Part - 1 Basics

We wrote a C++ program that performs read/write operation as follows:-

- Read - the read operation reads the file and prints a XOR value. For XOR Value we used code provided by the professor as reference. Our program outputs XOR value in “hex”.
- Write - The write operation of our program, generate a file with given name with random characters of given size(block_size * block_count)

Screenshot of our running program from part - 1

In Screenshot it is visible that when we read a file and change block_size and block_count such that it is the same, XOR value is also not changed. Same is observed with file write1.

Part - 2 Measurements

For this part of our project we created a new code file called run2 “./run2 <filename> <block_size>”. It takes filename and block size as arguments and returns the file size as output.

We have later used this program to get a file of reasonable size that is a file using which the program runs between 5 to 15 seconds only.

Along with filesize we are also printing XOR of file.

```
~/Desktop/OS Final Project/Final Version  ./run2 ubuntu-21.04-desktop-amd64.iso 1024
Time to Read: 725 ms
Final XOR Result (All Blocks): a7eeb2d9
Total Blocks Read: 2752674
File size within reasonable time: 2818738176 bytes
~/Desktop/OS Final Project/Final Version  ./run2 ubuntu-21.04-desktop-amd64.iso 2048
Time to Read: 722 ms
Final XOR Result (All Blocks): a7eeb2d9
Total Blocks Read: 1376337
File size within reasonable time: 2818738176 bytes
~/Desktop/OS Final Project/Final Version  ./run2 ubuntu-21.04-desktop-amd64.iso 4096
Time to Read: 709 ms
Final XOR Result (All Blocks): a7eeb2d9
Total Blocks Read: 688168
File size within reasonable time: 2818736128 bytes
~/Desktop/OS Final Project/Final Version  ./run2 ubuntu-21.04-desktop-amd64.iso 8192
Time to Read: 641 ms
Final XOR Result (All Blocks): a7eeb2d9
Total Blocks Read: 344084
File size within reasonable time: 2818736128 bytes
~/Desktop/OS Final Project/Final Version  ./run2 ubuntu-21.04-desktop-amd64.iso 16384
Time to Read: 483 ms
Final XOR Result (All Blocks): a7eeb2d9
Total Blocks Read: 172042
File size within reasonable time: 2818736128 bytes
~/Desktop/OS Final Project/Final Version  ./run2 ubuntu-21.04-desktop-amd64.iso 32768
Time to Read: 407 ms
Final XOR Result (All Blocks): a7eeb2d9
Total Blocks Read: 86021
File size within reasonable time: 2818736128 bytes
~/Desktop/OS Final Project/Final Version  ./run2 ubuntu-21.04-desktop-amd64.iso 65536
Time to Read: 371 ms
Final XOR Result (All Blocks): a7eeb2d9
Total Blocks Read: 43010
File size within reasonable time: 2818703360 bytes
~/Desktop/OS Final Project/Final Version  ./run2 20test.txt 65536
Time to Read: 8554 ms
Final XOR Result (All Blocks): 1a5539b0
Total Blocks Read: 327680
File size within reasonable time: 21474836480 bytes
```

Extra Credit:

“Dd” is a command-line utility in linux operating systems that stands for "data duplicator" or "disk dump." It is a utility program primarily used for copying and converting data, and it can perform a wide range of tasks related to data manipulation and transfer. dd is known for its flexibility and ability to work with binary data at a low level. Some of its key features are:

- Copying and Cloning: One of the most common uses of dd is to create exact copies of disks or partitions. It can be used to clone an entire hard drive or create backups.
- Data Conversion: dd can convert data from one format to another. For instance, you can use it to convert between different character encodings, change the byte order, or manipulate binary data.
- Creating and Writing to Files: dd can be used to create files filled with specific data patterns or zeros. For example, to create a file filled with zeros.
- Reading and Displaying Data: You can use dd to read data from files or devices and display it on the terminal. This can be useful for inspecting binary data or analyzing disk contents.
- Disk Benchmarking and Performance Measurement: dd can be used to measure disk I/O performance by reading or writing data with specified block sizes and counts. This is useful for testing the speed of storage devices.

Here's how we performed the comparison of our program's performance to the dd program in Linux:

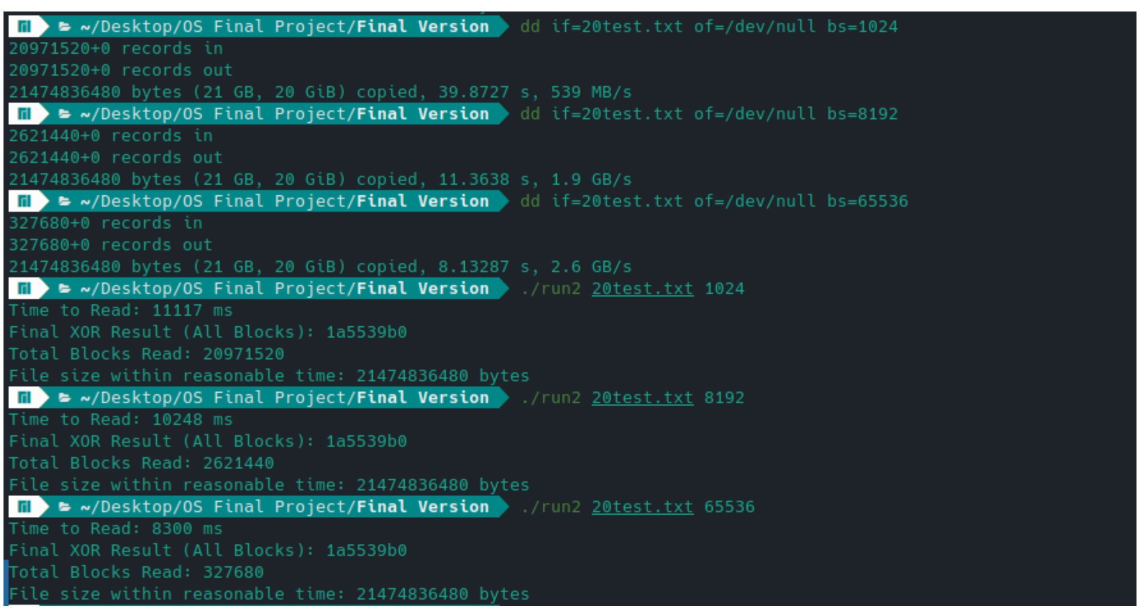
- We use 'dd' to measure the read time for the file you created:
  Command:
  dd <file_name> of=/dev/null bs=block_size
  This command reads the file you created and redirects the output to /dev/null to measure read time.
- After that we run our C++ program i.e. run2 with the same block size on the same file, and measure the read time we did for 'dd'.
- We repeat steps 1 and 2 with different sizes of file.

