



**VIT<sup>®</sup>**

**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

## **OPERATING SYSTEMS PROJECT**

**TITLE- ANALYSIS OF OPERATING SYSTEMS USING BENCHMARK  
TECHNIQUES**

### **GROUP MEMBERS-**

Sriyog Yogesh Dixit 19BCE2329

Balaji P.J 19BCE2306

Rohan H.R 19BCE2314

### **SLOT-**

L5+L6

**VIT<sup>®</sup>****Vellore Institute of Technology**

(Deemed to be University under section 3 of UGC Act, 1956)

## **School of Computer Science and Engineering**

### **DECLARATION**

I hereby declare that the project entitled “**Analysis Of Operating System Using Benchmark Techniques**” submitted by me to the School of Computer Science and Engineering, Vellore Institute of Technology, Vellore-14 towards the partial fulfillment of the requirements for the course CSE2005- Operating Systems is a record of bonafide work carried out by me under the supervision of **Prof. Jothi K.R.** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for any other course or purpose of this institute or of any other institute or university.

Signature

**Name-** Sriyog Yogesh Dixit**Reg.No-** 19BCE2329



**VIT<sup>®</sup>**  
**Vellore Institute of Technology**  
(Deemed to be University under section 3 of UGC Act, 1956)

**School of Computer Science and Engineering**  
**CERTIFICATE**

The project report entitled “**ANALYSIS OF OPERATING SYSTEMS USING BENCHMARK TECHNIQUES**” is prepared and submitted by **Sriyog Yogesh Dixit (Register No: 19BCE2329), Balaji P.J(Register No:19BCE2306), Rohan H.R(Register No:19BCE2314)** has been found satisfactory in terms of scope, quality and presentation as partial fulfilment of the course CSE2005-Operating Systems in Vellore Institute of Technology, Vellore-14, India.

**Guide**  
**(Name & Signature)**

## ACKNOWLEDGEMENT

The project “**ANALYSIS OF OPERATING SYSTEMS USING BENCHMARK TECHNIQUES**” was made possible because of inestimable inputs from everyone involved, directly or indirectly. I would first like to thank my guide, **Prof Jothi K.R**, who was highly instrumental in providing not only a required and innovative base for the project but also crucial and constructive inputs that helped make my final product. My guide has helped me perform research in the specified area and improve my understanding in the area of web development and internet of things and I am very thankful for her support all throughout the project.

I would also like to acknowledge the role of the HOD, **Dr. Santhi V.** , who was instrumental in keeping me updated with all necessary formalities and posting all the required formats and document templates through the mail, which I was glad to have had.

It would be no exaggeration to say that the Dean of SCOPE, **Dr. Saravanan R**, was always available to clarify any queries and clear the doubts I had during the course of my project.

Finally, I would like to thank **Vellore Institute of Technology**, for providing me with a flexible choice and execution of the project and for supporting my research and execution related to the project.

## TABLE OF CONTENT

CHAPTER	TOPIC	PAGE
1. ABSTRACT		1
2. INTRODUCTION	2.1 CPU BENCHMARK	2
	2.2 MEMORY BENCHMARK	3
	2.3 RAM BENCHMARK	3
3. LITERATURE SURVEY	3.1 Benchmarking and operational performance	4
	3.2 Theoretical peak FLOPS per instruction set	4
	3.3 DATA FILE HANDLING IN C++	5
4. DESIGN		6
5.IMPLEMENTATION	5.1 TOOLS USED	7
	5.2 CODE LANGUAGE	7
	5.3 OUTPUT SCREENSHOTS	8-13
6.RESULTS AND DISCUSSION	6.1 CPU BENCHMARK	14-16
	6.2 RAM BENCHMARK	17-20
	6.3 MEMORY BENCHMARK	21-23
7.CONCLUSION		24
8. REFERENCES		25

## 1. ABSTRACT

Operating System is an essential component of every electronic device which enables hardware to communicate and operate with the software i.e interlinking the hardware and software.

Without the Operating System the device and the software programs would be meaningless. Earlier the computers had humans as the operating system and required manual interruption to perform the operations or programs. With the evolving technology this has been changed to a software and the operations are performed with the inbuilt function.

There are many types of Operating Systems such as Windows, MS-DOS, Linux, Unix, MacOS and many more which have been constantly modified to give better performance and satisfy the user requirements.

To understand the performance of these Operating Systems Benchmarking Techniques gives us a clear understanding and accurate results to choose the Best Operating Systems in the market. This project report would give a clear result between the Windows OS and the Linux OS by comparing the various parameters.

## 2. INTRODUCTION

Benchmarking is a process of establishing a standard of excellence and is a medium of comparison between product/enterprise(in our case operating systems). The main aim of benchmarking is increasing the standard which ultimately improves the product and service quality. System Performance varies enormously from one application domain to another and the load on various components of the system also affect the performance considerably.

We are performing the benchmark on a Dual Boot Computer with Windows and Ubuntu. This would give the accurate results for Operating System performance as same resources are allotted to both the Operating Systems in the Computer.

The Parameters to analyse the performance of the Operating Systems are as follows-

- CPU Benchmarking
- Memory Benchmarking
- Disk Benchmarking

### 2.1. CPU Benchmarking-

In CPU benchmarking the aim is to calculate processor speed in terms of Floating Point Operations Per Second (FLOPS) and Integer Operations Per Second (IOPS). The program runs multiple instructions concurrently and gives the Execution time , FLOS and IOPS for 1 Giga iterations.

Two functions are created to calculate the GFLOPS and GIOPS respectively. Time taken to perform operations is calculated using clock() function using the header file “time.h”. The time difference thus obtained is divided by CLOCKS\_PER\_CYCLE giving time in seconds. Number of (ITERATIONS \* No of threads \* No of operations) are divided by calculated time to determine processor speed in terms of Giga FLOPS and Giga IOPS.

We have also included a 10min benchmark test which gives the values of floating point operation and integer operation every second using 4 threads at a time. This would give a clear result and the data is stored in a .txt file.

## **2.2 Memory Benchmarking-**

In the memory benchmark two operations are performed on memory, read and write. Read and write operations are performed sequentially and randomly with a given block size. Memory location is created using malloc function and a file containing data is transferred from one location to other.

Sequential read and write performs the function line by line in an order whereas the random read and write copies the data by generating a random memory location using the rand() function.

Three block sizes have been given 1B,1KB,1MB to set the transfer size at a time. The output of this benchmark would give the latency and throughput of the Operating System's Memory manager.

## **2.3 RAM Benchmarking-**

In RAM Benchmark the aim is to calculate the performance of Operating System handling the RAM on the basis of time required to create and delete data files stored on the disk.

We have used the concept of file handling to create, append and write data and using the time required for these operations and have compared this in different Operating Systems.

Three block sizes 1B,1KB,1MB have been used to set size of the data handled in one operation at a time.

We have used clock() function to determine the latency and the throughput of the operation. Difference of start time and end time is divided by CLOCKS\_PER\_CYCLE to calculate latency and throughput.



### 3. LITERATURE SURVEY

We have used several concepts by referring few research papers and text books related to benchmarking and Operating Systems. Few research papers referred are mentioned below-

#### 3.1 ‘Benchmarking and operational performance’ Author : CA Voss

**Abstract-** This deals with the benchmarking and operational performance using a sample of over 600 European manufacturing sites. Benchmarking is linked to the identification and adoption of improved operational practices, an increased understanding of competitive positioning, and the larger context of the “learning organization”. Shows that benchmarking may indeed contribute to improved operational performance, first through improving the firm’s understanding of its competitive position and its strengths and weaknesses, and second through providing a systematic process for effecting change. Learning organizations were more likely to benchmark than other firms.

#### 3.2 ‘Theoretical peak FLOPS per instruction set’ Author : Romain Dolbeau

**Abstract-**Traditionally, evaluating the theoretical peak performance of a CPU in FLOPS (floating-point operations per second) was merely a matter of multiplying the frequency by the number of floating-point instructions per cycle. Today however, CPUs have features such as vectorization, fused multiply-add, hyperthreading, and “turbo” mode. In this tutorial, we look into this theoretical peak for recent fully featured Intel CPUs and other hardware, taking into account not only the simple absolute peak, but also the relevant instruction sets, encoding and the frequency scaling behaviour of modern hardware.

