# Performance Evaluation

The document has the experimental result of all the benchmarks. Our experiments cover all the original parameter spaces required in each bench mark. Each experiment is done three times and the average and standard deviation are shown in the figures. First, we describe the environment, then the specification of the experiments of each benchmark is presented, and at last all the results will be shown and analyzed.

## 1. Experimental Environments

For all benchmark, we did our experiments on MAC/Linux-Ubuntu(External Benchmark), which has intel i7 processor, with 4 cores, DDR3 RAM of 8GB, GPU Nvdia Gforce GT 650M, DISK 256GB SSD.

## 2. Overall Experiments

For CPU benchmark, we measured processor speed FLOPs and IOPs at a varying concurrency level(1,2,4,8 threads).

For GPU benchmark, we measured GPU speed in FLOPs and IOPs with full concurrency and also the read and write memory bandwidth with different message size (1B, 1KB, 1MB).

For Memory benchmark, we measured the throughputs ( sequential read, sequential write, random read and random write) and latency(random read, random write). We varied the block size(1B, 1KB,) and the number of threads(1,2,4,8).

For Disk benchmark, we measured the throughputs ( sequential read, sequential write, random read and random write) and latency(random read, random write). We varied the block size(1B/1KB/1MB) and the number of threads(1,2,4,8).

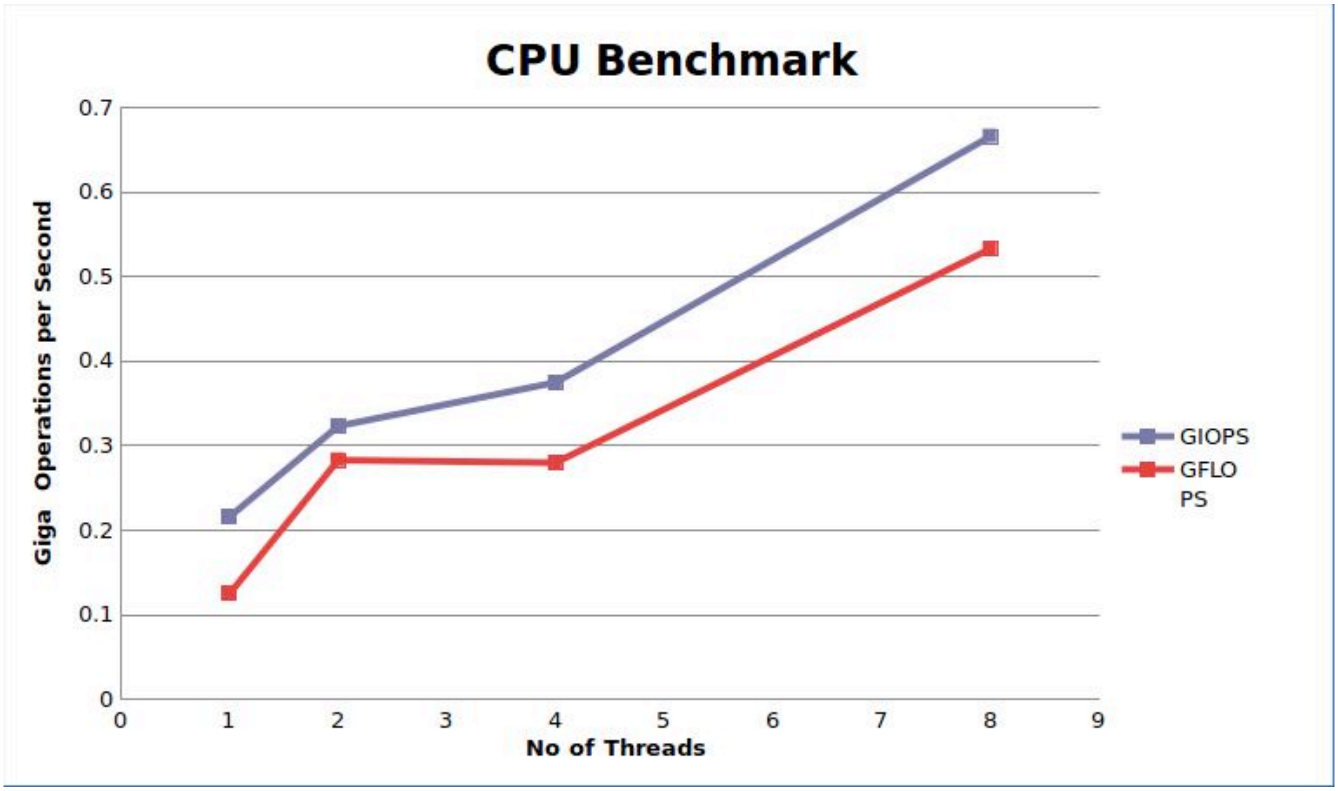
For Network benchmark, we measured throughput and latency for both TCP and UDP protocols. We varied the packet size(1b/1Kb/65Kb) and the number of threads(1,2,4,8).

## 3. Experiment Results and Analysis

In this section, we present the experiment result for each benchmark and give explanations for the trends in the results.

### 3.1 CPU benchmark results

- For CPU benchmark we are finding GIOPS and GFLOPS value for different number of threads.
- In graph we had done experiments for 1,2,4 and 8 threads and according to that we are getting result between 0.1 GFLOPS and GIOPS to 0.7 GFLOPS and GIOPS.



- c.No. of optimal threads are “8” to get the best performance.
- d.Theoretical peak performance of the processor is given by:

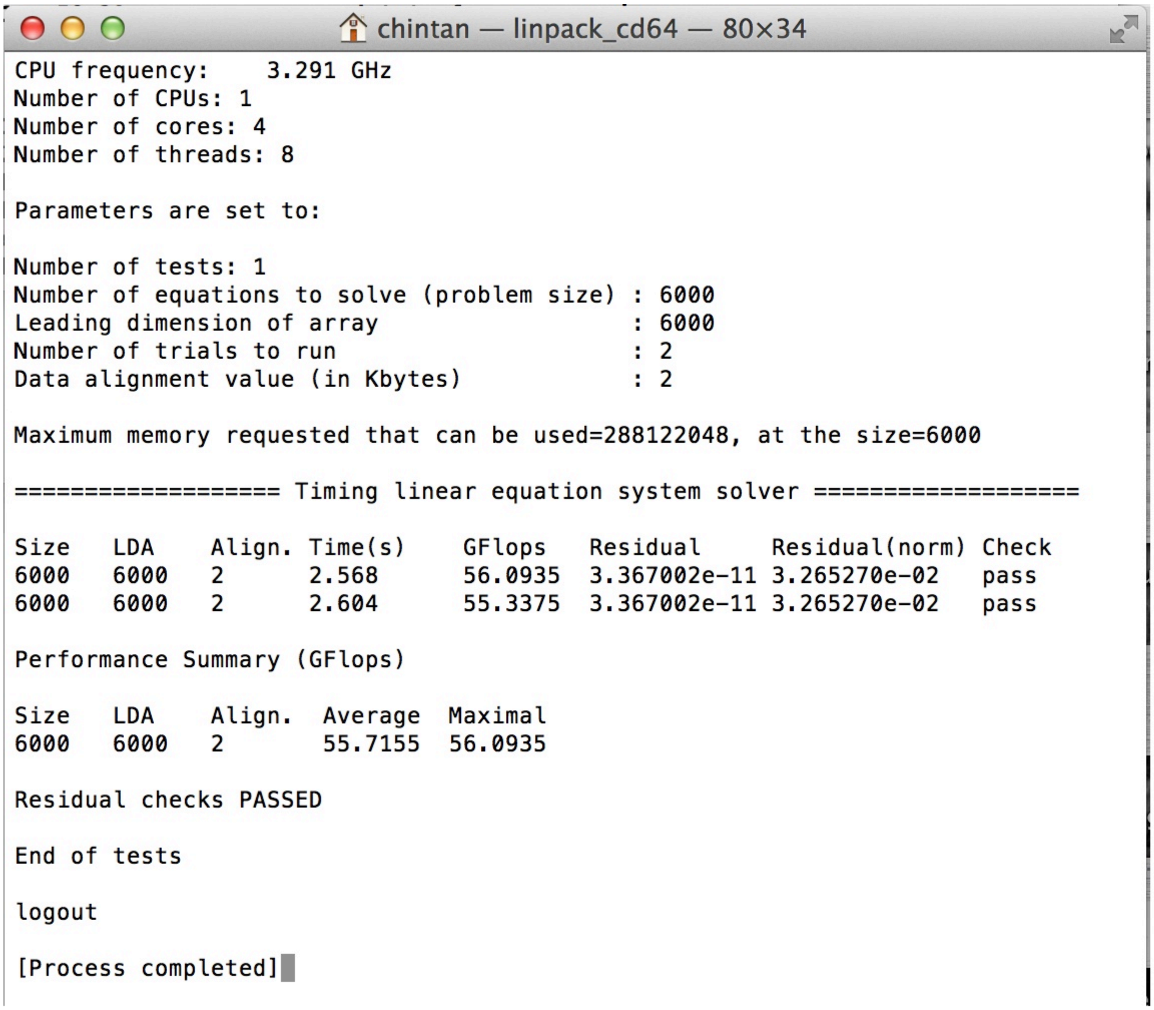
  cores*clocks*flops/cycle<br>
  =4*3.281*4<br>
  =58.56Gflops/sec

- e. We achieve 1% efficiency as compared to theoritical performance.
- f. Average and standard deviation for IOPS and FLOPS are shown in below table:

| Number of thread | Average(GIOPS) | Standard Deviation(GIOPS) | Average(GFLOPS) | Standard Deviation(GFLOPS) |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 1 | 0.21642 | 0.05172 | 0.12567 | 0.01153 |
| 2 | 0.3233 | 0.1537 | 0.2832 | 0.05336 |
| 4 | 0.1537 | 0.19077 | 0.43637 | 0.1465 |
| 8 | 0.56033 | 0.11138 | 0.58861 | 0.05344 |

- g. We had run the linpack benchmark and result is being shown in below screen shot.



```
CPU frequency:    3.291 GHz
Number of CPUs: 1
Number of cores: 4
Number of threads: 8

Parameters are set to:

Number of tests: 1
Number of equations to solve (problem size) : 6000
Leading dimension of array                  : 6000
Number of trials to run                     : 2
Data alignment value (in Kbytes)            : 2

Maximum memory requested that can be used=288122048, at the size=6000

================== Timing linear equation system solver ===================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
6000   6000   2      2.568      56.0935  3.367002e-11 3.265270e-02   pass
6000   6000   2      2.604      55.3375  3.367002e-11 3.265270e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
6000   6000   2       55.7155  56.0935

Residual checks PASSED

End of tests

logout

[Process completed]
```

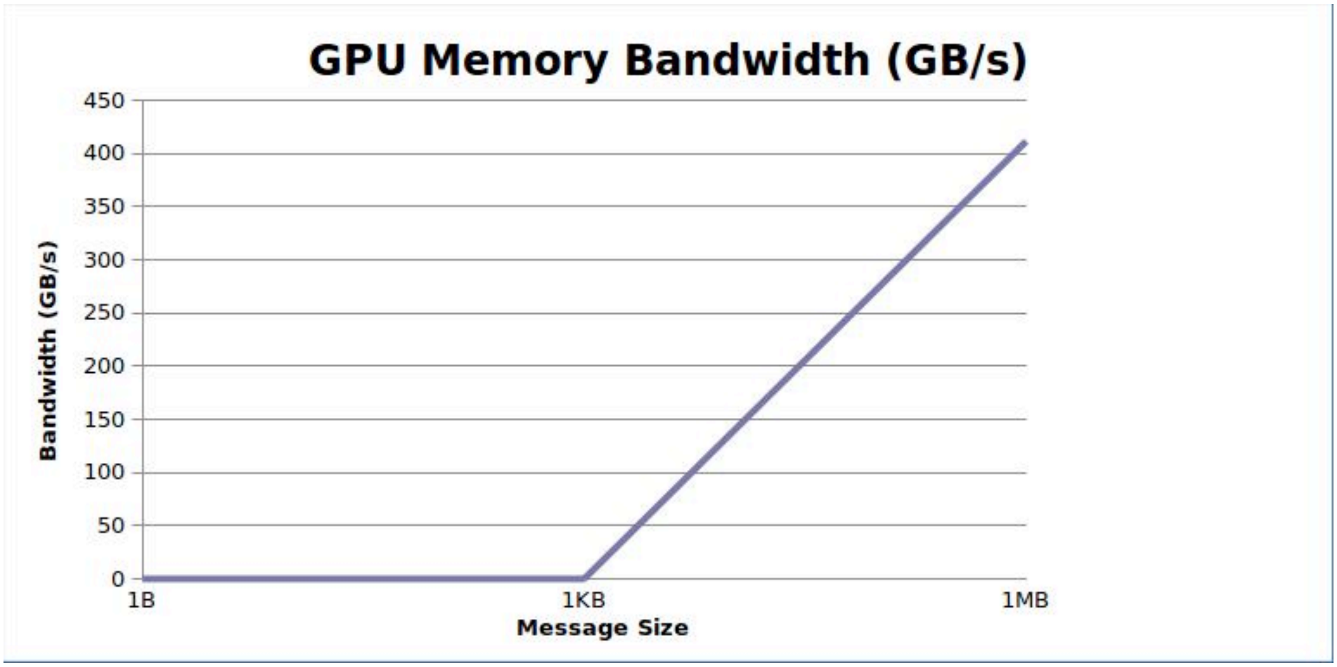
From practical performance we got 56.7155 Gflops.

- Efficiency = (55.7155/58.56)*100 = 95.14 %

### 3.2 GPU benchmark results

- c. GPU theoritical speed is 564 GB/s
- while in practical we got 22.6133 Gb/s
- So efficiency is almost 4%.

- d. For GPU benchmark memory bandwidth, we had created that shows that we have 3 block sizes. 1B, 1KB and 1MB.
- From graph we can say that when message size is between 1 byte and 1 kilobyte then bandwidth remains same. While between 1 kb and 1 mb it increases upto 400GB/sec.



