Performance Evaluation AWS-EC2 Cloud Benchmark

This document contain the performance evaluation of CPU Benchmark, Disk Benchmark and Memory Benchmark on AWS EC-2 Cloud. My experiments cover all the original specification for all benchmarks of AWS-EC2 cloud. There are so many experiments done for all benchmarks and every experiments have done 3-times for calculating average and standard deviation of all figures.

First of all, I describe the Environment of AWS-EC2 Cloud which is shown in below figure. Then the specification of each benchmark is presented followed by experiment analysis and result in form of table and graph.

1. AWS-EC2 Environment

write this command on AWS-EC2 instance after launching. It gives you following output of all specification

[ec2-user@ip-172-31-49-170 ~]$ **lscpu**

```
[ec2-user@ip-172-31-49-170 ~]$ clear
[ec2-user@ip-172-31-49-170 ~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                1
On-line CPU(s) list:   0
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             1
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Model name:            Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz
Stepping:              2
CPU MHz:               2400.046
BogoMIPS:              4800.09
Hypervisor vendor:     Xen
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
NUMA node0 CPU(s):     0
[ec2-user@ip-172-31-49-170 ~]$
```

2. Benchmark Experiment and Result Analysis

I represent all experimental result analysis of three benchmark of CPU, Disk and Memory and explanations of each and every outcomes with comparison of theoretical performance.

2.1 CPU Benchmark

a. For CPU Benchmark, calculating results in terms of FLOPS(FLoating point Operations per Second) and IOPS (Interger Operations per Second) and then find GFLOPS and GIOPS . (G – Giga 10^9).

b. Results of FLOPS and IOPS according to no. of threads(1, 2 and 4 threads) shown in figure Fig 1. below in form of graph.