### **3.3 'DATA FILE HANDLING IN C++' Author : Kripashanker Yadav**

**Abstract-**Data file handling has a very vital role. This is because files help in storing data permanently as well modifying or deleting data from the files. This paper contains introduction to data files, opening and closing files using constructors and open() function, sequential I/O with files, detecting EOF, file pointers and random access and basic binary operations on files (including searching, appending data, inserting data, deleting data and modifying data) with error handling during file I/O.

## 4.DESIGN

The Basic Work Flow of the Benchmark using the three parameters is shown in the below figure-

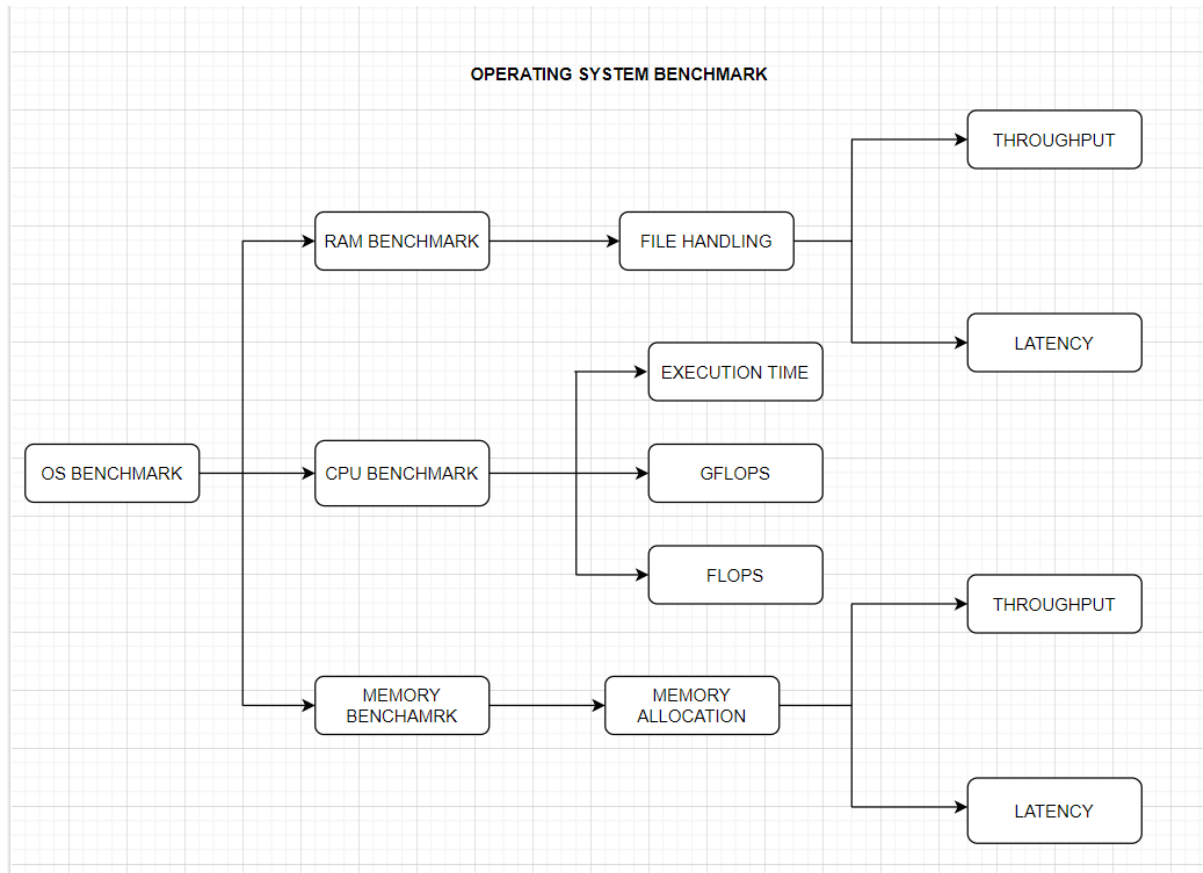


Fig No-4.1- Design Of Benchmark Program

## **5. IMPLEMENTATION**

### **5.1 Tools Used-**

We have used a dual booted computer with Windows OS and Linux.

Specification of System:-

Model: Dell Inspiron 7373

OS: Ubuntu 20.04.1 / Windows 10 Single

CPU: Intel i7-8550U(4 Cores & 8 Threads)

CPU Speed: 4.00GHz

GPU: Intel UHD Graphics 620

Memory: 8 GB RAM & 512GB ROM

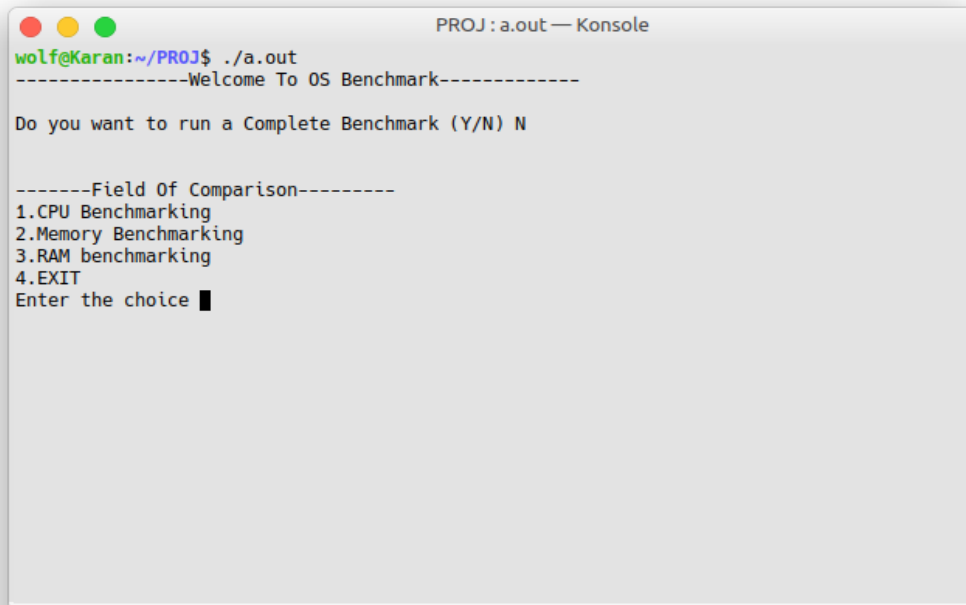
IPC: 8 approx.

### **5.2 Code Language-**

The code is implemented in C programming language using Code Blocks in Windows and in Ubuntu it is implemented in terminal using gcc compiler.

## 5.3 Output Screenshots-

### 5.3.1 For Ubuntu OS-

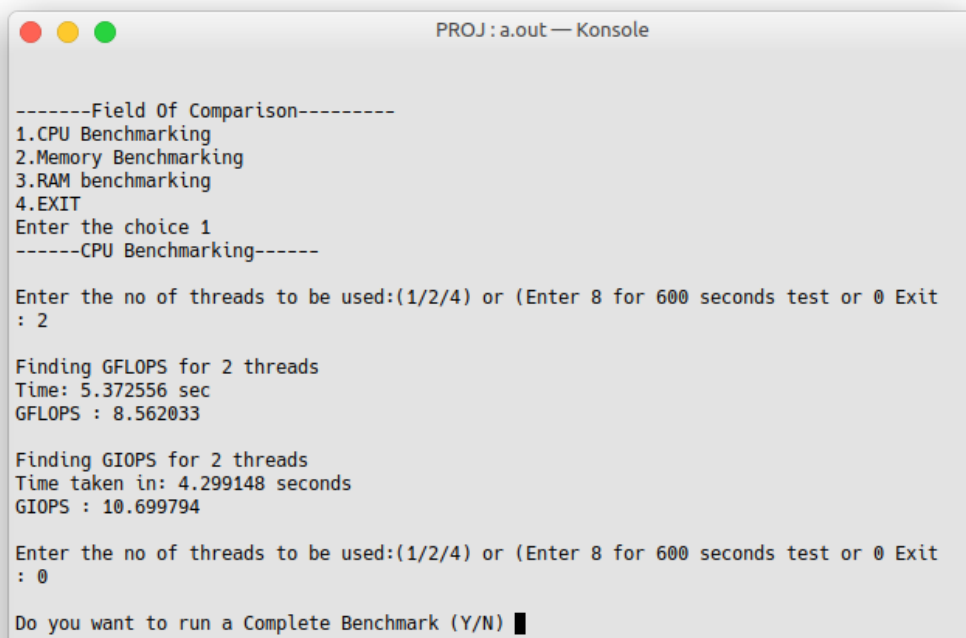


```
PROJ: a.out — Konsole
wolfeKaran:~/PROJ$ ./a.out
-----Welcome To OS Benchmark-----

Do you want to run a Complete Benchmark (Y/N) N

-----Field Of Comparison-----
1.CPU Benchmarking
2.Memory Benchmarking
3.RAM benchmarking
4.EXIT
Enter the choice █
```