Analysis of the read times and file sizes between our program and dd:
We performed Analysis using 2 different file:

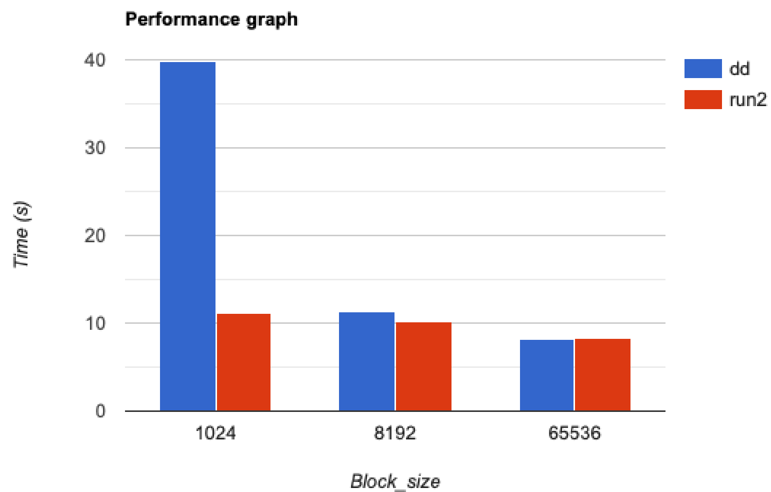
1. File bigger than RAM size
2. File smaller than RAM size

File bigger than RAM size:

Block Size	‘dd’ Time taken to Read (s)	our program (run2) Time taken to Read (s)
1024	39.87	11.11
8192	11.3638	10.248
65536	8.13228	8.3



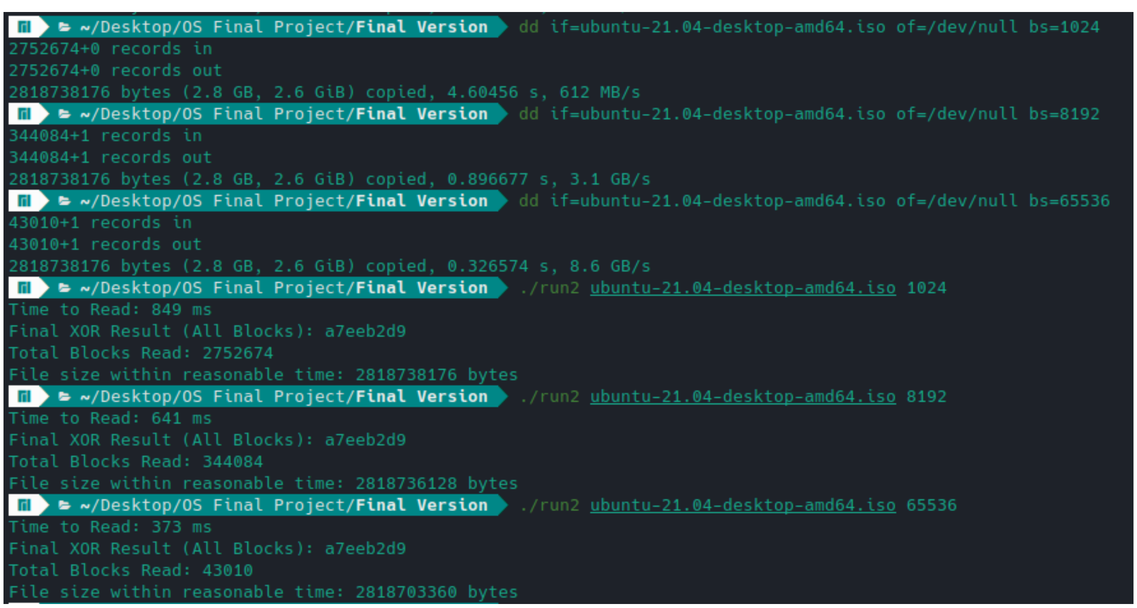
Screenshot of our readings



Graph for above readings

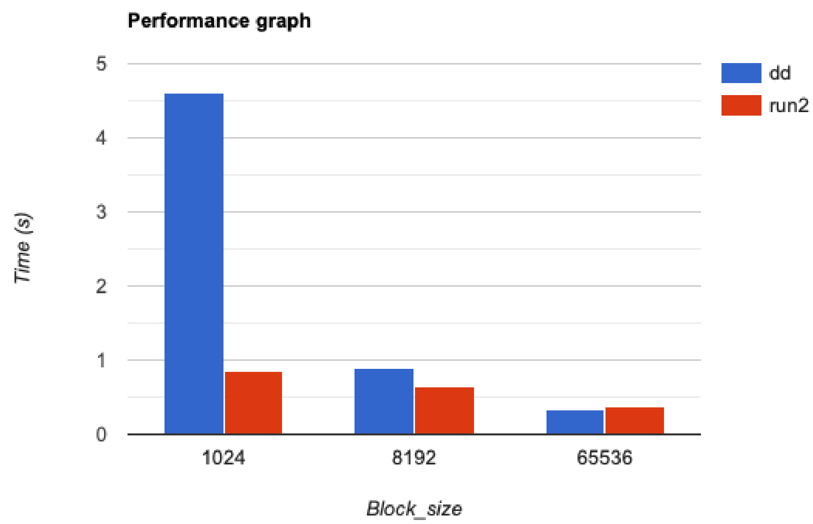
File smaller than RAM size:

Block Size	‘dd’ Time taken to Read (s)	our program (run2) Time taken to Read (s)
1024	4.6	0.849
8192	0.896	0.641
65536	0.326	0.373



```
~/Desktop/OS Final Project/Final Version  dd if=ubuntu-21.04-desktop-amd64.iso of=/dev/null bs=1024
2752674+0 records in
2752674+0 records out
2818738176 bytes (2.8 GB, 2.6 GiB) copied, 4.60456 s, 612 MB/s
~/Desktop/OS Final Project/Final Version  dd if=ubuntu-21.04-desktop-amd64.iso of=/dev/null bs=8192
344084+1 records in
344084+1 records out
2818738176 bytes (2.8 GB, 2.6 GiB) copied, 0.896677 s, 3.1 GB/s
~/Desktop/OS Final Project/Final Version  dd if=ubuntu-21.04-desktop-amd64.iso of=/dev/null bs=65536
43010+1 records in
43010+1 records out
2818738176 bytes (2.8 GB, 2.6 GiB) copied, 0.326574 s, 8.6 GB/s
~/Desktop/OS Final Project/Final Version  ./run2 ubuntu-21.04-desktop-amd64.iso 1024
Time to Read: 849 ms
Final XOR Result (All Blocks): a7eeb2d9
Total Blocks Read: 2752674
File size within reasonable time: 2818738176 bytes
~/Desktop/OS Final Project/Final Version  ./run2 ubuntu-21.04-desktop-amd64.iso 8192
Time to Read: 641 ms
Final XOR Result (All Blocks): a7eeb2d9
Total Blocks Read: 344084
File size within reasonable time: 2818736128 bytes
~/Desktop/OS Final Project/Final Version  ./run2 ubuntu-21.04-desktop-amd64.iso 65536
Time to Read: 373 ms
Final XOR Result (All Blocks): a7eeb2d9
Total Blocks Read: 43010
File size within reasonable time: 2818703360 bytes
```