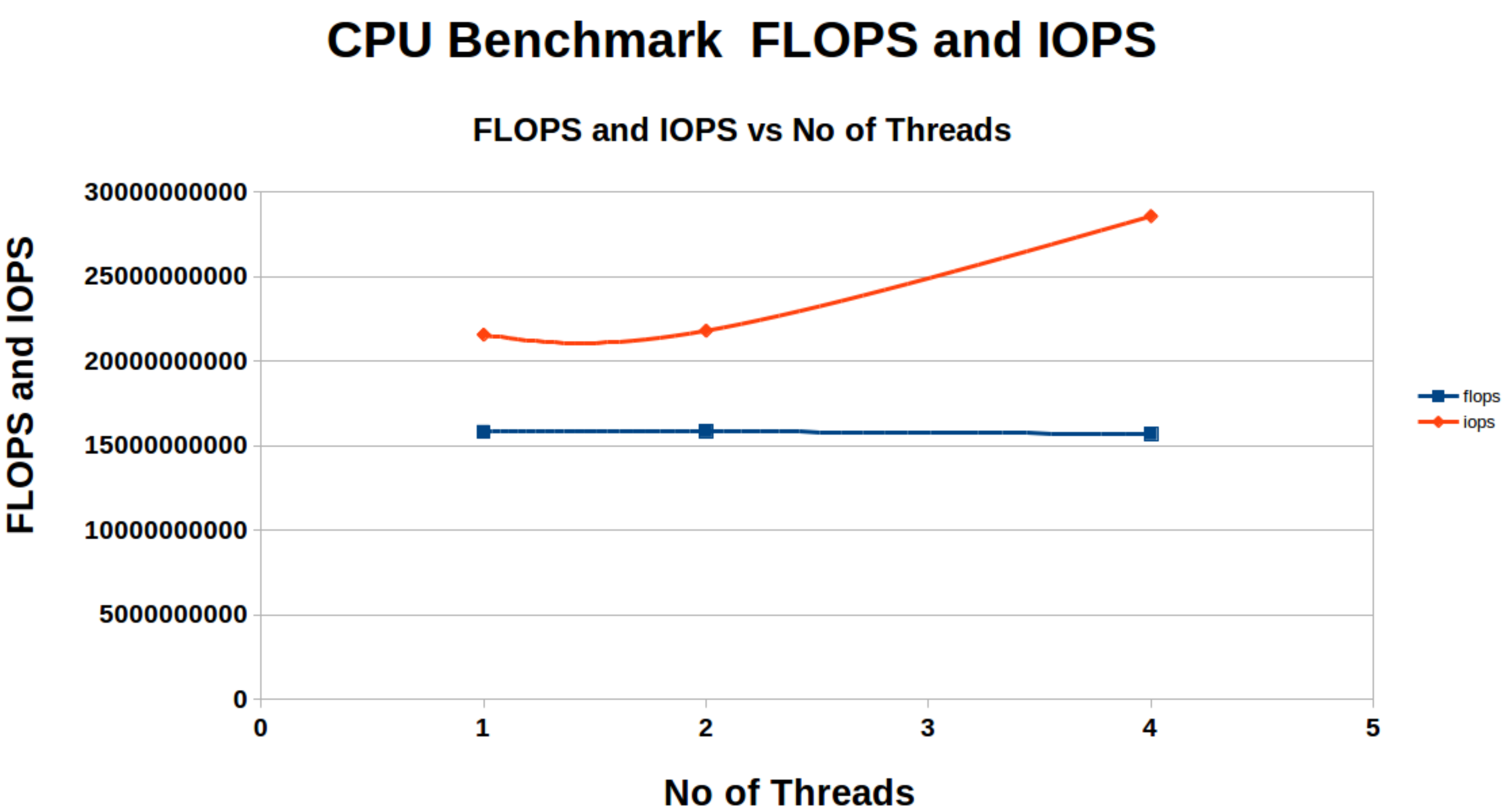


Fig 1.

c. According to practical result no. of optimal threads is “4” for best performance

No. of Threads	Average(GFLOPS)	SD(GFLOPS)	Average(GIOPS)	SD(GIOPS)
1	15.794472300	0.00934	21.573456800	0.17122
2	15.810272300	0.01747	21.642424300	0.11734
4	15.871454500	0.00794	28.533168500	0.12344

d. Practicle performance

FLOPS = **15810272300** and GFLOPS = **15.81**
IOPS = **21563328500** and GIOPS = **21.63**

e. Theoritical peak performance of AWS-EC2 cloud is given below:

performance = (CPU Speed in GHz) * (CPU IPC)* (no.of cores in CPU) * (number of CPU per node)

performance = 2.4*16*1*1 = 38.4 GFLOPS

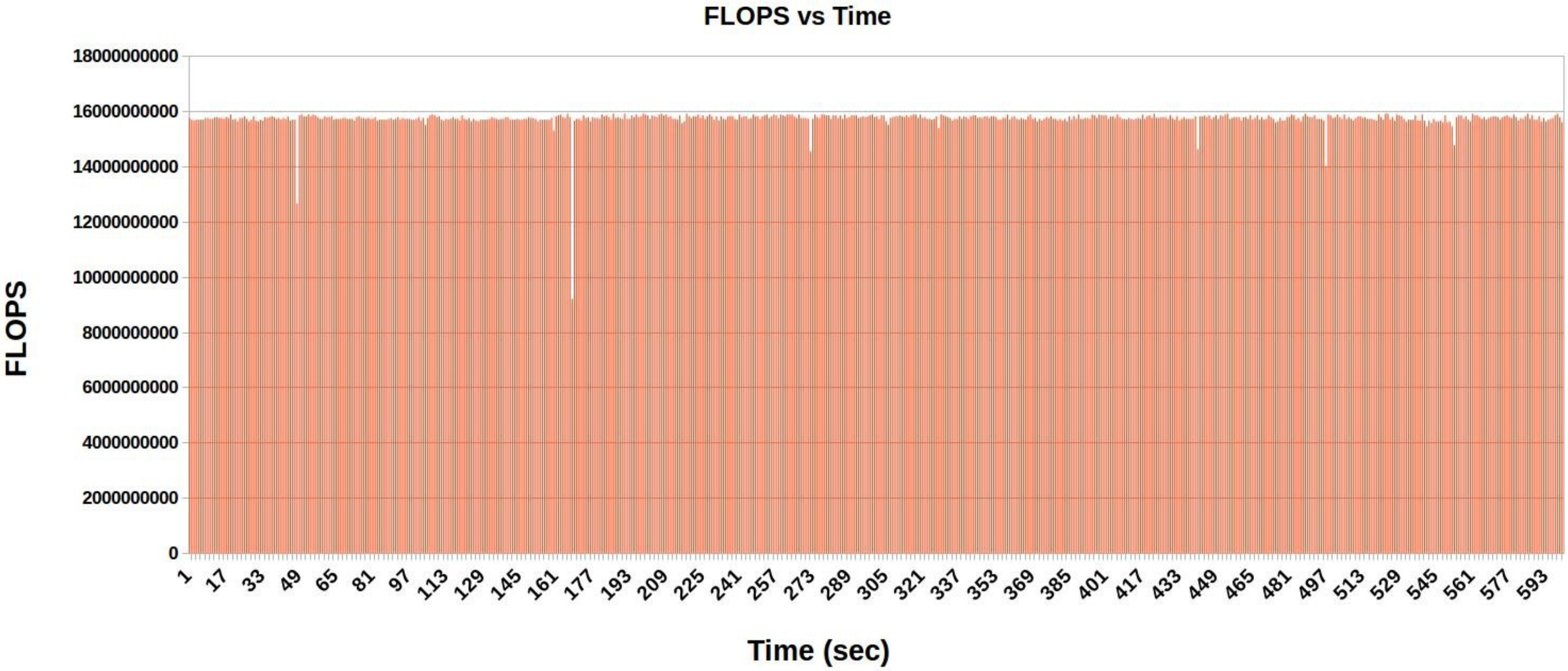
f. Efficiency of CPU Speed (GFLOPS) (**49.65 %**)