Fig No-5.3.1-Main Benchmark



```
PROJ: a.out — Konsole

-----Field Of Comparison-----
1.CPU Benchmarking
2.Memory Benchmarking
3.RAM benchmarking
4.EXIT
Enter the choice 1
-----CPU Benchmarking-----

Enter the no of threads to be used:(1/2/4) or (Enter 8 for 600 seconds test or 0 Exit
: 2

Finding GFLOPS for 2 threads
Time: 5.372556 sec
GFLOPS : 8.562033

Finding GIOPS for 2 threads
Time taken in: 4.299148 seconds
GIOPS : 10.699794

Enter the no of threads to be used:(1/2/4) or (Enter 8 for 600 seconds test or 0 Exit
: 0

Do you want to run a Complete Benchmark (Y/N) █
```

Fig No-5.3.2-CPU Benchmark

```
PROJ : a.out — Konsole

-----Memory Benchmarking-----

1.BYTE
2.KILOBYTE
3.MEGABYTE
4.EXIT
Enter the Block size
1

Enter the number of threads(1/2) :
1

Byte Read+Write for 1 thread
--Sequential Read+Write--
-----
Latency : 0.431600 ms
Throughput:2316.960148 MB/s

--Random Read+Write--
-----
Latency : 2.760000 ms
Throughput:362.318841 MB/s

1.BYTE
2.KILOBYTE
3.MEGABYTE
4.EXIT
```

Fig No-5.3.3-Memory Benchmark for 1B

```
PROJ : a.out — Konsole

Enter the Block size
2

Enter the number of threads(1/2) :
1

KiloByte Read+Write for 1 thread
--Sequential Read+Write--
-----
Latency : 0.685451 ms
Throughput:1458.893529 MB/s

--Random Read+Write--
-----
Latency : 0.713745 ms
Throughput:1401.060410 MB/s

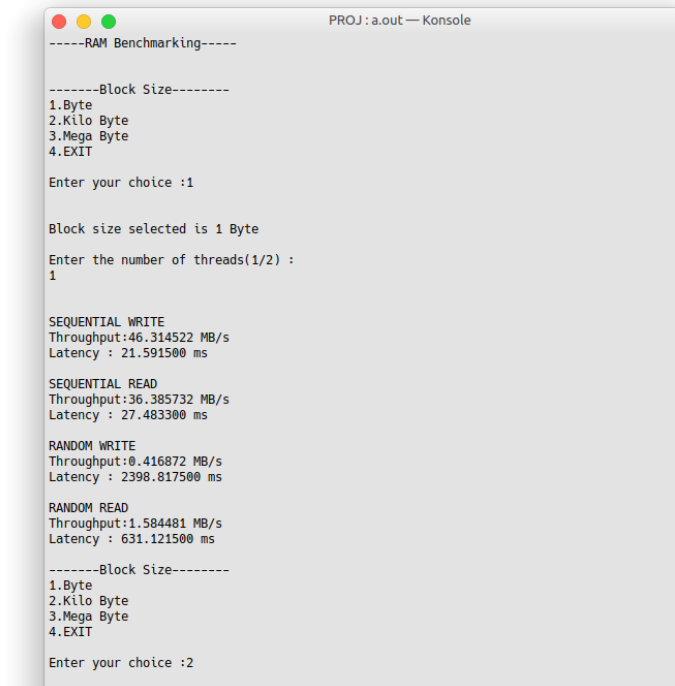
1.BYTE
2.KILOBYTE
3.MEGABYTE
4.EXIT
Enter the Block size
3

Enter the number of threads(1/2) :
1

MegaByte read+write for 1 thread
--Sequential Read+Write--
-----
Latency : 7.440800 ms
Throughput:134.394151 MB/s

--Random read+write--
-----
Latency : 4.305300 ms
Throughput:232.271851 MB/s
```

Fig No-5.3.4-Memory Benchmark for 1KB and 1MB



```

PROJ: a.out — Konsole

-----RAM Benchmarking-----

-----Block Size-----
1.Byte
2.Kilo Byte
3.Mega Byte
4.EXIT

Enter your choice :1

Block size selected is 1 Byte

Enter the number of threads(1/2) :
1

SEQUENTIAL WRITE
Throughput:46.314522 MB/s
Latency : 21.591500 ms

SEQUENTIAL READ
Throughput:36.385732 MB/s
Latency : 27.483300 ms

RANDOM WRITE
Throughput:0.416872 MB/s
Latency : 2398.817500 ms

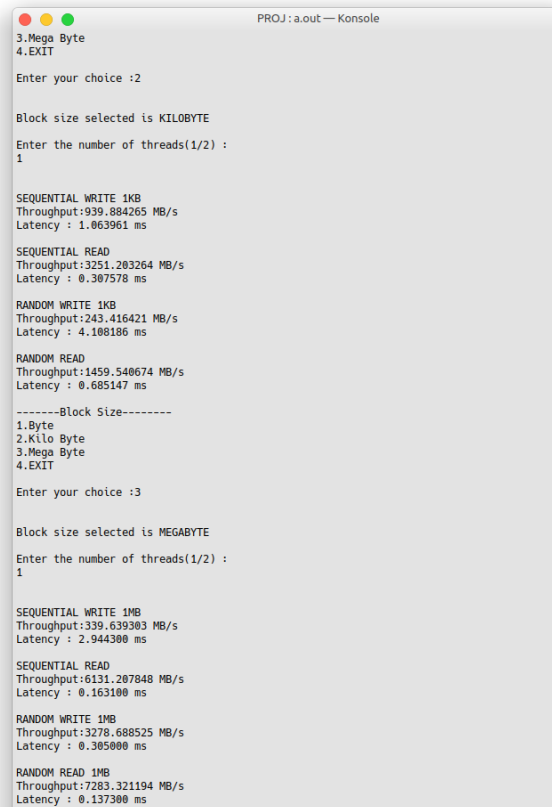
RANDOM READ
Throughput:1.584481 MB/s
Latency : 631.121500 ms

-----Block Size-----
1.Byte
2.Kilo Byte
3.Mega Byte
4.EXIT

Enter your choice :2