- **e.** theoritical bandwidth = 64GB/sec

practical bandwidth =0.35413 GB/sec

- so efficiency is 0.55 %
- g. Average and Standard deviation
- Average GIOPS = 22.6133
- Standard deviation = 0.8434
- Average GIOPS = 0.00519
- Standard deviation = 0

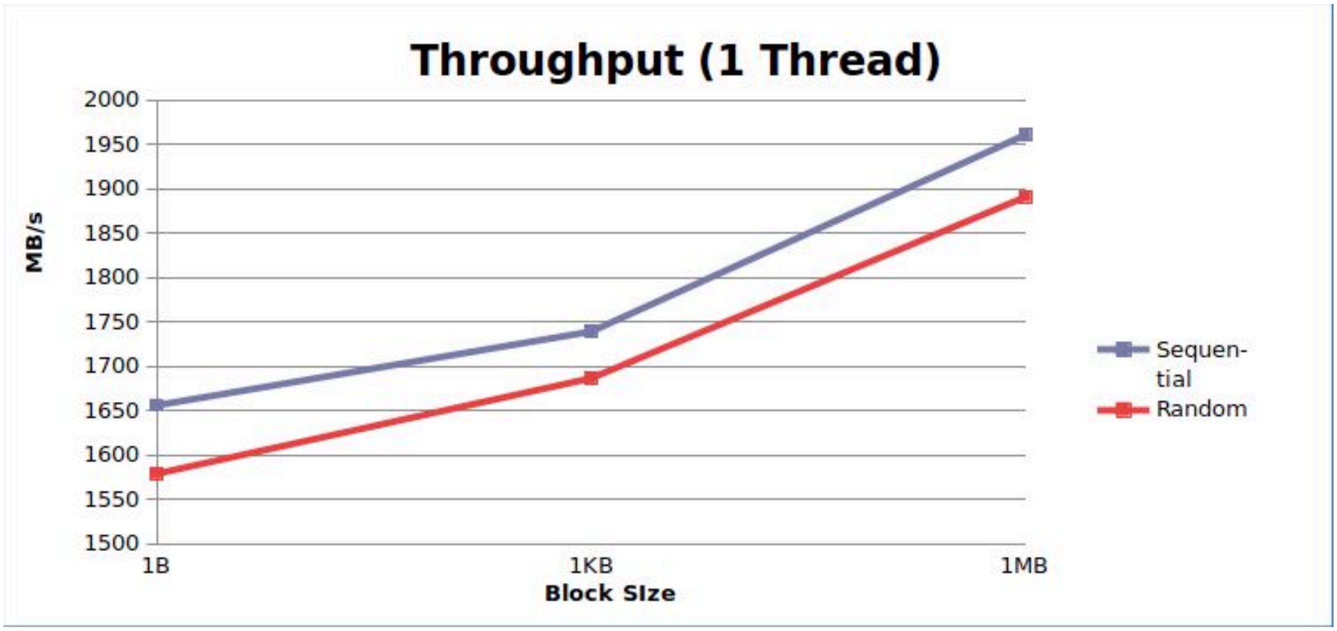
| **Number of blocks** | **Average** | **Standard Deviation** |
|---|---|---|
| 1 B | 0.00041 | 0.00002 |
| 1 KB | 0.35413 | 0.1185 |
| 1 MB | 0.1185 | 1.405 |

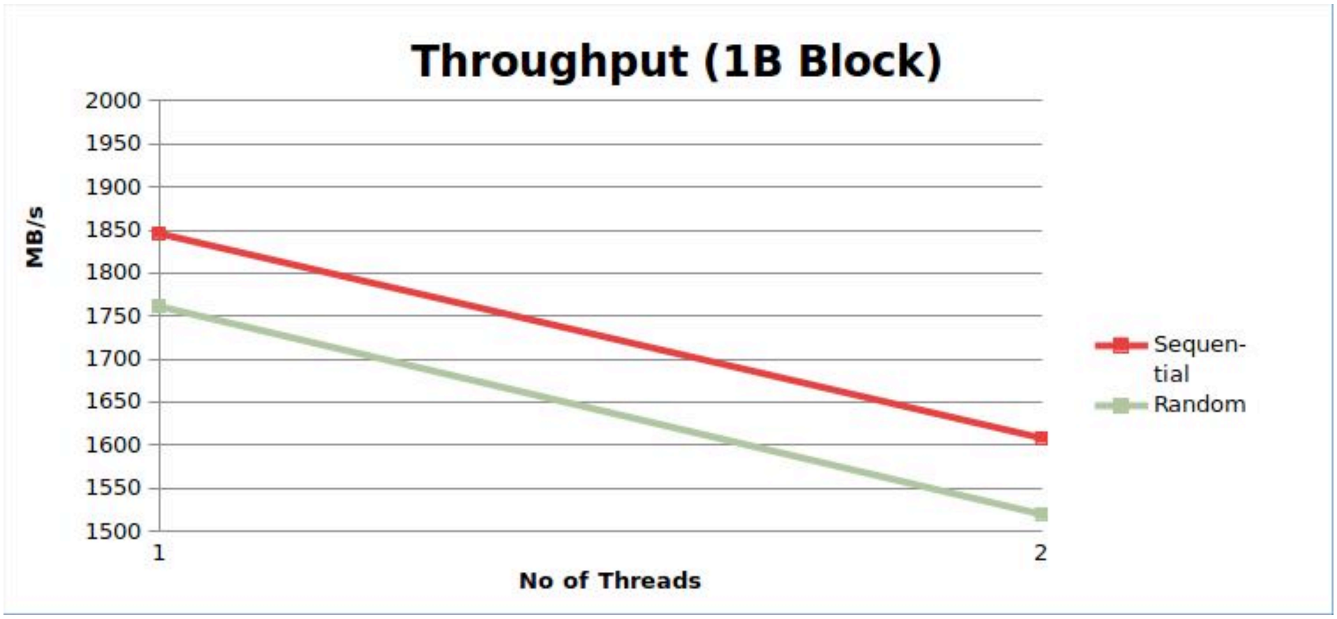
### 3.3 Memory benchmark results

### 3.3.1 Thread number effect on throughput

- For single thread we had calculated sequential and random speed.

- The graph shows the implementation and comparison between sequential and random throughput. It's going upward as we increase the memory size.

