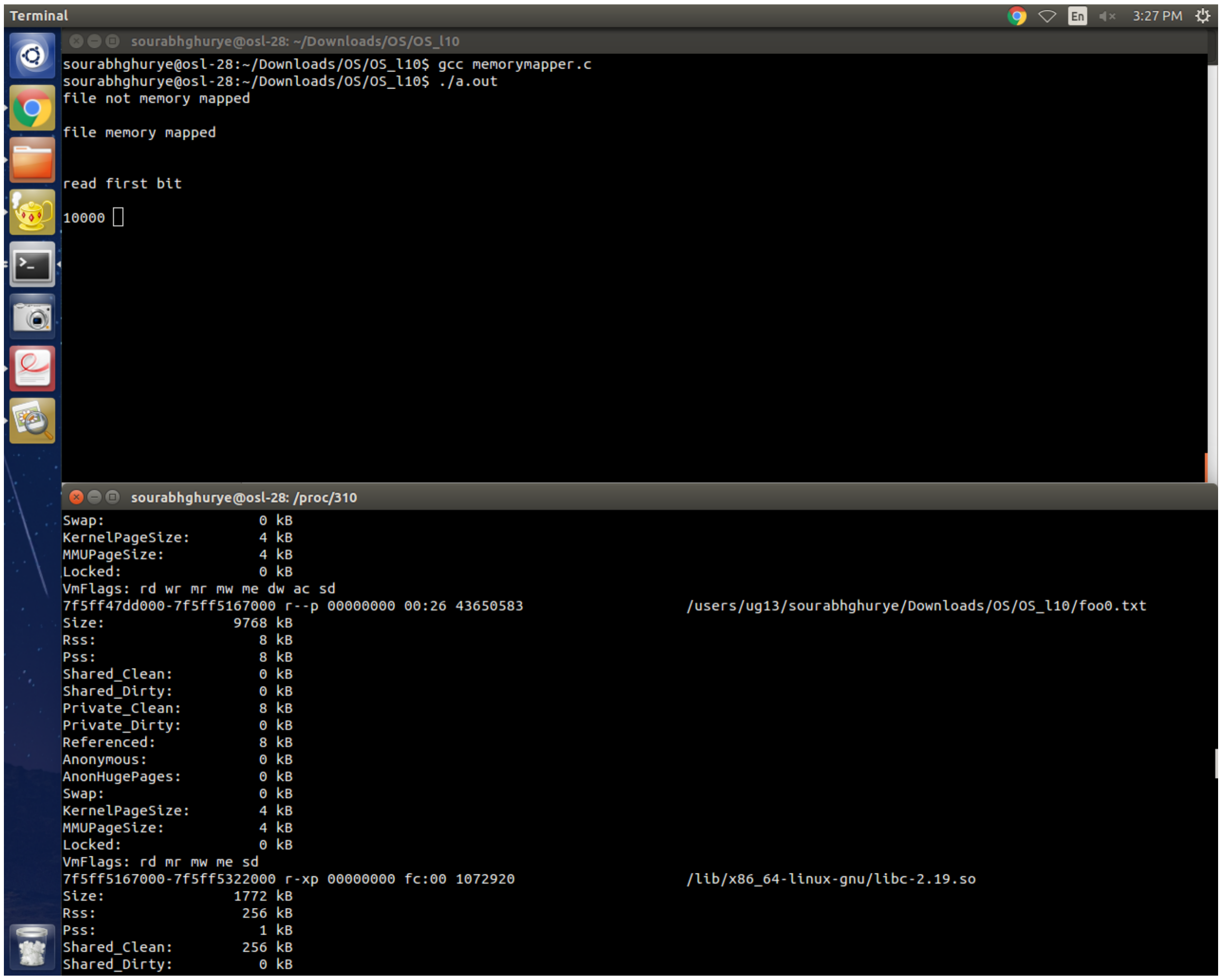
```
sourabhghurye@osl-28:~/Downloads/OS/OS_l10$ gcc memorymapper.c
sourabhghurye@osl-28:~/Downloads/OS/OS_l10$ ./a.out
file not memory mapped

file memory mapped


read first bit
```

```
Swap:                  0 kB
KernelPageSize:        4 kB
MMUPageSize:           4 kB
Locked:                0 kB
VmFlags: rd wr mr mw me dw ac sd
7f5ff47dd000-7f5ff5167000 r--p 00000000 00:26 43650583                   /users/ug13/sourabhghurye/Downloads/OS/OS_l10/foo0.txt
Size:               9768 kB
Rss:                   4 kB
Pss:                   4 kB
Shared_Clean:          0 kB
Shared_Dirty:          0 kB
Private_Clean:         4 kB
Private_Dirty:         0 kB
Referenced:            4 kB
Anonymous:             0 kB
AnonHugePages:         0 kB
Swap:                  0 kB
KernelPageSize:        4 kB
MMUPageSize:           4 kB
Locked:                0 kB
VmFlags: rd mr mw me sd
7f5ff5167000-7f5ff5322000 r-xp 00000000 fc:00 1072920                    /lib/x86_64-linux-gnu/libc-2.19.so
Size:               1772 kB
Rss:                 248 kB
Pss:                   1 kB
Shared_Clean:        248 kB
Shared_Dirty:          0 kB
```