```

Fig No-5.3.5-RAM Benchmark for 1B



```

PROJ: a.out — Konsole

3.Mega Byte
4.EXIT

Enter your choice :2

Block size selected is KILOBYTE

Enter the number of threads(1/2) :
1

SEQUENTIAL WRITE 1KB
Throughput:939.884265 MB/s
Latency : 1.063961 ms

SEQUENTIAL READ
Throughput:3251.203264 MB/s
Latency : 0.307578 ms

RANDOM WRITE 1KB
Throughput:243.416421 MB/s
Latency : 4.108186 ms

RANDOM READ
Throughput:1459.540674 MB/s
Latency : 0.685147 ms

-----Block Size-----
1.Byte
2.Kilo Byte
3.Mega Byte
4.EXIT

Enter your choice :3

Block size selected is MEGABYTE

Enter the number of threads(1/2) :
1

SEQUENTIAL WRITE 1MB
Throughput:339.639303 MB/s
Latency : 2.944300 ms

SEQUENTIAL READ
Throughput:6131.207848 MB/s
Latency : 0.163100 ms

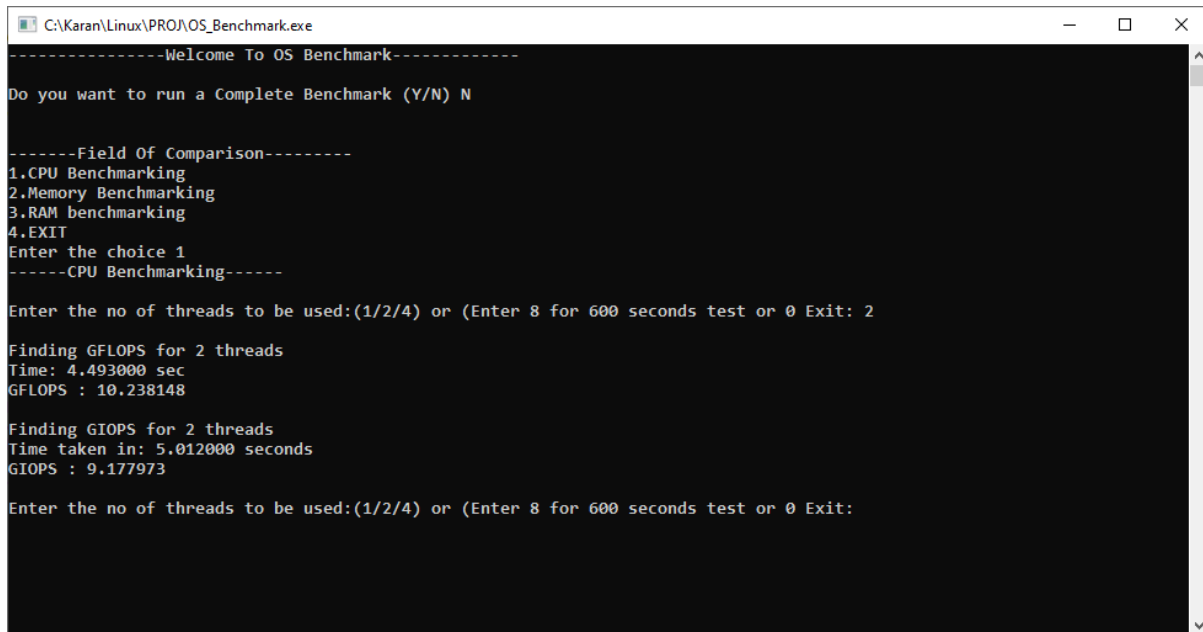
RANDOM WRITE 1MB
Throughput:3278.688525 MB/s
Latency : 0.305000 ms

RANDOM READ 1MB
Throughput:7283.321194 MB/s
Latency : 0.137300 ms

```

Fig No-5.3.6-RAM Benchmark for 1KB,1MB

### 5.3.2. For Windows OS-



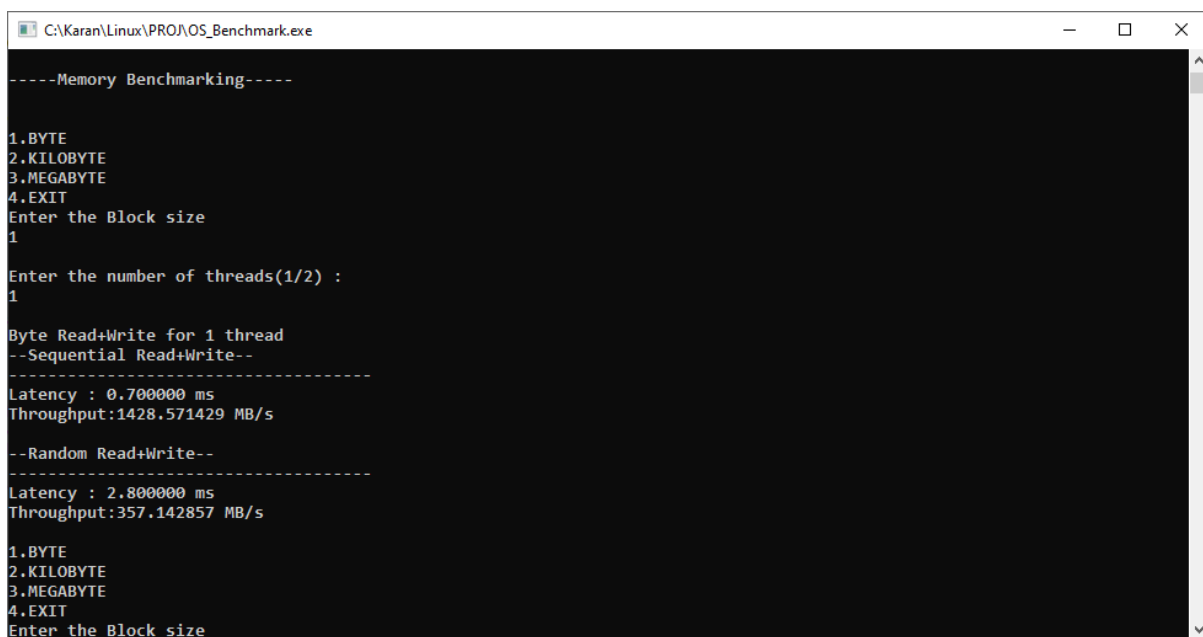
```

C:\Karan\Linux\PRO\OS_Benchmark.exe
-----Welcome To OS Benchmark-----
Do you want to run a Complete Benchmark (Y/N) N

-----Field Of Comparison-----
1.CPU Benchmarking
2.Memory Benchmarking
3.RAM benchmarking
4.EXIT
Enter the choice 1
-----CPU Benchmarking-----
Enter the no of threads to be used:(1/2/4) or (Enter 8 for 600 seconds test or 0 Exit: 2
Finding GFLOPS for 2 threads
Time: 4.493000 sec
GFLOPS : 10.238148
Finding GIOPS for 2 threads
Time taken in: 5.012000 seconds
GIOPS : 9.177973
Enter the no of threads to be used:(1/2/4) or (Enter 8 for 600 seconds test or 0 Exit:

```

Fig No-5.3.7-CPU Benchmark for WIN



```

C:\Karan\Linux\PRO\OS_Benchmark.exe
-----Memory Benchmarking-----
1.BYTE
2.KILOBYTE
3.MEGABYTE
4.EXIT
Enter the Block size
1
Enter the number of threads(1/2) :
1
Byte Read+Write for 1 thread
--Sequential Read+Write--
-----
Latency : 0.700000 ms
Throughput:1428.571429 MB/s
--Random Read+Write--
-----
Latency : 2.800000 ms
Throughput:357.142857 MB/s
1.BYTE
2.KILOBYTE
3.MEGABYTE
4.EXIT
Enter the Block size

```

Fig No-5.3.8-Memory Benchmark for 1 byte WIN



```

C:\Karan\Linux\PROJ\OS_Benchmark.exe
Enter the Block size
2
Enter the number of threads(1/2) :
1

KiloByte Read+Write for 1 thread
--Sequential Read+Write--
-----
Latency : 1.166667 ms
Throughput:857.142857 MB/s

--Random Read+Write--
-----
Latency : 0.098039 ms
Throughput:10200.000000 MB/s

1.BYTE
2.KILOBYTE
3.MEGABYTE
4.EXIT
Enter the Block size
3
Enter the number of threads(1/2) :
1

MegaByte read+write for 1 thread
--Sequential Read+Write--
-----
Latency : 11.600000 ms
Throughput:86.206897 MB/s

--Random read+write--
-----
Latency : 1.400000 ms
Throughput:714.285714 MB/s

1.BYTE
2.KILOBYTE
3.MEGABYTE
4.EXIT
Enter the Block size

```

Fig No-5.3.9-Memory Benchmark for 1KB,1MB WIN

```

C:\Karan\Linux\PROJ\OS_Benchmark.exe
-----RAM Benchmarking-----

-----Block Size-----
1.Byte
2.Kilo Byte
3.Mega Byte
4.EXIT
Enter your choice :1

Block size selected is 1 Byte

Enter the number of threads(1/2) :
1

SEQUENTIAL WRITE
Throughput:28.818444 MB/s
Latency : 34.700000 ms

SEQUENTIAL READ
Throughput:18.115942 MB/s
Latency : 55.200000 ms

RANDOM WRITE
Throughput:0.255768 MB/s
Latency : 3909.800000 ms

RANDOM READ
Throughput:0.179973 MB/s
Latency : 5556.400000 ms

-----Block Size-----
1.Byte
2.Kilo Byte
3.Mega Byte
4.EXIT
Enter your choice :2

Block size selected is KILOBYTE