Screenshot of our readings



Graph for above readings

Google Benchmark:
We integrated Google Benchmark to enhance the evaluation of disk I/O performance. This framework enabled us to conduct more precise and consistent benchmarking of file reading operations. Our program, designed in C++, utilizes Google Benchmark to repeatedly measure the time taken to read files of various block sizes. The test file used for benchmarking is "ubuntu-21.04-desktop-amd64.iso", size 2.8GB and we experimented with block sizes ranging from 1024 to 16384 bytes.

The function, **BM_MeasureReadTime**, automates the process of reading the file with different block sizes, measuring the time taken for each operation. This method provides a comprehensive analysis of how block size affects read performance. By Using Google Benchmark, we achieved a standardized and repeatable measurement approach, allowing for a more accurate assessment of the factors influencing disk I/O performance.

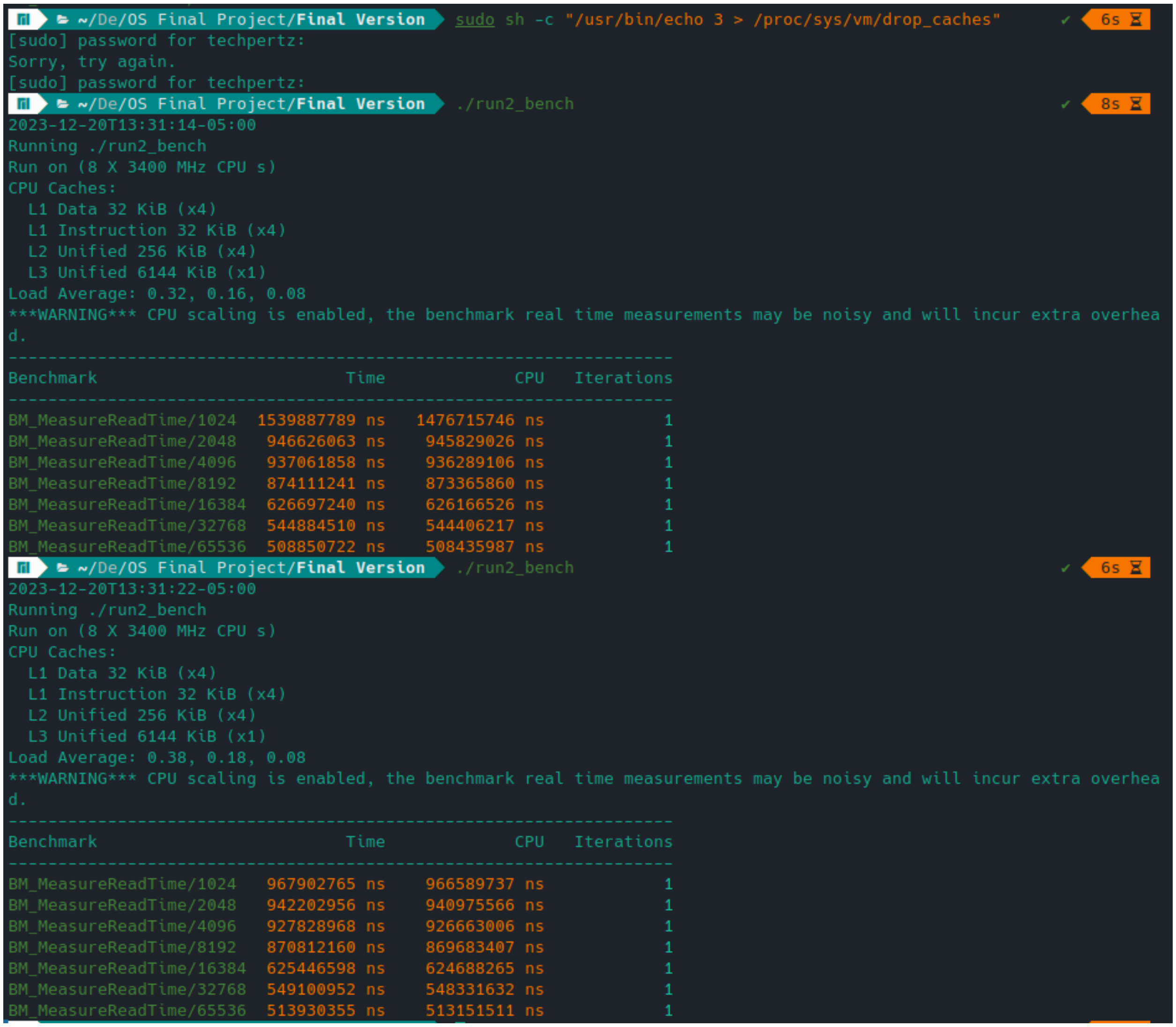
In our code we removed the main() function and instead used BENCHMARK_MAIN(). BENCHMARK_MAIN() is a macro that expands to a main function and is the entry point for the benchmark tests.

BENCHMARK(BM_MeasureReadTime)->Arg(1024)->...->Arg(16384) registers the benchmark function with different arguments, representing different block sizes to test.

Inside of BM_MeasureReadTime Benchmark Function, which is designed for Google Benchmark to measure the read time performance.
We set up a benchmark for the file "ubuntu-21.04-desktop-amd64.iso" with varying block sizes.

Inside the benchmark loop (for (auto _ : state)), we call measureReadTime for each block size specified.