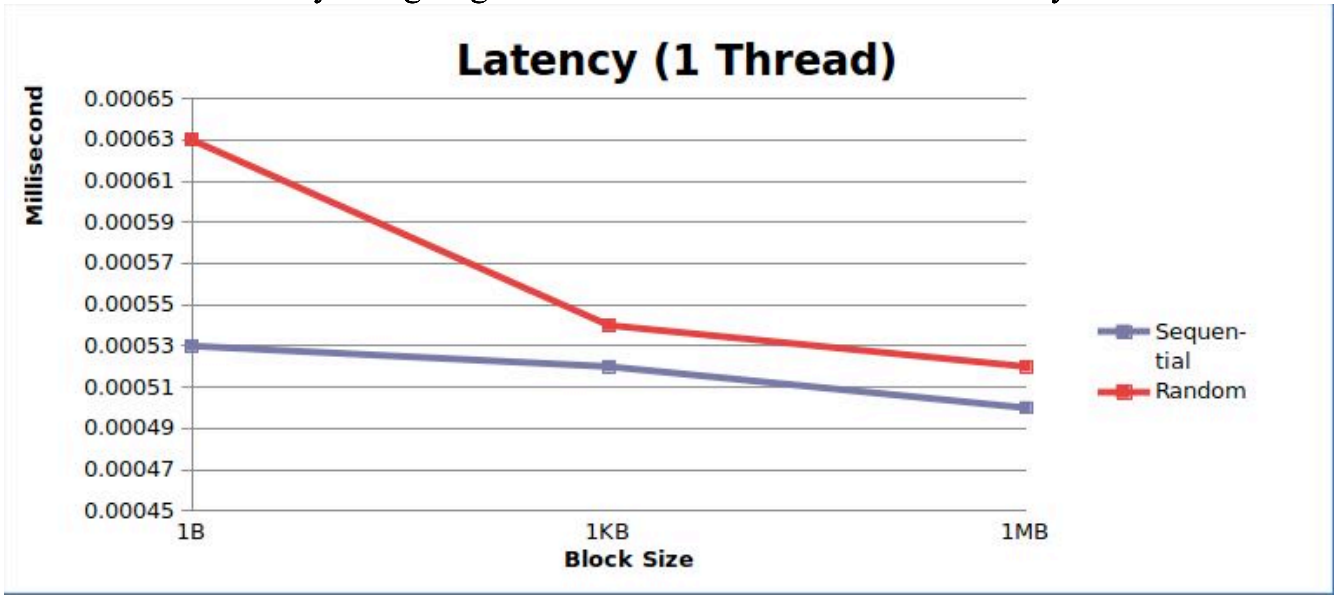




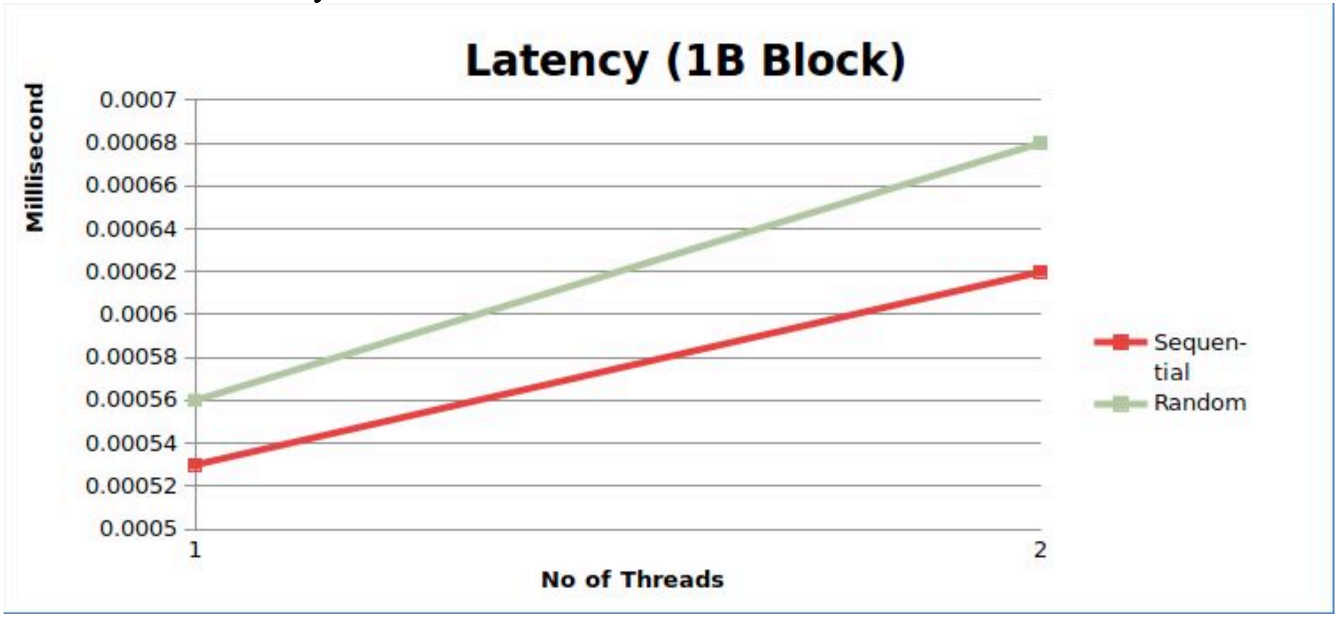
- The graph shows the implementation and comparison between sequential and random throughput.
- It gives the the result that, when number of threads are being increased, throughput of 1 byte block will decrease.

### 3.3.3 Thread number effect on latency

- Here we had calculated the latency of single thread and for block size for sequential and random time.
- The graph dictects the implementation and comparison between sequential and random latency. It's going downward as we increase the memory size.



- The graph shows the implementation and comparison between sequential and random latency.



- It gives the the result that, when number of threads are being increased, throughput of 1 byte block will increase.
- c. theoretical throughput = 12800 Mb/sec , Latency = 0.625 ms .
- d. The optimal number of thread is 1, for 1 byte of block size.
- e. practical throughput = 1845.91 Mb/sec and latency= 0.00053 ms.

- So efficiency is almost 14%.
- f. Average and Standard deviation

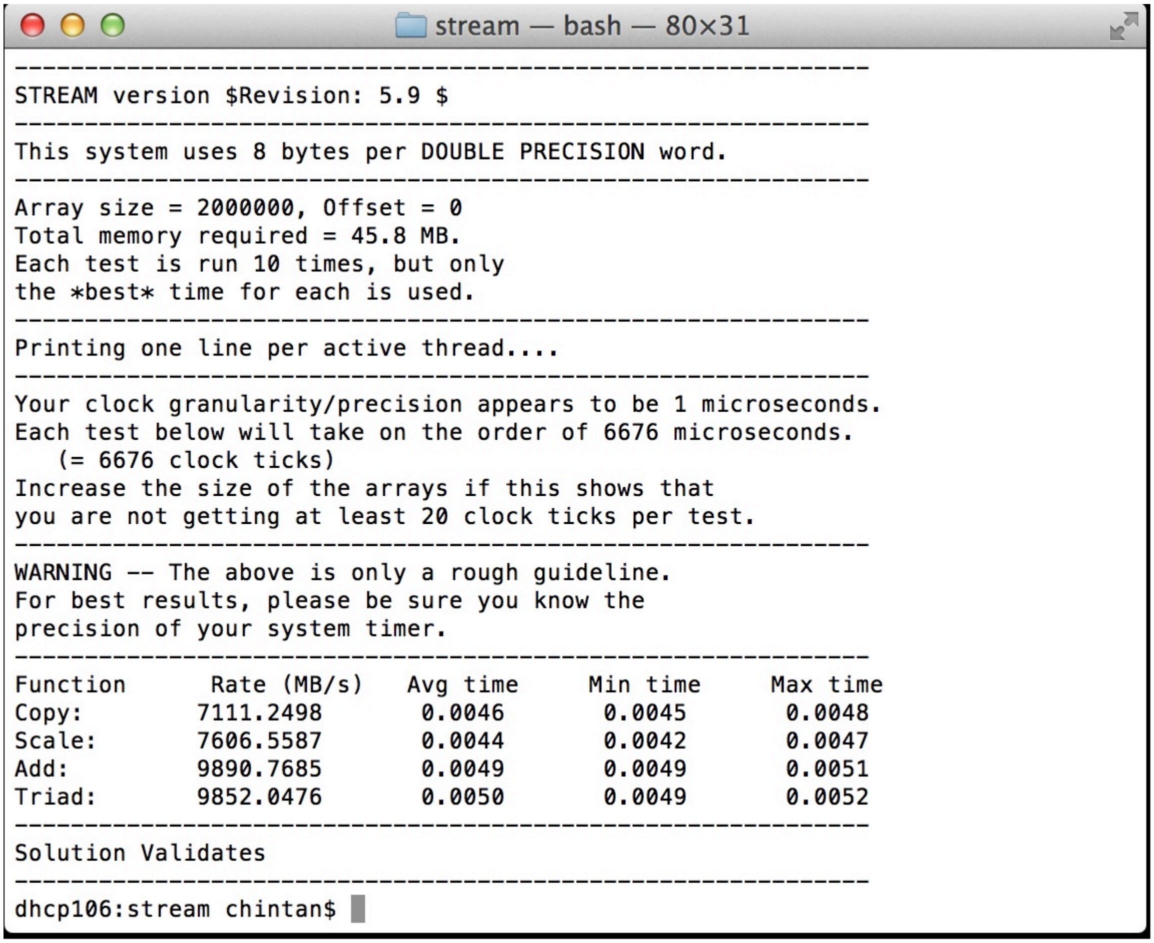
For Sequential:

| **Number of Thread** | **Number of blocks** | **Average(Throughput)** | **Standard Deviation(Throuput)** | **Average(Latency)** | **Standard Deviation(Latency)** |
|---|---|---|---|---|---|
| 1 | 1 B | 1845.91 | 13.25118 | 0.0005 | 0 |
| 1 | 1 KB | 1786.93 | 67.78326 | 0.00056 | 0.00002 |
| 1 | 1 MB | 1880.41 | 90.15204 | 0.00053 | 0.00003 |
| 2 | 1 B | 1608 | 41.32796 | 0.00062 | 0.00002 |
| 2 | 1 KB | 1600.33 | 48.58326 | 0.00062 | 0.00002 |
| 2 | 1 MB | 1618.33 | 38.8873 | 0.00062 | 0.00001 |

For Random:

| **Number of Thread** | **Number of blocks** | **Average(Throughput)** | **Standard Deviation(Throuput)** | **Average(Latency)** | **Standard Deviation(Latency)** |
|---|---|---|---|---|---|
| 1 | 1 B | 1781.66 | 186.14 | 0.00056 | 0.00006 |
| 1 | 1 KB | 1833 | 207.88 | 0.00055 | 0.00006 |
| 1 | 1 MB | 1783.33 | 90.16 | 0.00056 | 0.00003 |
| 2 | 1 B | 1533.66 | 90.89 | 0.00062 | 0.00007 |
| 2 | 1 KB | 1597.66 | 114.15 | 0.00063 | 0.00005 |
| 2 | 1 MB | 1618.33 | 63.55 | 0.00064 | 0.000013 |

- g. We had run the Stream benchmark and result is being shown in below screen shot.
- From thoretical performance we got 12800 MB/sec throughput and 0.625ms latency.
- While practical performance is 8025.45 MB/sec throughput and 0.0046ms latency.



```
-------------------------------------------------------------
STREAM version $Revision: 5.9 $
-------------------------------------------------------------
This system uses 8 bytes per DOUBLE PRECISION word.
-------------------------------------------------------------
Array size = 2000000, Offset = 0
Total memory required = 45.8 MB.
Each test is run 10 times, but only
the *best* time for each is used.
-------------------------------------------------------------
Printing one line per active thread....
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 6676 microseconds.
   (= 6676 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function      Rate (MB/s)   Avg time     Min time     Max time
Copy:        7111.2498       0.0046       0.0045       0.0048
Scale:       7606.5587       0.0044       0.0042       0.0047
Add:         9890.7685       0.0049       0.0049       0.0051
Triad:       9852.0476       0.0050       0.0049       0.0052
-------------------------------------------------------------
Solution Validates
-------------------------------------------------------------
dhcp106:stream chintan$ 
```

- Efficiency for throughput =62.69%
- Efficiency for latency =0.7%

### 3.4 Network benchmark results

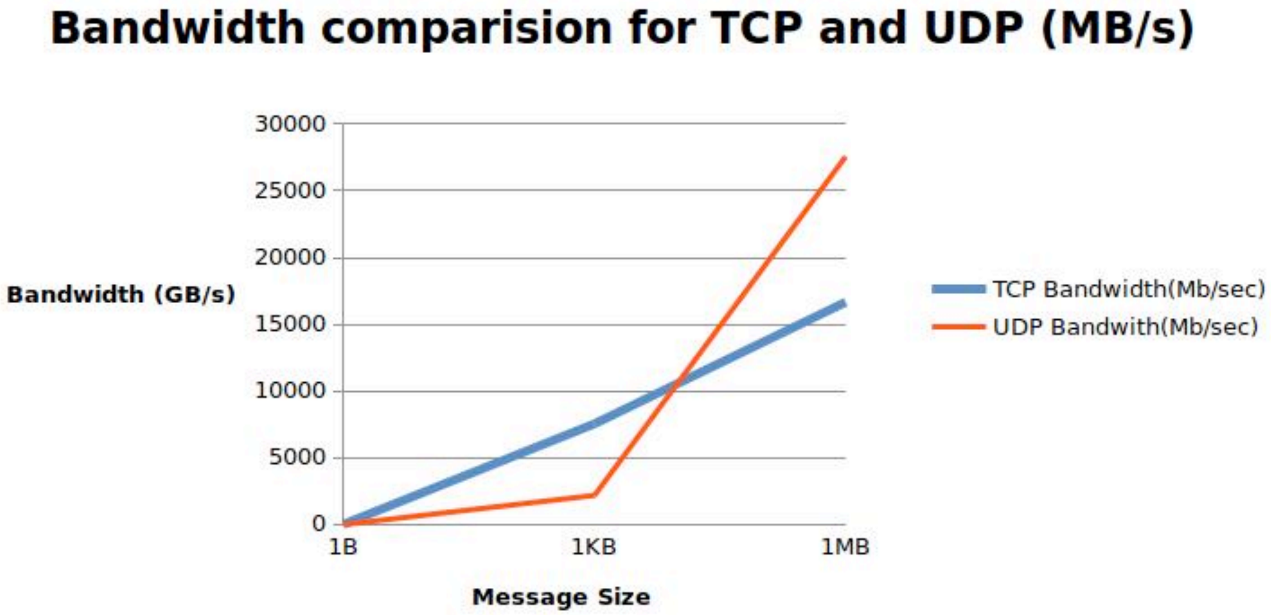
- c. throughput = 2.12 Mb/sec , Latency = 4.716E-4 ms .
- d. Average and Standard deviation

For throught:

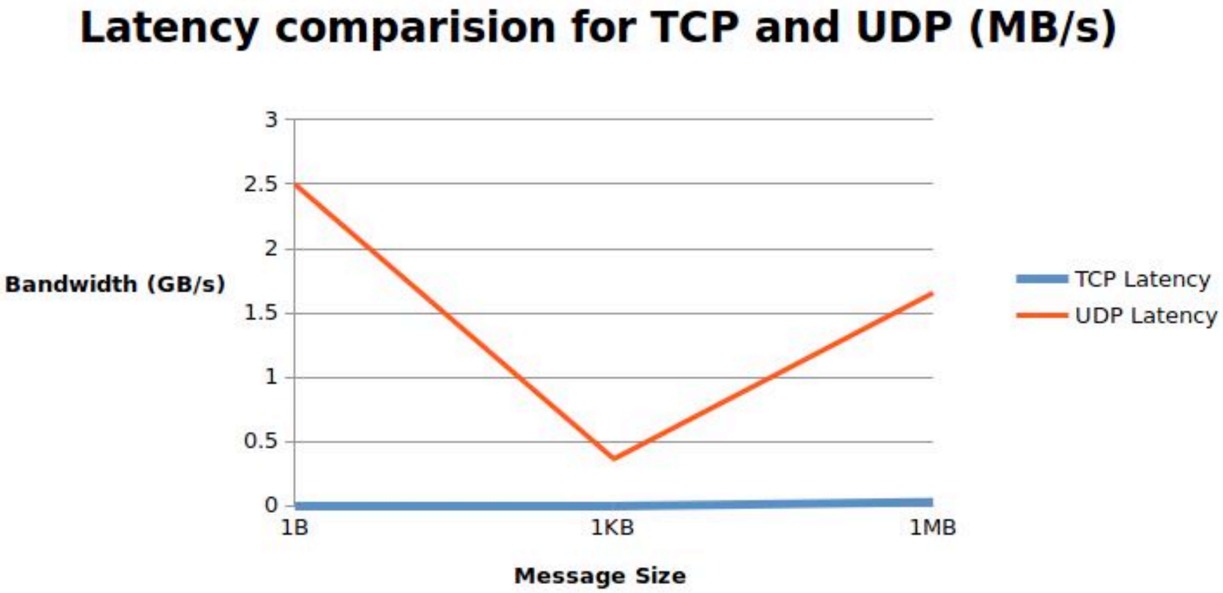
| **Number of blocks** | **Average( Through put- TCP)** | **Standard Deviation(Throughput-TCP)** | **Average(Throughput-UDP)** | **Standard Deviation(Throughput-UDP)** |
|---|---|---|---|---|
| 1 B | 15.63 | 3.28 | 2.18 | 3.425 |
| 1 KB | 7547.17 | 67.76 | 2185.79 | 70.25 |
| 1 MB | 16666.66 | 85.14 | 27566.20 | 90.584 |

For Latency:

| Number of blocks | Average(Latency-TCP) | Standard Deviation(Latency-TCP) | Average(Latency-UDP) | Standard Deviation(Latency-UDP) |
|---|---|---|---|---|
| 1 B | 0.00047 | 0.00005 | 2 | 0.000025 |
| 1 KB | 0.00106 | 0.00002 | 0.366 | 0.0025 |
| 1 MB | 0.03072 | 0.0001 | 1.656 | 0.056 |

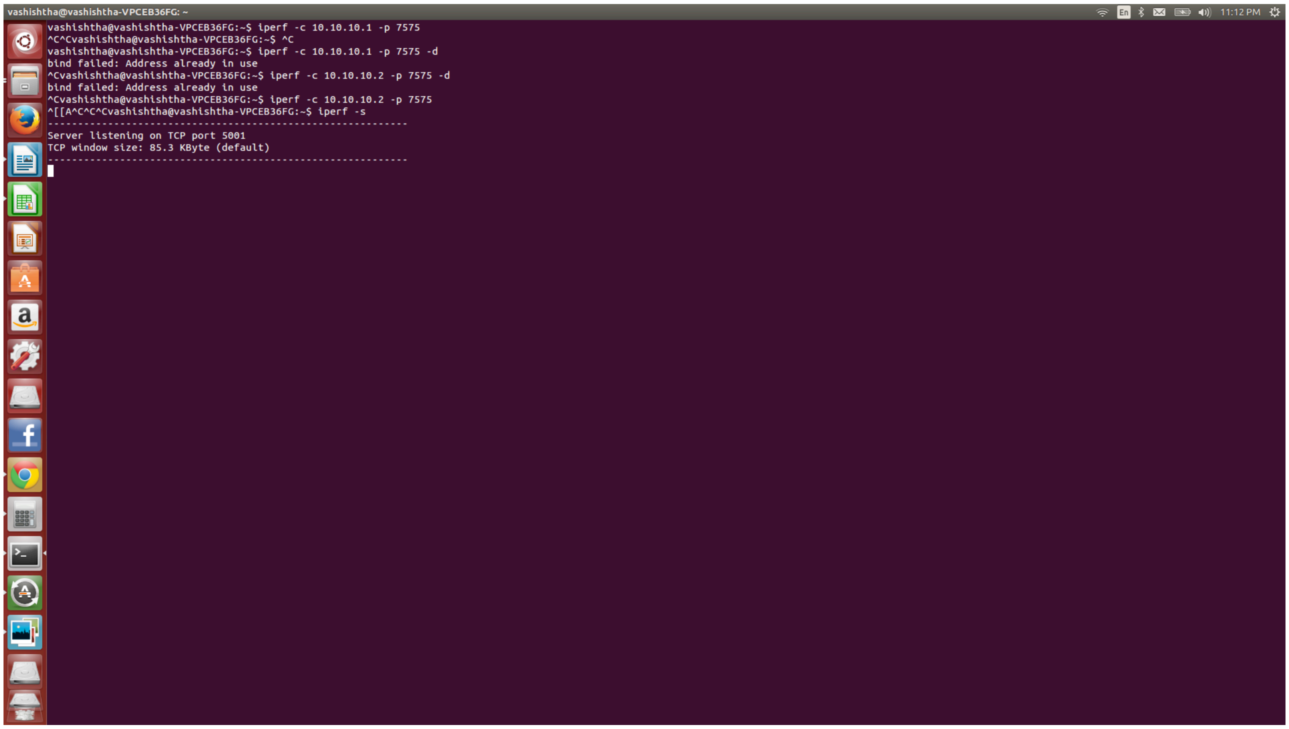


- The graph shows the implementation and comparison between TCP and UDP bandwidths. The result is being displayed as above:



The graph shows the implementation and comparison between TCP and UDP latencies. The result is being displayed as above.