Part(4) : cat /proc/<pid>/status

VMalloc = 13964 kB
VMRSS = 356 kB
VMRSS for mapped file section = **8 kB**   cat /proc/<pid>/smaps

Explanation : status file shows the VMRSS up to the granularity of ~350kB. Hence there is no change. VMRSS for the mapped file section is now 8kB. The 10,000th byte was not in the RAM initially(only 4kB was there). Therefore, more content is fetched to serve the access.



```
sourabhghurye@osl-28: ~/Downloads/OS/OS_l10
sourabhghurye@osl-28:~/Downloads/OS/OS_l10$ gcc memorymapper.c
sourabhghurye@osl-28:~/Downloads/OS/OS_l10$ ./a.out
file not memory mapped

file memory mapped


read first bit

10000
```

```
sourabhghurye@osl-28: /proc/310
Swap:                  0 kB
KernelPageSize:        4 kB
MMUPageSize:           4 kB
Locked:                0 kB
VmFlags: rd wr mr mw me dw ac sd
7f5ff47dd000-7f5ff5167000 r--p 00000000 00:26 43650583                   /users/ug13/sourabhghurye/Downloads/OS/OS_l10/foo0.txt
Size:               9768 kB
Rss:                   8 kB
Pss:                   8 kB
Shared_Clean:          0 kB
Shared_Dirty:          0 kB
Private_Clean:         8 kB
Private_Dirty:         0 kB
Referenced:            8 kB
Anonymous:             0 kB
AnonHugePages:         0 kB
Swap:                  0 kB
KernelPageSize:        4 kB
MMUPageSize:           4 kB
Locked:                0 kB
VmFlags: rd mr mw me sd
7f5ff5167000-7f5ff5322000 r-xp 00000000 fc:00 1072920                    /lib/x86_64-linux-gnu/libc-2.19.so
Size:               1772 kB
Rss:                 256 kB
Pss:                   1 kB
Shared_Clean:        256 kB
Shared_Dirty:          0 kB
```

Q [B].

Part(1) :

Average Throughput = 54.78 MBps	[block size=512 byte]
Average Throughput = 53.61 MBps	[block size=1 byte]

Explanation : Disk bottleneck is around ~50MBps. As the cache was cleared before the experiment, the data has to be brought from the disk at least once.

Part(2) :

Average Throughput = 48.63 MBps	[block size=512 byte]
Average Throughput = 9.91 MBps	[block size=1 byte]

Explanation : Disk bottleneck is around ~50MBps. If the block size is 512KB we achieve disk bottleneck. But if the block size is smaller, it is far worse than the throughput from part(a) because in that case memory mapped files get the huge advantage of spatial locality over the normal disk read.

Part(3) :

In memory mapped files, the data is stored in the pages allocated to the process(until there is a page fault). Therefore a lot of file content is present in the physical memory which has faster access. It has a much more cache locality advantage over the disk which only has a small buffer cache which is not process specific.

Part(4) :

Disk buffer cache is utilised more while reading from a regular file. Every access to the file goes through the disk buffer cache and therefore the blocks corresponding to the file are not quickly cleared from the cache till the buffer is mostly utilised.
In case of memory mapped file, once the part of the file is brought into the physical memory, accesses to that part of the file do not go to the disk buffer cache and hence, those blocks are quickly cleared from the disk buffer cache.

Part(5) :

Average Throughput = 24.91 MBps [block size=512 byte]
Average Throughput = 23.86 MBps [block size=1 byte]

Explanation : Disk bottleneck is around ~50MBps. As the cache was cleared before the experiment, the data has to be brought from the disk and again has to be written back to the disk. Hence input once and output once from disk is causing throughput reduce by half compared to part(a).

Part(6) :

Average Throughput = 24.91 MBps [block size=512 byte]
Average Throughput = 2.42 MBps [block size=1 byte]

Explanation : Disk bottleneck is around ~50MBps. If the block size is 512KB we achieve disk bottleneck similar to part(a). But if the block size is smaller, it is far worse than the throughput from part(e) because in that case too, memory mapped files get the huge advantage of spatial locality over the normal disk read.

Part(7) :

Memory mapping will give huge advantage over regular disk writes, because in case of regular disk I/Os buffer pages may be replaced on high usage(buffer size < file size), but in case of memory mapping read/write buffer size actually equals file size hence replacement never takes place.

Part(8) :

Number of files=2

Disk average Throughput = 248.79 MBps [block size=512 byte]
Memory mapped average throughput = 37.03 MB/s [block size=1 byte]

In this case memory map is not giving any advantage over regular disk I/O because buffer is totally storing the files (buffer size = file size), hence both methods are equivalent.

Note: Disk throughput is so large, because the time of writing back to disk from buffer is not included in case of regular I/O.