Efficiency = (15.81)/(38.4)
= **49.65 %**

g. I run this program for 10 min and took 600 samples of 1 second each. I took samples of FLOPS and IOPS. It gives me a better idea of CPU performance while performing this much of exsperiments.

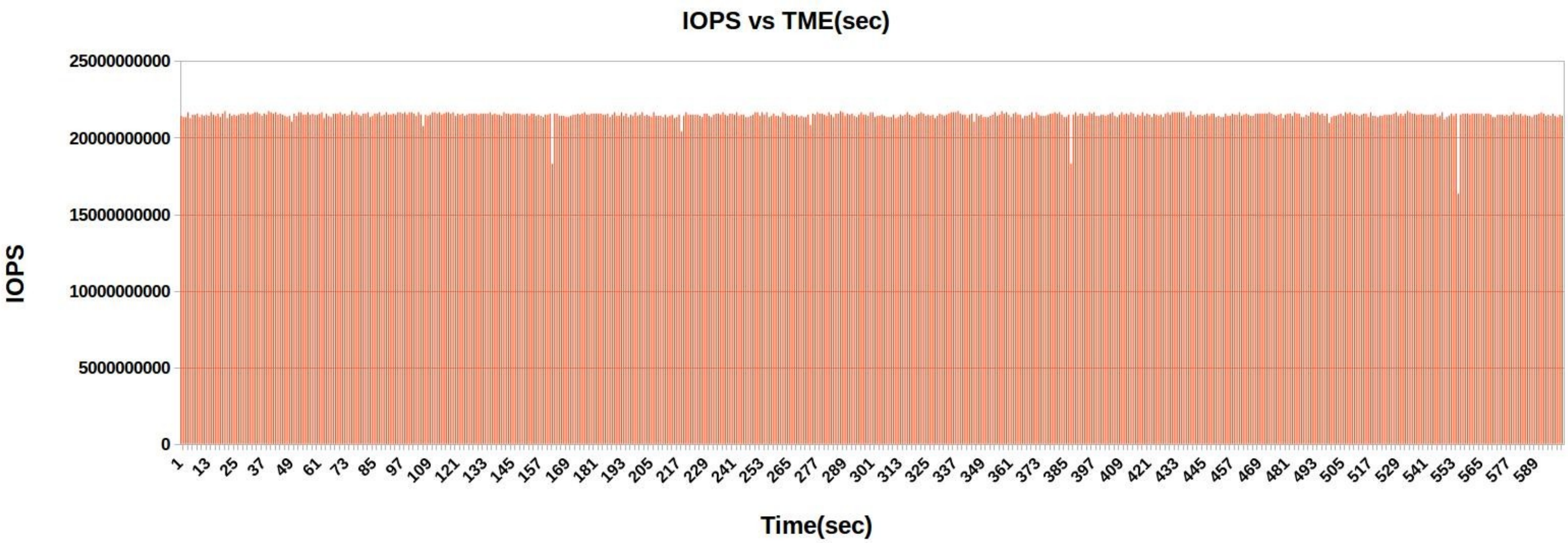
Given below graph gives better idea of FLOPS and IOPS

Graph of FLOPS vs Time for 600 Samples



h. 10 mins samples of IOPS and measured CPU Performance.