```

Fig No-5.3.10-RAM Benchmark for 1 byte WIN

```
C:\Karan\Linux\PROJ\OS_Benchmark.exe
Block size selected is KILOBYTE
Enter the number of threads(1/2) :
1

SEQUENTIAL WRITE 1KB
Throughput:214.736842 MB/s
Latency : 4.656863 ms

SEQUENTIAL READ
Throughput:528.497409 MB/s
Latency : 1.892157 ms

RANDOM WRITE 1KB
Throughput:225.165563 MB/s
Latency : 4.441176 ms

RANDOM READ
Throughput:117.919075 MB/s
Latency : 8.480392 ms

-----Block Size-----
1.Byte
2.Kilo Byte
3.Mega Byte
4.EXIT
Enter your choice :3

Block size selected is MEGABYTE
Enter the number of threads(1/2) :
1

SEQUENTIAL WRITE 1MB
Throughput:227.272727 MB/s
Latency : 4.400000 ms

SEQUENTIAL READ
Throughput:769.230769 MB/s
Latency : 1.300000 ms
```

Fig No-5.3.11-RAM Benchmark for 1 KB WIN

```
C:\Karan\Linux\PROJ\OS_Benchmark.exe
Enter your choice :3

Block size selected is MEGABYTE
Enter the number of threads(1/2) :
1

SEQUENTIAL WRITE 1MB
Throughput:227.272727 MB/s
Latency : 4.400000 ms

SEQUENTIAL READ
Throughput:769.230769 MB/s
Latency : 1.300000 ms

RANDOM WRITE 1MB
Throughput:416.666667 MB/s
Latency : 2.400000 ms

RANDOM READ 1MB
Throughput:909.090909 MB/s
Latency : 1.100000 ms

-----Block Size-----
1.Byte
2.Kilo Byte
3.Mega Byte
4.EXIT
Enter your choice :
```

Fig No-5.3.12-RAM Benchmark for 1 MB WIN

## 6.RESULTS AND DISCUSSION

### 6.1 CPU Benchmark-

#### 6.1.1 Ubuntu OS-

Theoretical Peak Performance = (CPU Speed \* No of cores \* Instructions per cycle)  
= 128 GFLOPS

Efficiency = (FLOPS for 1 thread /Theoretical Peak Performance) \*100

For 2 Thread-

a)GFLOP = 8.562033

b)TIME = 5.372556 s

Efficiency for GFLOP = 13.3%

c)GIOPS = 10.699794

d)TIME = 4.299148 s

Efficiency for GIOPS = 16.6%

#### 6.1.2 Windows OS-

Theoretical Peak Performance = (CPU Speed \* No of cores \* Instructions per cycle)  
= 128 GFLOPS