Result of running benchmark on our program:



```
~/De/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"   ✔  6s
[sudo] password for techpertz:
Sorry, try again.
[sudo] password for techpertz:
~/De/OS Final Project/Final Version  ./run2_bench   ✔  8s
2023-12-20T13:31:14-05:00
Running ./run2_bench
Run on (8 X 3400 MHz CPU s)
CPU Caches:
  L1 Data 32 KiB (x4)
  L1 Instruction 32 KiB (x4)
  L2 Unified 256 KiB (x4)
  L3 Unified 6144 KiB (x1)
Load Average: 0.32, 0.16, 0.08
***WARNING*** CPU scaling is enabled, the benchmark real time measurements may be noisy and will incur extra overhea
d.
-------------------------------------------------------------------
Benchmark                          Time             CPU   Iterations
-------------------------------------------------------------------
BM_MeasureReadTime/1024   1539887789 ns   1476715746 ns            1
BM_MeasureReadTime/2048    946626063 ns    945829026 ns            1
BM_MeasureReadTime/4096    937061858 ns    936289106 ns            1
BM_MeasureReadTime/8192    874111241 ns    873365860 ns            1
BM_MeasureReadTime/16384   626697240 ns    626166526 ns            1
BM_MeasureReadTime/32768   544884510 ns    544406217 ns            1
BM_MeasureReadTime/65536   508850722 ns    508435987 ns            1
~/De/OS Final Project/Final Version  ./run2_bench   ✔  6s
2023-12-20T13:31:22-05:00
Running ./run2_bench
Run on (8 X 3400 MHz CPU s)
CPU Caches:
  L1 Data 32 KiB (x4)
  L1 Instruction 32 KiB (x4)
  L2 Unified 256 KiB (x4)
  L3 Unified 6144 KiB (x1)
Load Average: 0.38, 0.18, 0.08
***WARNING*** CPU scaling is enabled, the benchmark real time measurements may be noisy and will incur extra overhea
d.
-------------------------------------------------------------------
Benchmark                          Time             CPU   Iterations
-------------------------------------------------------------------
BM_MeasureReadTime/1024    967902765 ns    966589737 ns            1
BM_MeasureReadTime/2048    942202956 ns    940975566 ns            1
BM_MeasureReadTime/4096    927828968 ns    926663006 ns            1
BM_MeasureReadTime/8192    870812160 ns    869683407 ns            1
BM_MeasureReadTime/16384   625446598 ns    624688265 ns            1
BM_MeasureReadTime/32768   549100952 ns    548331632 ns            1
BM_MeasureReadTime/65536   513930355 ns    513151511 ns            1
```

Screenshot of benchmark result

- We used the "`ubuntu-21.04-desktop-amd64.iso`" file for the benchmark test.

Part - 3 Raw Performance

For this part we are running a program called ./run3, which is the object of run3.cpp.
Here we tried using the same file with different block_size. And for each block size noted the speed in MiB/s in the below given table.

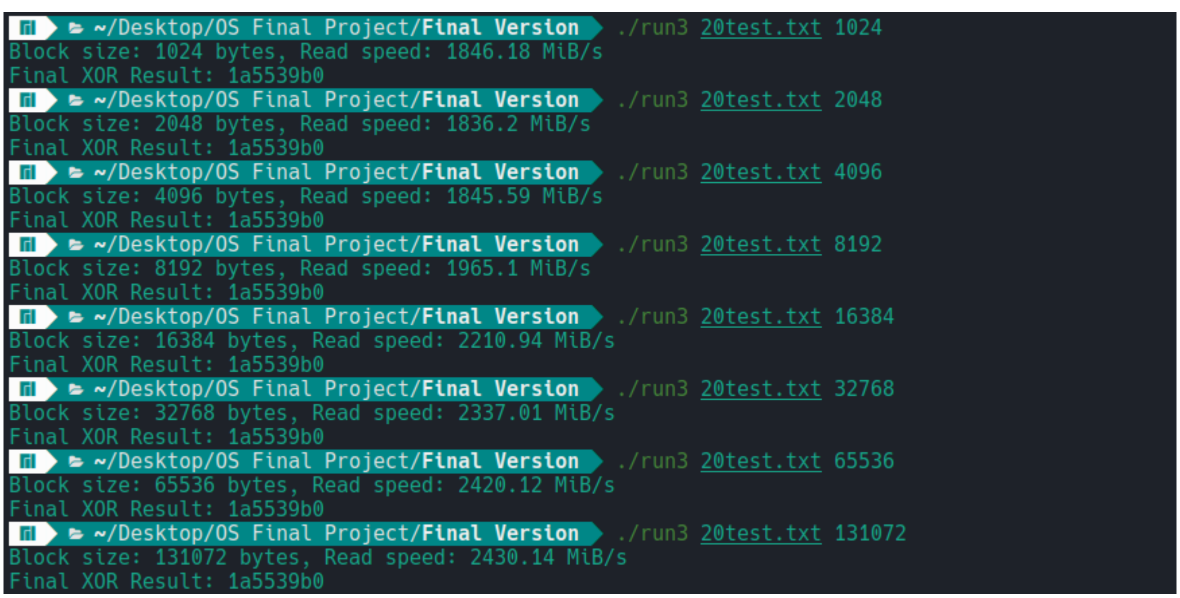
For the better performance reading we performed check with two files:

- File larger than RAM size
- File smaller than RAM size

Screenshots of both are attached below, also the respective graph of two different performance checks.

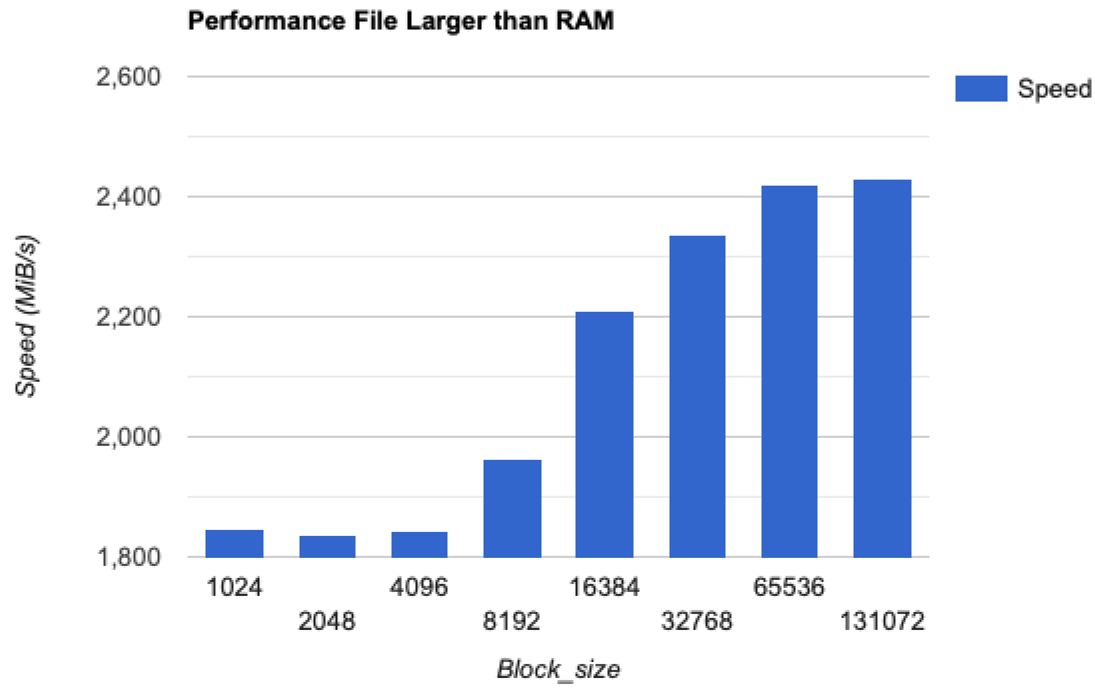
File larger than RAM size:

Blocksize	Speed in MiB/s
1024	1846
2048	1836
4096	1845
8192	1965
16384	2210
32768	2337
65536	2420
131072	2430



```
~/Desktop/OS Final Project/Final Version ./run3 20test.txt 1024
Block size: 1024 bytes, Read speed: 1846.18 MiB/s
Final XOR Result: 1a5539b0
~/Desktop/OS Final Project/Final Version ./run3 20test.txt 2048
Block size: 2048 bytes, Read speed: 1836.2 MiB/s
Final XOR Result: 1a5539b0
~/Desktop/OS Final Project/Final Version ./run3 20test.txt 4096
Block size: 4096 bytes, Read speed: 1845.59 MiB/s
Final XOR Result: 1a5539b0
~/Desktop/OS Final Project/Final Version ./run3 20test.txt 8192
Block size: 8192 bytes, Read speed: 1965.1 MiB/s
Final XOR Result: 1a5539b0
~/Desktop/OS Final Project/Final Version ./run3 20test.txt 16384
Block size: 16384 bytes, Read speed: 2210.94 MiB/s
Final XOR Result: 1a5539b0
~/Desktop/OS Final Project/Final Version ./run3 20test.txt 32768
Block size: 32768 bytes, Read speed: 2337.01 MiB/s
Final XOR Result: 1a5539b0
~/Desktop/OS Final Project/Final Version ./run3 20test.txt 65536
Block size: 65536 bytes, Read speed: 2420.12 MiB/s
Final XOR Result: 1a5539b0
~/Desktop/OS Final Project/Final Version ./run3 20test.txt 131072
Block size: 131072 bytes, Read speed: 2430.14 MiB/s
Final XOR Result: 1a5539b0
```

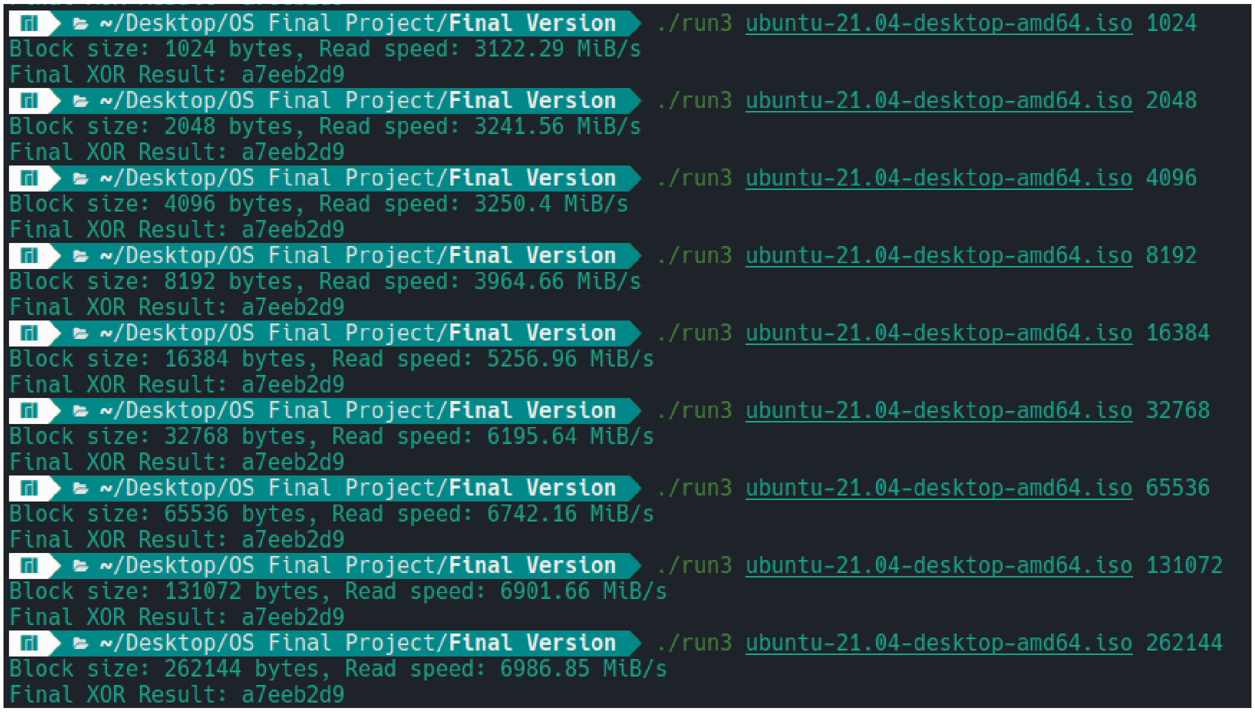
Screenshot of readings for File Larger than RAM