IOPS - 600 Samples (each per 1 second)



I. Practical Performance by running “linpack” on the AWS-EC2 Cloud. Below is the figure to see the performance of CPU benchmark.

Efficiency of **linpack** result to theoretical result = (33.69/38.4)
= **87.73%**

Efficiency = 87.73%.

Figure is for “linpack” CPU Benchmark.

Fig 3.

```
[ec2-user@ip-172-31-49-170 linpack]$ ./runme_xeon64
This is a SAMPLE run script for SMP LINPACK. Change it to reflect
the correct number of CPUs/threads, problem input files, etc..
Fri Feb 12 00:19:36 UTC 2016
Intel(R) Optimized LINPACK Benchmark data

Current date/time: Fri Feb 12 00:19:36 2016

CPU frequency:    2.992 GHz
Number of CPUs: 1
Number of cores: 1
Number of threads: 1

Parameters are set to:

Number of tests: 15
Number of equations to solve (problem size) : 1000  2000  5000  10000 15000 18000 20000 22000 25000 26000 27000 30000 35000 40000 45000
Leading dimension of array                  : 1000  2000  5008  10000 15000 18008 20016 22008 25000 26000 27000 30000 35000 40000 45000
Number of trials to run                     : 4     2     2     2     2     2     2     2     2     2     1     1     1     1     1
Data alignment value (in Kbytes)            : 4     4     4     4     4     4     4     4     4     4     4     1     1     1     1

Maximum memory requested that can be used=800204096, at the size=10000

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

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
1000   1000   4      0.026      26.1378  9.632295e-13 3.284860e-02   pass
1000   1000   4      0.025      26.2410  9.632295e-13 3.284860e-02   pass
1000   1000   4      0.025      26.4202  9.632295e-13 3.284860e-02   pass
1000   1000   4      0.026      26.0428  9.632295e-13 3.284860e-02   pass
2000   2000   4      0.195      27.3354  4.746648e-12 4.129002e-02   pass
2000   2000   4      0.193      27.7365  4.746648e-12 4.129002e-02   pass
5000   5008   4      2.494      33.4359  2.651185e-11 3.696863e-02   pass
5000   5008   4      2.489      33.5038  2.651185e-11 3.696863e-02   pass
10000  10000  4      18.682     35.6959  9.014595e-11 3.178637e-02   pass
10000  10000  4      18.440     36.1648  9.014595e-11 3.178637e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
1000   1000   4       26.2104  26.4202
2000   2000   4       27.5359  27.7365
5000   5008   4       33.4698  33.5038
10000  10000  4       35.9303  36.1648

Residual checks PASSED

End of tests

Done: Fri Feb 12 00:20:25 UTC 2016
[ec2-user@ip-172-31-49-170 linpack]$
```

Explanations

CPU performance in terms of GFLOPS and GIOPS is gradually increase initially when doing millions of operations. Then after it is stable. So in the graph is shows stable value for billions of operations. We need to do billions of operations for CPU performance for better and consistent result. It gives accurate answer for 100000000 iterations for 1s and I took 600 samples by 10 mins so that I got consistent result.

Issued Faced

While measuring the CPU benchmark first thing I m confused with the IPC of processor so that I could not evaluate the theoretical peak performance. While I was doing program than faced problem with some JAVA time related functions and how to calculate cpu time and wall clock time.

2.2 Disk Benchmark

a. AWS-EC2 Cloud had disk to measure a performance of it in terms of disk throughput(MB/s) and latency(ms). For Disk Benchmark I did Read and Write operation on file of 1B, 1KB, 1MB sequential and Random. Find result for multiple experiment like Sequential Read for 1B with 1-Thread, Random Write for 1KB with 2-thread and so on.