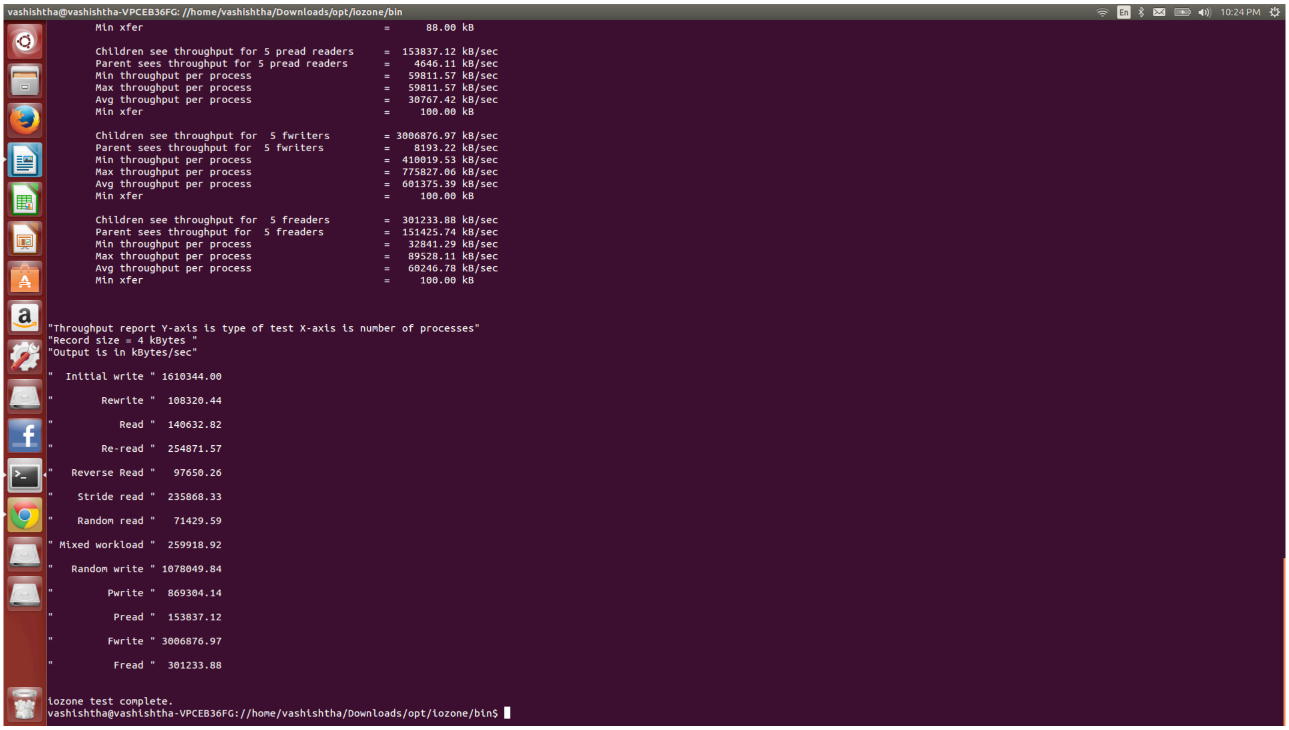
- f. We had run the IPerf benchmark and result is being shown in below screen shot.



```
vashishtha@vashishtha-VPCEB36FG:~$ iperf -c 10.10.10.1 -p 7575
^C^Cvashishtha@vashishtha-VPCEB36FG:~$ ^C
vashishtha@vashishtha-VPCEB36FG:~$ iperf -c 10.10.10.1 -p 7575 -d
bind failed: Address already in use
^Cvashishtha@vashishtha-VPCEB36FG:~$ iperf -c 10.10.10.2 -p 7575 -d
bind failed: Address already in use
^Cvashishtha@vashishtha-VPCEB36FG:~$ iperf -c 10.10.10.2 -p 7575
^[[A^C^C^Cvashishtha@vashishtha-VPCEB36FG:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

### 3.5 Disk benchmark

- f. We had run the IOzone benchmark and result is being shown in below screen shot.


vashishtha@vashishtha-VPCEB36FG: //home/vashishtha/Downloads/opt/iozone/bin
10:24 PM

```
        Min xfer                                        =      88.00 kB

        Children see throughput for 5 pread readers     =  153837.12 kB/sec
        Parent sees throughput for 5 pread readers      =    4646.11 kB/sec
        Min throughput per process                      =   59811.57 kB/sec
        Max throughput per process                      =   59811.57 kB/sec
        Avg throughput per process                      =   30767.42 kB/sec
        Min xfer                                        =     100.00 kB

        Children see throughput for  5 fwriters         = 3006876.97 kB/sec
        Parent sees throughput for  5 fwriters          =    8193.22 kB/sec
        Min throughput per process                      =  410019.53 kB/sec
        Max throughput per process                      =  775827.06 kB/sec
        Avg throughput per process                      =  601375.39 kB/sec
        Min xfer                                        =     100.00 kB

        Children see throughput for  5 freaders         =  301233.88 kB/sec
        Parent sees throughput for  5 freaders          =  151425.74 kB/sec
        Min throughput per process                      =   32841.29 kB/sec
        Max throughput per process                      =   89528.11 kB/sec
        Avg throughput per process                      =   60246.78 kB/sec
        Min xfer                                        =     100.00 kB



"Throughput report Y-axis is type of test X-axis is number of processes"
"Record size = 4 kBytes "
"Output is in kBytes/sec"