Efficiency = (FLOPS for 1 thread /Theoretical Peak Performance) \*100

For 2 Thread-

a)GFLOP = 10.238

b)TIME = 4.493 s

Efficiency for GFLOP = 15.9%

c)GIOPS = 9.177

d)TIME = 5.012 s

Efficiency for GIOPS = 14.3%

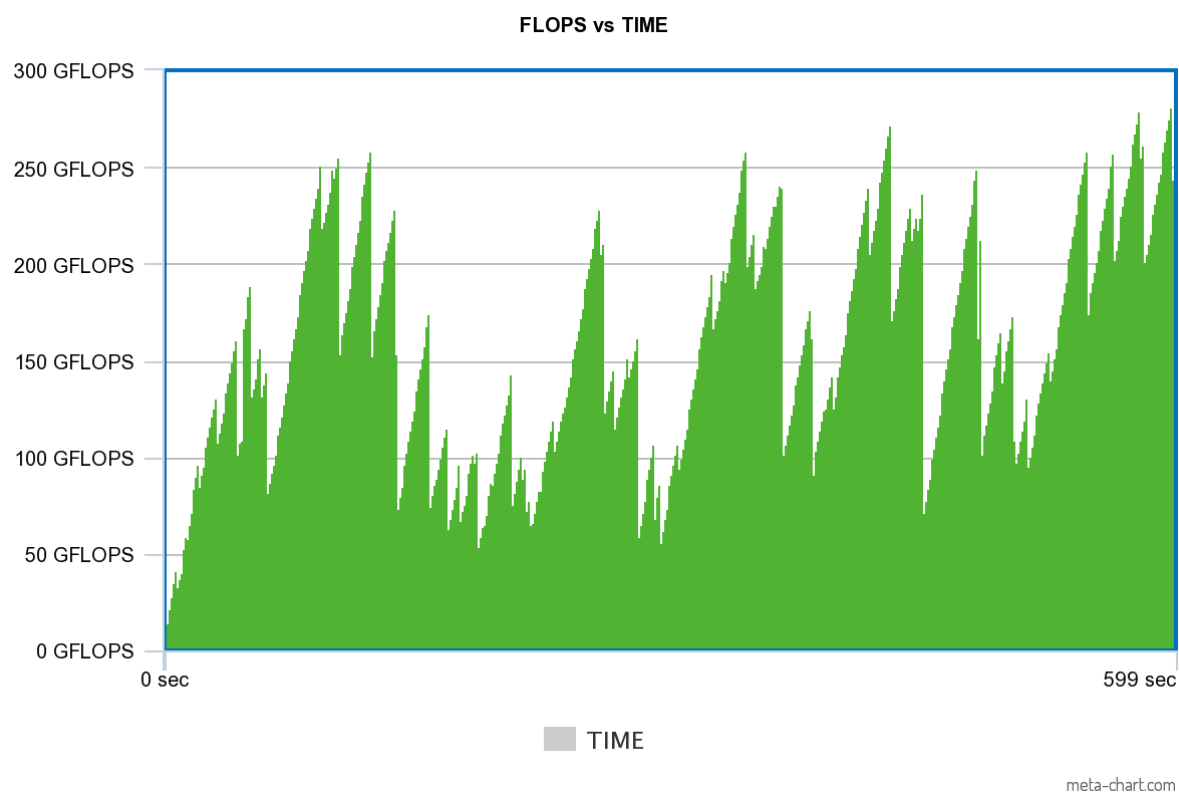


Fig.No-6.1.1-FLOPS vs Time in ubuntu

Observation-Max at 596<sup>th</sup> sec with 281.06 GFLOPS

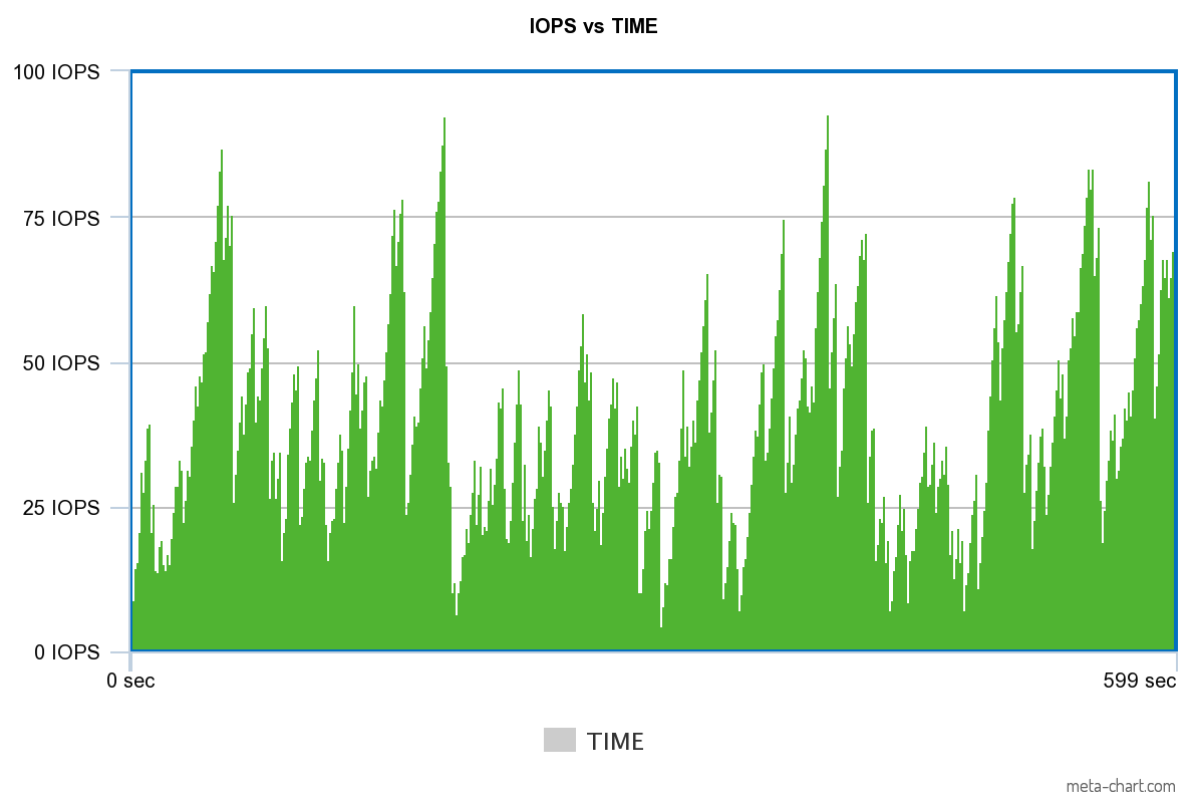


Fig No-6.1.2-IOPS vs Time for Ubuntu

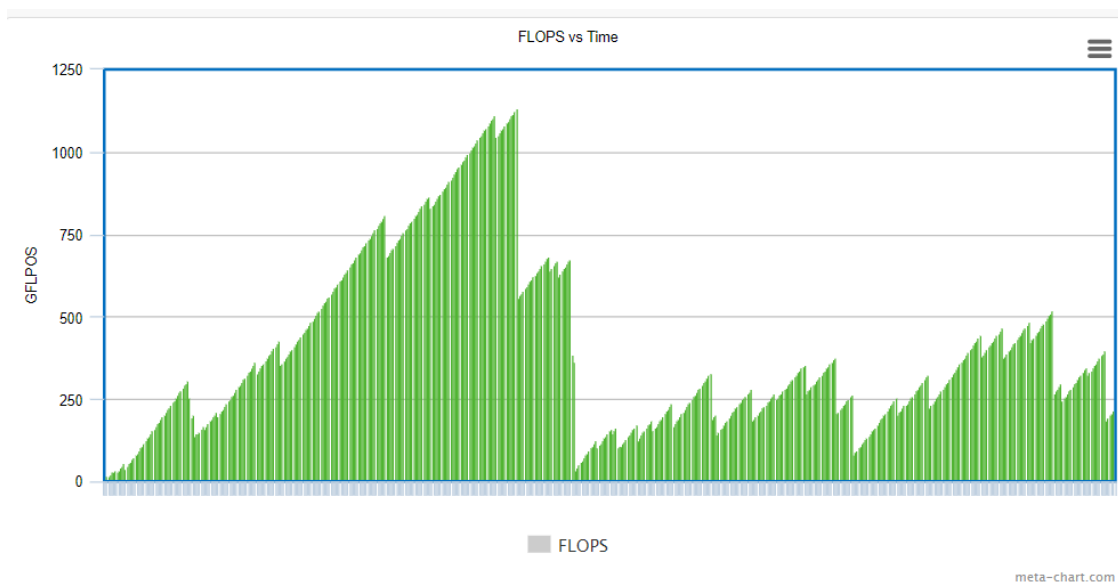
Observation-Max at 399<sup>th</sup> sec with 92.758 GIOPS

Fig.No-6.1.3-FLOPS vs Time in Windows

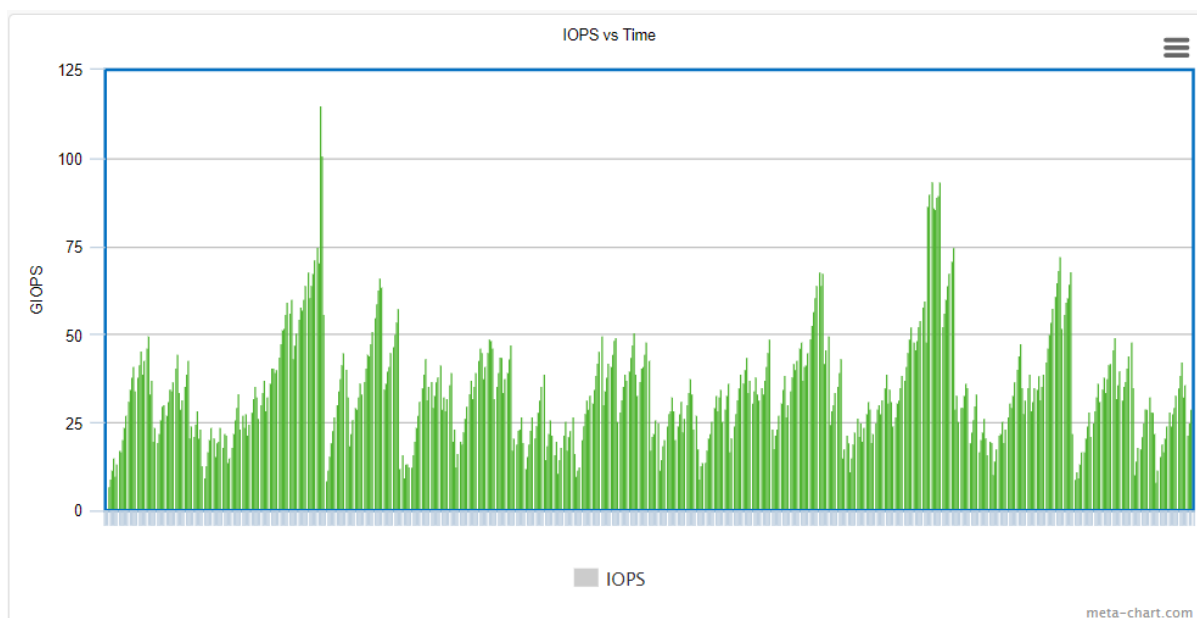
Observation-Max at 244<sup>th</sup> sec with 1132.642 GFLOPS

Fig No-6.1.4-IOPS vs Time for Windows

Observation-Max at 118<sup>th</sup> sec with 115.16 GIOPS

## 6.2 RAM Benchmark-

### 6.2.1 For Ubuntu-

For 1 Thread-

Sequential Write	Throughput	Latency
1 Byte	46.314MB/s	21.51 ms
1 KiloByte	939.88MB/s	1.063 ms
1 MegaByte	339.639MB/s	2.944 ms

Sequential Read	Throughput	Latency
1 Byte	36.385MB/s	27.483 ms
1 KiloByte	3251.20MB/s	0.307 ms
1 MegaByte	6131.207MB/s	0.16 ms

Random Read	Throughput	Latency
1 Byte	1.584MB/s	631.121 ms
1 KiloByte	1459.54MB/s	0.685 ms
1 MegaByte	7283.32 MB/s	0.137 ms

Random Write	Throughput	Latency
1 Byte	0.4416MB/s	2398.817 ms
1 KiloByte	243.416MB/s	4.10 ms
1 MegaByte	3278.688MB/s	0.305 ms

## 6.2.2 For Windows-

### For 1 Thread-

Sequential Write	Throughput	Latency
1 Byte	28.818 MB/s	34.7 ms
1 KiloByte	214.736 MB/s	4.65 ms
1 MegaByte	227.27 MB/s	4.4 ms

Sequential Read	Throughput	Latency
1 Byte	18.115 MB/s	55.2ms
1 KiloByte	528.49 MB/s	1.892ms
1 MegaByte	769.23MB/s	1.3 ms