The graph for generated readings

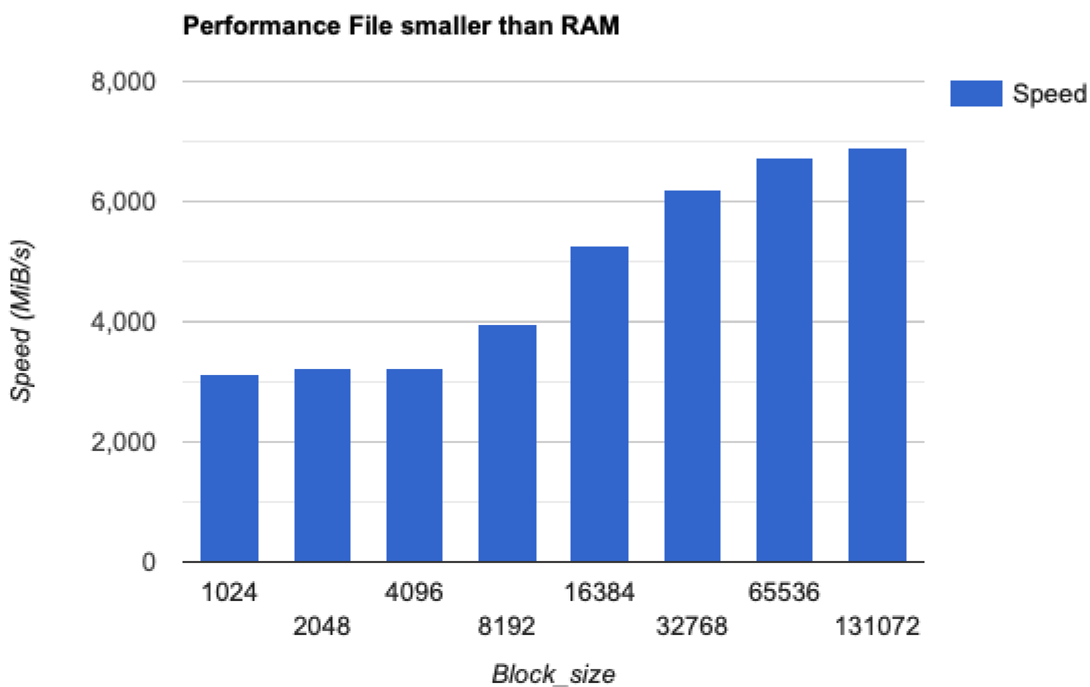
I. File smaller than RAM size

Blocksize	Speed in MiB/s
1024	3122
2048	3241
4096	3250
8192	3964
16384	5256
32768	6195
65536	6742
131072	6901



```
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 1024
Block size: 1024 bytes, Read speed: 3122.29 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 2048
Block size: 2048 bytes, Read speed: 3241.56 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 4096
Block size: 4096 bytes, Read speed: 3250.4 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 8192
Block size: 8192 bytes, Read speed: 3964.66 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 16384
Block size: 16384 bytes, Read speed: 5256.96 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 32768
Block size: 32768 bytes, Read speed: 6195.64 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 65536
Block size: 65536 bytes, Read speed: 6742.16 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 131072
Block size: 131072 bytes, Read speed: 6901.66 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 262144
Block size: 262144 bytes, Read speed: 6986.85 MiB/s
Final XOR Result: a7eeb2d9
```

Screenshot of our readings



The graph for generated readings

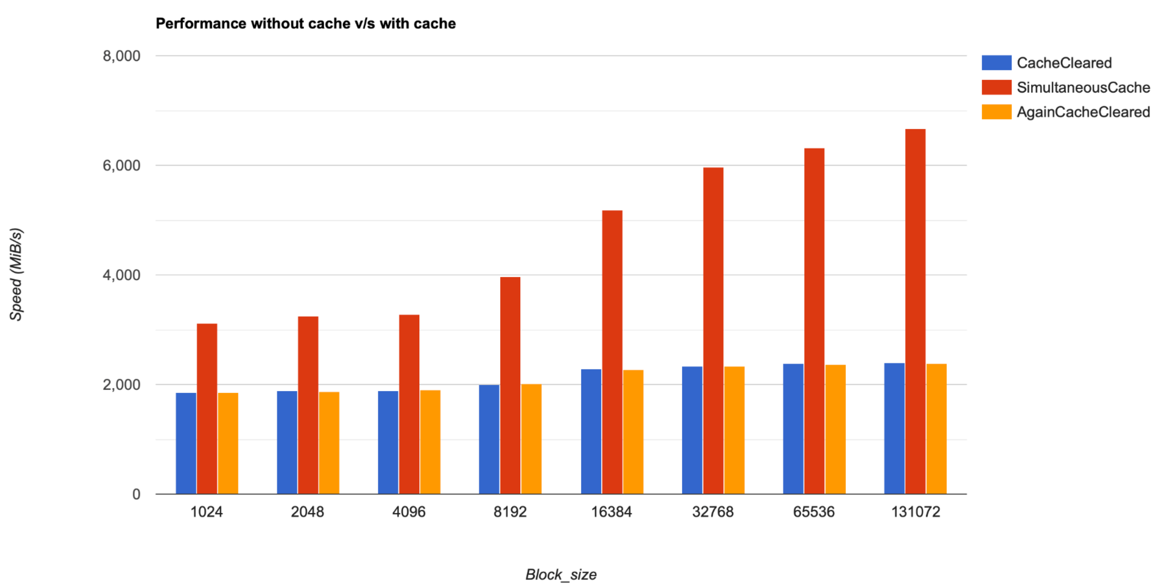
Part - 4 Raw Performance

Using the same program ./run3 will got reading for different block size in following steps

1. After clearing cache.
2. Immediate run with cache (MiB/s)
3. Running After clearing cache again (MiB/s)

When File is smaller than RAM:

Blocksize	When cache is cleared (MiB/s)	Immediate run with cache (MiB/s)	Third run with cleared cache (MiB/s)
1024	1863	3127	1854
2048	1884	3244	1875
4096	1882	3288	1901
8192	2000	3966	2009
16384	2289	5183	2274
32768	2333	5969	2333
65536	2391	6323	2368
131072	2396	6672	2381



The graph for generated readings

Screenshots of our readings:

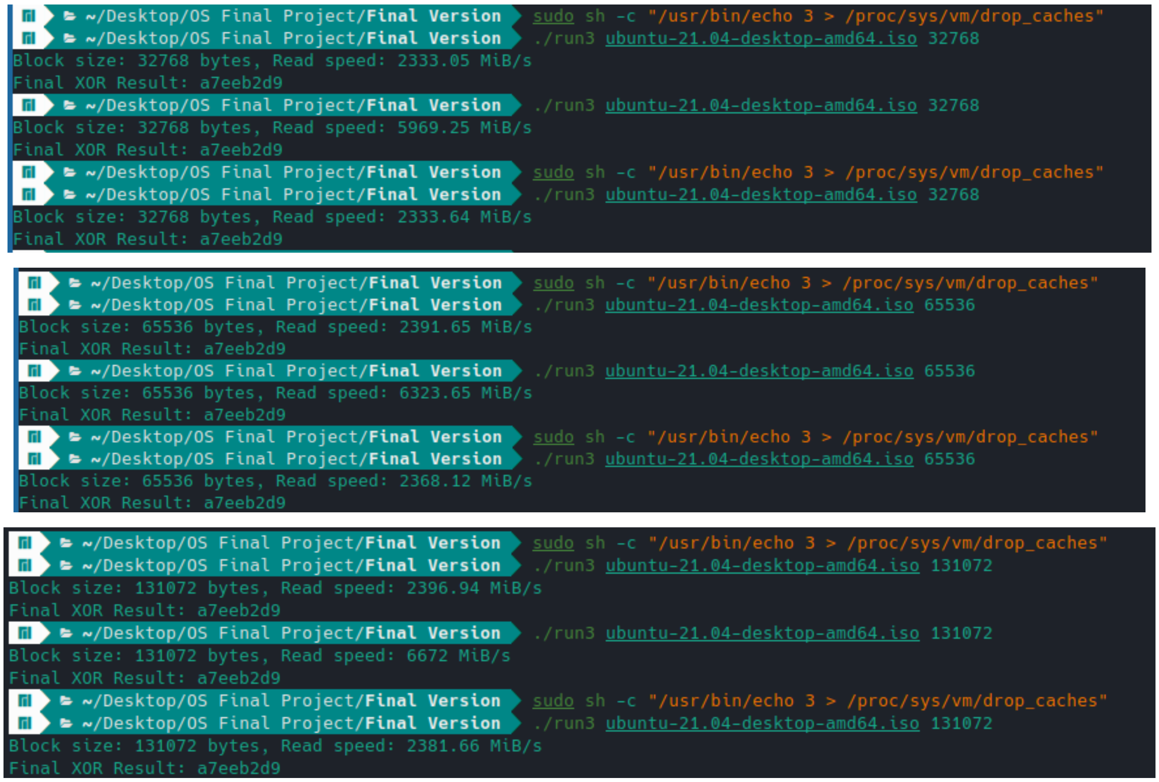
```
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
[sudo] password for techpertz:
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 1024
Block size: 1024 bytes, Read speed: 1863.05 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 1024
Block size: 1024 bytes, Read speed: 3127.49 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 1024
Block size: 1024 bytes, Read speed: 1854.16 MiB/s
Final XOR Result: a7eeb2d9
```

```
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 2048
Block size: 2048 bytes, Read speed: 1884.21 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 2048
Block size: 2048 bytes, Read speed: 3244.51 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 2048
Block size: 2048 bytes, Read speed: 1875.77 MiB/s
Final XOR Result: a7eeb2d9
```

```
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 4096
Block size: 4096 bytes, Read speed: 1882.14 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 4096
Block size: 4096 bytes, Read speed: 3288 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 4096
Block size: 4096 bytes, Read speed: 1901.07 MiB/s
Final XOR Result: a7eeb2d9
```

```
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 8192
Block size: 8192 bytes, Read speed: 2000.97 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 8192
Block size: 8192 bytes, Read speed: 3966.87 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 8192
Block size: 8192 bytes, Read speed: 2009.47 MiB/s
Final XOR Result: a7eeb2d9
```

```
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 16384
Block size: 16384 bytes, Read speed: 2289.75 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 16384
Block size: 16384 bytes, Read speed: 5183 MiB/s
Final XOR Result: a7eeb2d9
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
~/Desktop/OS Final Project/Final Version  ./run3 ubuntu-21.04-desktop-amd64.iso 16384
Block size: 16384 bytes, Read speed: 2274.67 MiB/s
Final XOR Result: a7eeb2d9
```



Screenshot of our readings

Similarly for file larger than RAM:

Blocksize	When cache is cleared (MiB/s)	Simultaneous run with cache (MiB/s)	Third run with cleared cache (MiB/s)
4096	1871	1843	1890
65536	2379	2432	2371

Screenshot of our readings

Extra Credit:
On Linux there is a way to clear the disk caches without rebooting your machine. That is "`sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"`.
Why "`3`"? Read up on it and explain.

In Linux "`/proc/sys/vm/drop_caches`" is used to clear the system's page cache, dentries (directory entries), and inodes. The number "3" in the command to the kernel, specifying which caches to drop.

The "`/proc/sys/vm/drop_caches`" is a special file that allows you to control how the kernel manages its caches. Writing different numbers to this file triggers different behaviors:

"1":

This clears the page cache, which contains cached data from files on disk. The page cache speeds up data access by holding recently accessed file data in memory. It clears this cache, freeing up memory.

"2":

This clears the dentries and inodes. Dentries cache directory entries, which are used to keep track of the hierarchy and metadata of the filesystem. Inodes which are used for storing information about files and directories, such as file ownership, access mode (file permissions), and file type.

"3":

This option combines the effects of "1" and "2". It clears the page cache, dentries, and inodes. Writing "3" is a comprehensive way to free up most of the memory used for caching file system metadata and file content without affecting system stability.

Part 5: System Calls

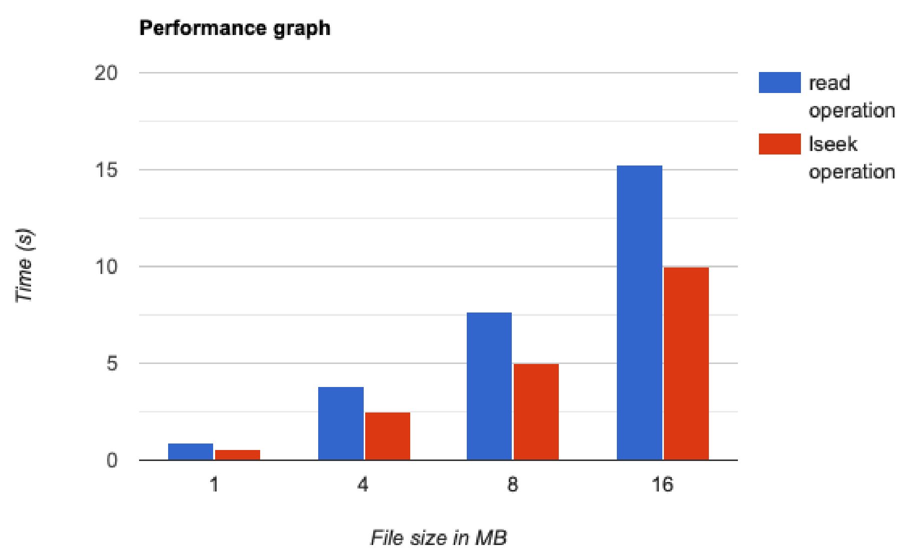
We have did this task in two part: “run5a” and “run5b”:

1. “run5a”:
   This is the object of the run5a.cpp file. In this file we are measuring performance for “read” system call with block size = 1.
   We tested the performance for files of multiple sizes, as shown in the screenshot below. And noticed that for larger size file the time to read file file also increases by larger value.
2. “run5b”:
   This is the object of the run5b.cpp file. In this file we are measuring performance for “lseek” system call with block size = 1.
   We tested the performance for files of multiple sizes, as shown in the screenshot below. And noticed that for larger size file the time to read file file also increases by larger value. Which is similar to “read”.
   But when comparing two “lseek” is faster than “read” due to less number of kernel interrupt inside of “lseek”

Readings:

File size in MB	Time Taken in seconds	
	READ	LSEEK
1	0.95	0.62
4	3.81	2.49
8	7.63	4.99
16	15.27	10.04

Graph for above readings:



Screenshot of readings:

```
~/De/OS Final Project/Final Version  ./run5a 1mbtest.txt 1                                4s
Measuring performance for read system call with block size: 1
Read: Performed 1048576 operations in 0.95 seconds.
Performance: 1.05 MiB/s, 1101591.90 B/s
File size: 1048576 bytes
Number of blocks read: 1048576
~/Desktop/OS Final Project/Final Version  ./run5b 1mbtest.txt 1
Measuring performance for lseek system call with block size: 1
lseek: Performed 1048576 operations in 0.62 seconds.
Performance: 1.60 MiB/s, 1679282.50 B/s
File size: 1048576 bytes
Number of blocks read: 1048576
```

```
~/Desktop/OS Final Project/Final Version  ./run5a 4mbtest.txt 1
Measuring performance for read system call with block size: 1
Read: Performed 4194304 operations in 3.81 seconds.
Performance: 1.05 MiB/s, 1102293.78 B/s
File size: 4194304 bytes
Number of blocks read: 4194304
~/De/OS Final Project/Final Version  ./run5b 4mbtest.txt 1                                4s
Measuring performance for lseek system call with block size: 1
lseek: Performed 4194304 operations in 2.49 seconds.
Performance: 1.61 MiB/s, 1682993.33 B/s
File size: 4194304 bytes
Number of blocks read: 4194304
```

```
~/Desktop/OS Final Project/Final Version  ./run5a 8mbtest.txt 1
Measuring performance for read system call with block size: 1
Read: Performed 8388608 operations in 7.63 seconds.
Performance: 1.05 MiB/s, 1099010.93 B/s
File size: 8388608 bytes
Number of blocks read: 8388608
~/De/OS Final Project/Final Version  ./run5b 8mbtest.txt 1                                8s
Measuring performance for lseek system call with block size: 1
lseek: Performed 8388608 operations in 4.99 seconds.
Performance: 1.60 MiB/s, 1679936.36 B/s
File size: 8388608 bytes
Number of blocks read: 8388608
```

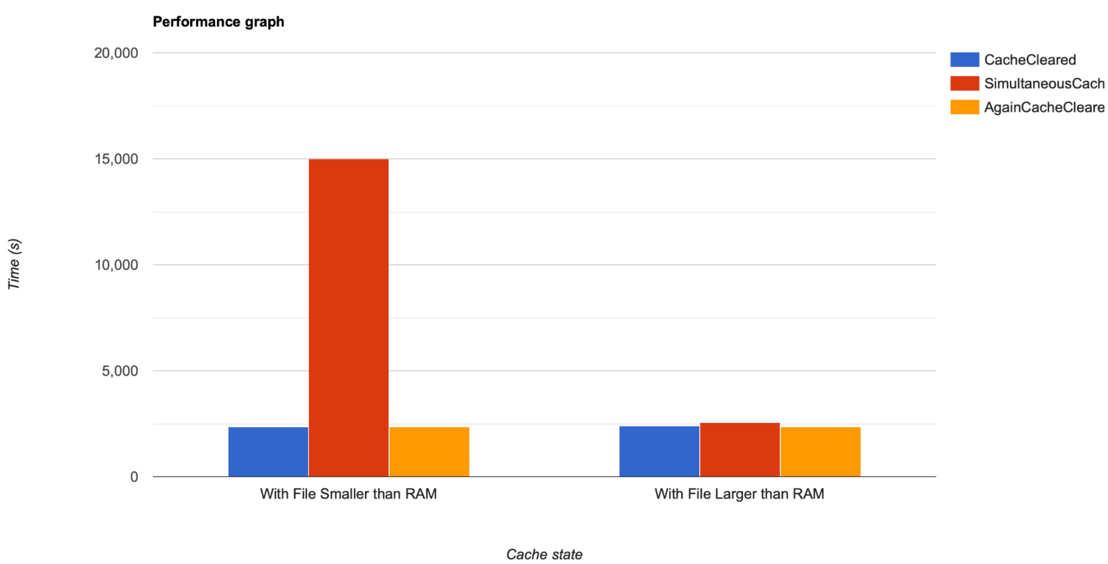
```
~/De/OS Final Project/Final Version  ./run5a 16mbtest.txt 1                               5s
Measuring performance for read system call with block size: 1
Read: Performed 16777216 operations in 15.27 seconds.
Performance: 1.05 MiB/s, 1098952.58 B/s
File size: 16777216 bytes
Number of blocks read: 16777216
~/De/O/Final Version  ./run5b 16mbtest.txt 1                                             15s
Measuring performance for lseek system call with block size: 1
lseek: Performed 16777216 operations in 10.04 seconds.
Performance: 1.59 MiB/s, 1671773.32 B/s
File size: 16777216 bytes
Number of blocks read: 16777216
```

Part 6: Raw performance

Here we did the optimization on code of “run3.cpp”, and renamed the new code to “fast.cpp”.
To improve the performance we looked for a good enough block, and for our system we got 65536 as our optimal block_size.
Also the use of Multiple thread inside our program utilizes the multiple cores of our system resulting in increase of speed.

Below is the report of performance:

	Speed in MiB/s		
	When cache is cleared (MiB/s)	Simultaneous run with cache (MiB/s)	Third run with cleared cache (MiB/s)
With File Smaller than RAM	2364	15003	2363
With File larger than RAM	2390	2566	2350



Graph for above reading

```
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
[sudo] password for techpertz:
~/Desktop/OS Final Project/Final Version  ./fast ubuntu-21.04-desktop-amd64.iso
Final XOR Result: a7eeb2d9
Performance: 2364.68 MiB/s
~/Desktop/OS Final Project/Final Version  ./fast ubuntu-21.04-desktop-amd64.iso
Final XOR Result: a7eeb2d9
Performance: 15003.2 MiB/s
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
~/Desktop/OS Final Project/Final Version  ./fast ubuntu-21.04-desktop-amd64.iso
Final XOR Result: a7eeb2d9
Performance: 2363.56 MiB/s
```

```
~/Desktop/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"
~/Desktop/OS Final Project/Final Version  ./fast 20test.txt
Final XOR Result: 1a5539b0
Performance: 2390.58 MiB/s
~/De/OS Final Project/Final Version  ./fast 20test.txt                                        9s
Final XOR Result: 1a5539b0
Performance: 2566.57 MiB/s
~/De/OS Final Project/Final Version  sudo sh -c "/usr/bin/echo 3 > /proc/sys/vm/drop_caches"  8s
~/Desktop/OS Final Project/Final Version  ./fast 20test.txt
Final XOR Result: 1a5539b0
Performance: 2350.36 MiB/s
```

Screenshot of readings

From the above graph and screenshots it is observed that the performance has increased a lot compared to run3.