"  Initial write " 1610344.00

"        Rewrite "  108320.44

"           Read "  140632.82

"        Re-read "  254871.57

"   Reverse Read "   97650.26

"    Stride read "  235868.33

"    Random read "   71429.59

" Mixed workload "  259918.92

"   Random write " 1078049.84

"         Pwrite "  869304.14

"          Pread "  153837.12

"         Fwrite " 3006876.97

"          Fread "  301233.88


iozone test complete.
vashishtha@vashishtha-VPCEB36FG://home/vashishtha/Downloads/opt/iozone/bin$ 
```

From practical Throughput= 5.7155 Gb/sec = 5715.54 Mb/sec.

- And latency = 0.547 ms.
- Efficiency = 95.14 %
- Here we had calculated the throughput of single thread and for block size for sequential and random read/write.
- The graph dictects the implementation and comparison between sequential and random read/write throughput according to block size. It's going downward as we increase the memory size.
- Sequential throughput will give more speed than random throughput for both read and write.
- Write throughput will give less speed than read throughput. Hence, we can say that sequential throughput is more efficient.
- Another Graph shows the implementation and comparison of number of threads in sequential read, sequential write, random read and random write with respect to latency in milliseconds.
- Sequential latency will give more speed than random throughput for both read and write.
- Write throughput will give less speed than read throughput. Hence, we can say that sequential throughput is more efficient.
- e. Standard Deviation and Average

For Sequential Read:

| **Number of Thread** | **Average(Throughput-Sequential Read)** | **Standard Deviation(Throughput-Sequential Read)** | **Average(Latency-Sequential Read)** | **Standard Deviation((Latency-Sequential Read)** |
|---|---|---|---|---|
| 1 | 25.45 | 18.14 | 0.004 | 0.00006 |
| 2 | 20.23 | 27.88 | 0.0042 | 0.00007 |
| 4 | 19.12 | 75.55 | 0.0058 | 0.0023 |

For Random read:

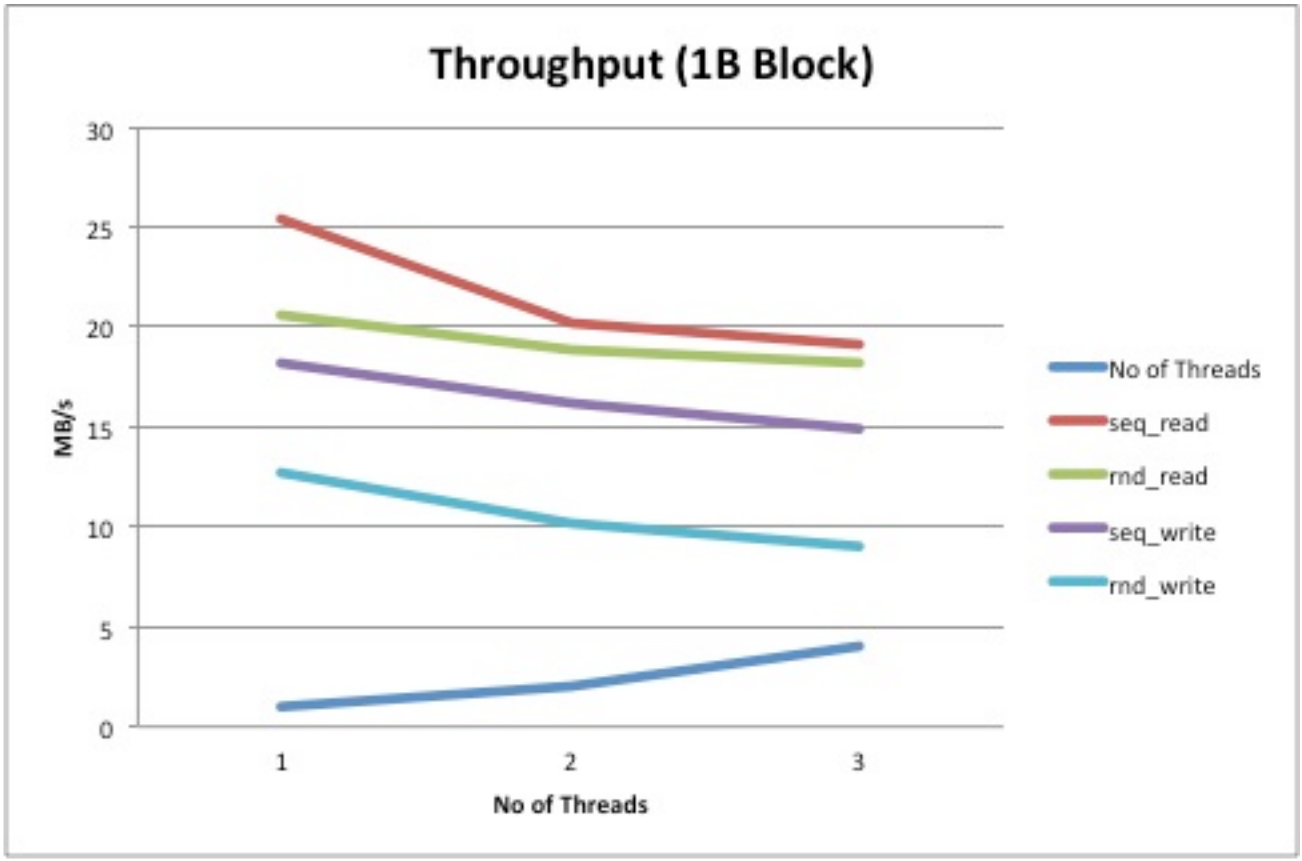
| **Number of Thread** | **Average(Throughput-Random Read)** | **Standard Deviation(Throughput-Random Read)** | **Average(Latency-Random Read)** | **Standard Deviation((Latency-Random Read)** |
|---|---|---|---|---|
| 1 | 20.56 | 18.14 | 0.00063 | 0.00006 |
| 2 | 18.89 | 26.650 | 0.00071 | 0.00004 |
| 4 | 18.2 | 7.36 | 0.0009 | 0.00002 |

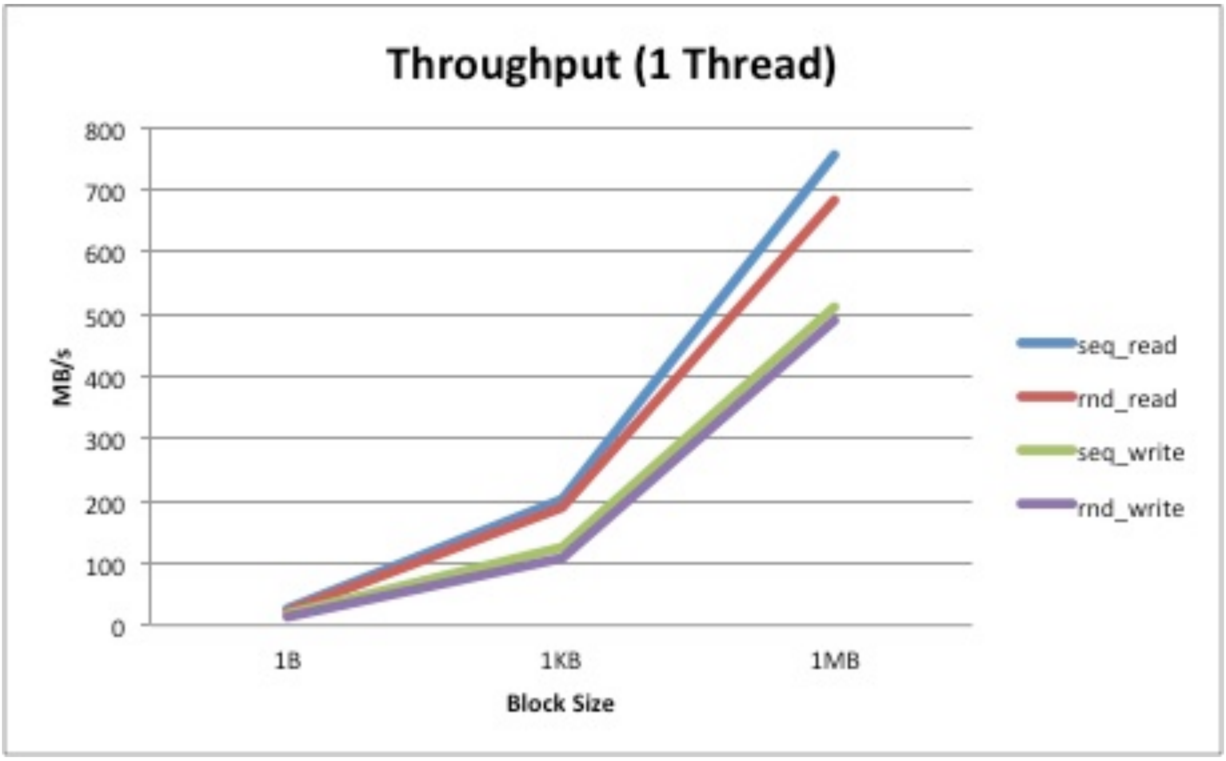
For Sequential write:

| **Number of Thread** | **Average(Throughput-Sequential Write)** | **Standard Deviation(Throughput-Sequential Write)** | **Average(Latency-Sequential write)** | **Standard Deviation((Latency-Sequential Write)** |
|---|---|---|---|---|
| 1 | 18.2 | 16.14 | 0.0081 | 0.007 |
| 2 | 16.12 | 14.75 | 0.0096 | 0.006 |
| 4 | 14.9 | 25.65 | 0.012 | 0.008 |

For Random write :

| **Number of Thread** | **Average(Throughput-Random write)** | **Standard Deviation(Throughput-Random write)** | **Average(Latency-Random write)** | **Standard Deviation((Latency-Random write)** |
|---|---|---|---|---|
| 1 | 12.67 | 17.12 | 0.0101 | 0.006 |
| 2 | 10.11 | 13.65 | 0.0123 | 0.008 |
| 4 | 9.02 | 8.30 | 0.0156 | 0.025 |


Throughput (1B Block)
30
25
20
15
10
5
0
MB/s
1
2
3
No of Threads
No of Threads
seq_read
rnd_read
seq_write
rnd_write


Throughput (1 Thread)
800
700
600
500
400
300
200
100
0
MB/s
1B
1KB
1MB
Block Size
seq_read
rnd_read
seq_write
rnd_write