Random Write	Throughput	Latency
1 Byte	0.255 MB/s	3909.8 ms
1 KiloByte	225.165MB/s	4.441ms
1 MegaByte	416.66 MB/s	2.4 ms

Random Read	Throughput	Latency
1 Byte	0.179 MB/s	5556.4 ms
1 KiloByte	117.91MB/s	8.48 ms
1 MegaByte	909.09 MB/s	1.1ms

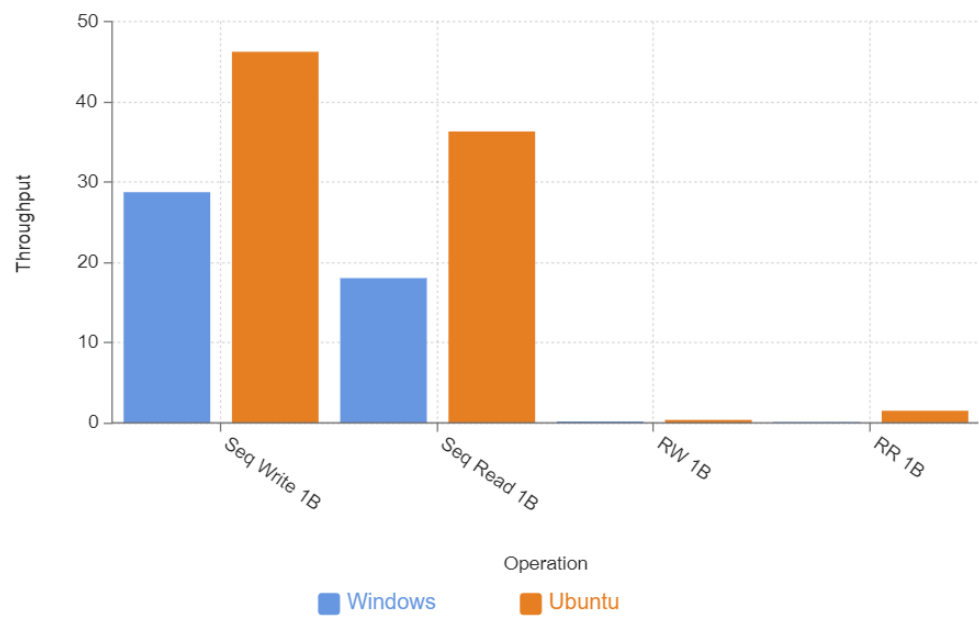


Fig.No-6.2.1-RAM BENCHMARK(Byte)

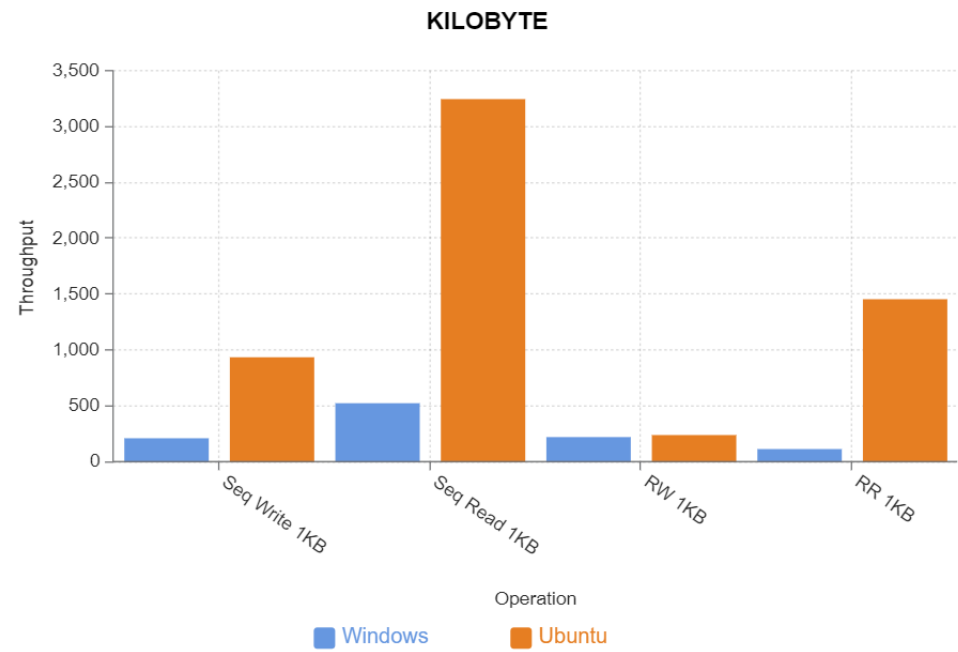


Fig.No-6.2.2-RAM BENCHMARK(KiloByte)



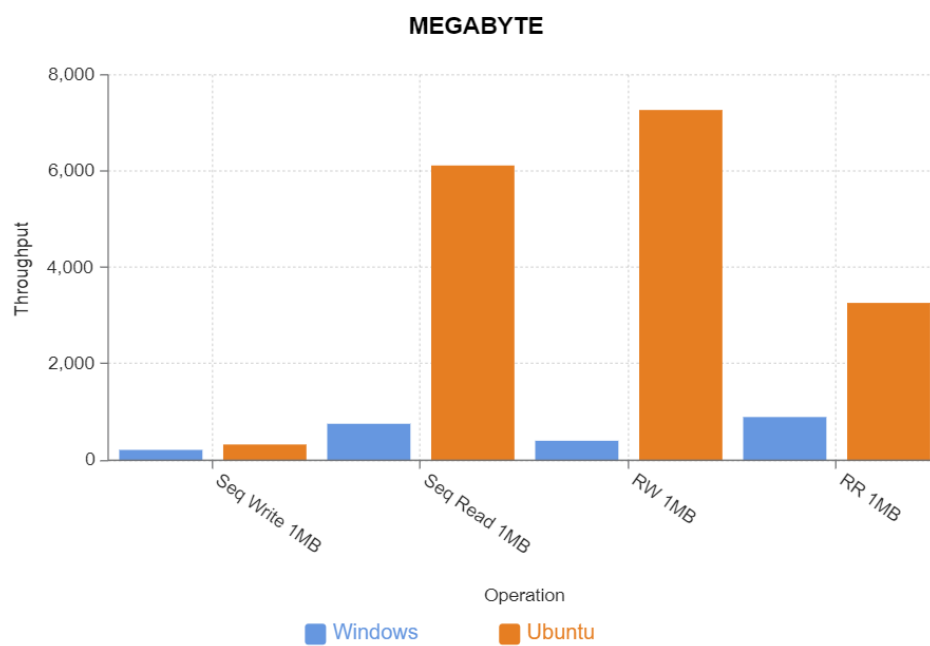


Fig.No-6.2.3-RAM BENCHMARK(MegaByte)

## 6.3.Memory Benchmark-

### 6.3.1. For Ubuntu-

For 1 Thread-

Sequential Read and Write	Throughput	Latency
1 Byte	2316.960148MB/s	0.4316ms
1 KiloByte	458.89MB/s	0.6854ms
1 MegaByte	134.394MB/s	7.4408ms

Random Read and Write	Throughput	Latency
1 Byte	362.3188MB/s	2.76ms
1 KiloByte	401.061MB/s	0.713ms
1 MegaByte	232.27MB/s	4.305ms

### 6.3.2. For Windows-

For 1 Thread-

Sequential Read and Write	Throughput	Latency
1 Byte	1428.57 MB/s	0.7 ms
1 KiloByte	857.14 MB/s	1.16ms
1 MegaByte	86.2 MB/s	11.6ms

Random Read and Write	Throughput	Latency
1 Byte	357.14MB/s	2.8ms

1 KiloByte	10200MB/s	0.09ms
1 MegaByte	714.28MB/s	1.4ms

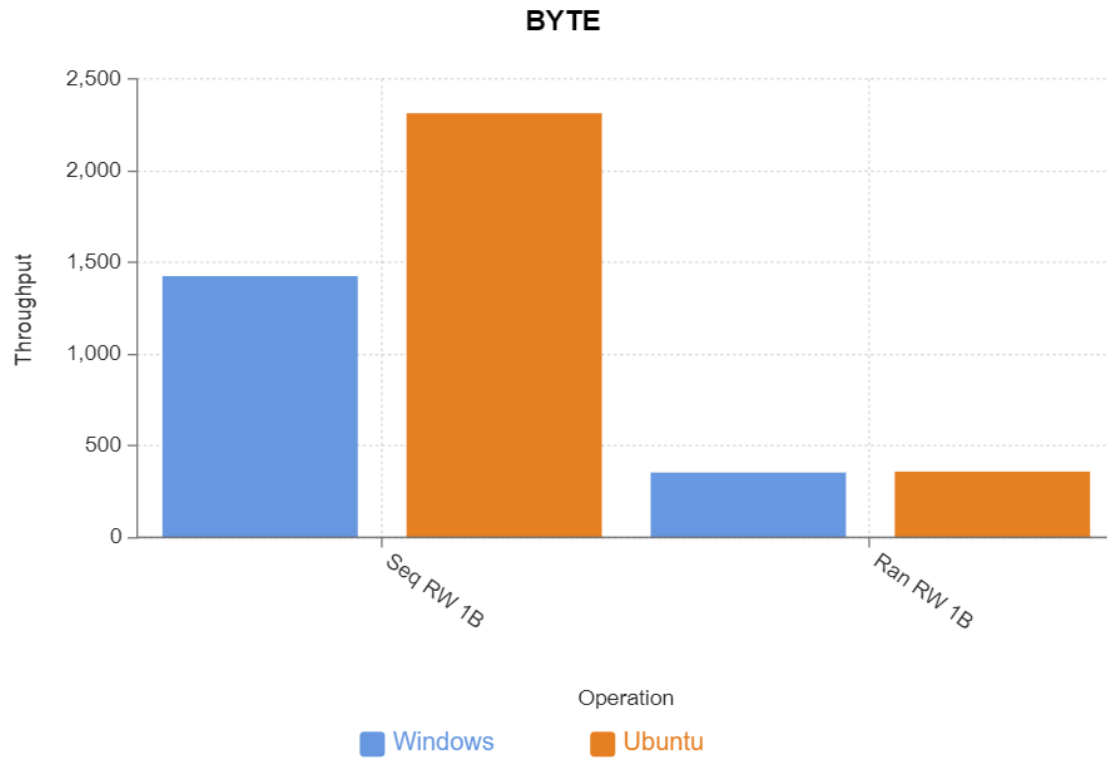


Fig.No-6.3.1-MEMORY BENCHMARK(Byte)

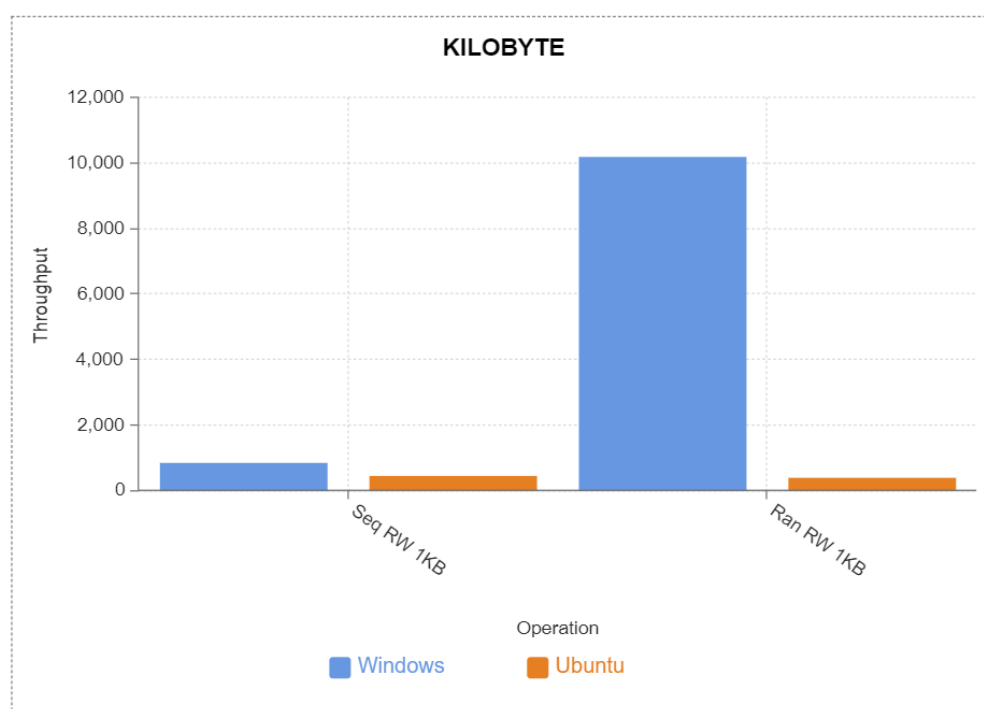


Fig.No-6.3.2-MEMORY BENCHMARK(KiloByte)

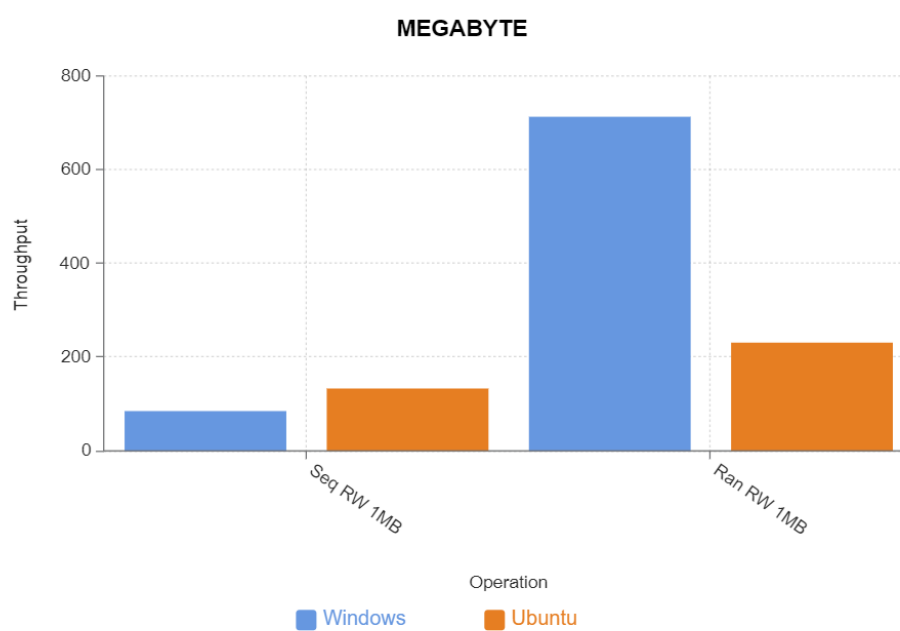


Fig.No-6.3.3-MEMORY BENCHMARK(MegaByte)

## 7. CONCLUSION

After performing the Benchmark we conclude the following –

- In terms of CPU Utilization Windows hit a higher peak performance compared to Ubuntu but as seen from the charts of 600 sec test Ubuntu could consistently attain high peak performance over long time while windows achieved the peak once.
- Ubuntu is able to perform better on a more consistent level for handling CPU
- In terms of RAM Management we can conclude that Ubuntu has a upper hand on handling the RAM efficiently as Ubuntu is much lighter OS than Windows hence has a lower base RAM usage .
- In terms of Memory Management we can conclude that Windows has a upper hand on handling the Memory efficiently .

Finally we can conclude that Both have there pros and cons but Windows being more consumer friendly and secure. Windows is better than Ubuntu as it has a better memory manager and has higher peak in the Floating Point Operation per Second as well as in Integer Operation per Second.

## 8. REFERENCES

- Voss, C.A., Åhlström, P. and Blackmon, K. (1997), "Benchmarking and operational performance: some empirical results", *International Journal of Operations & Production Management*, Vol. 17 No. 10, pp. 1046-1058.
- Dolbeau, Romain. (2017). Theoretical peak FLOPS per instruction set: a tutorial. *The Journal of Supercomputing*. 74. 10.1007/s11227-017-2177-5.
- Kripashanker Yadav, (2015) DATA FILE HANDLING IN C++ *IJIRT*, Volume 2 Issue 6, ISSN: 2349-6002