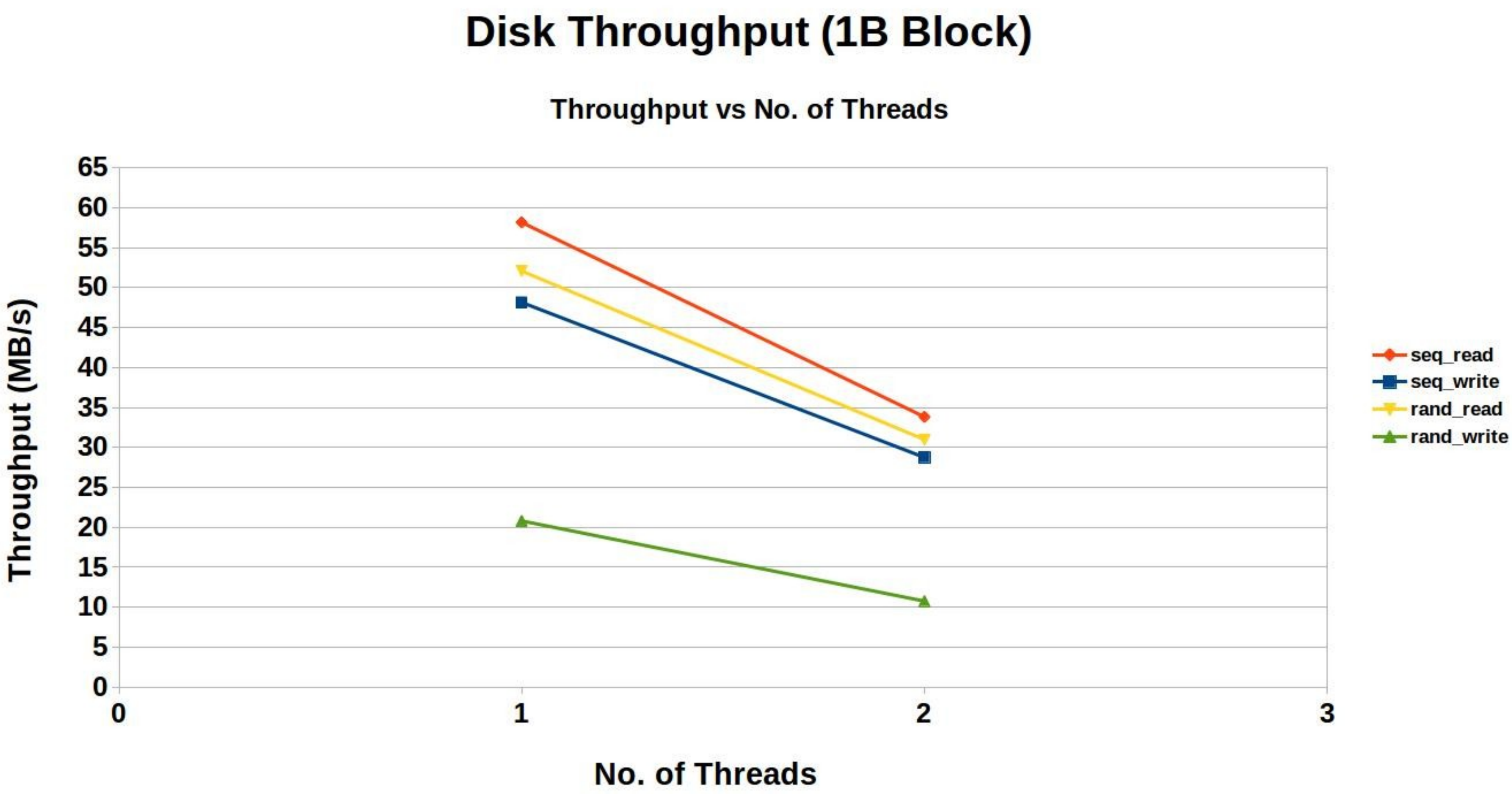
No of experiments = 2*2*3*2 = 24
(Read/Write) * (Sequential/Random) * (1B/1KB/1MB) * (1-thread/2-thread)

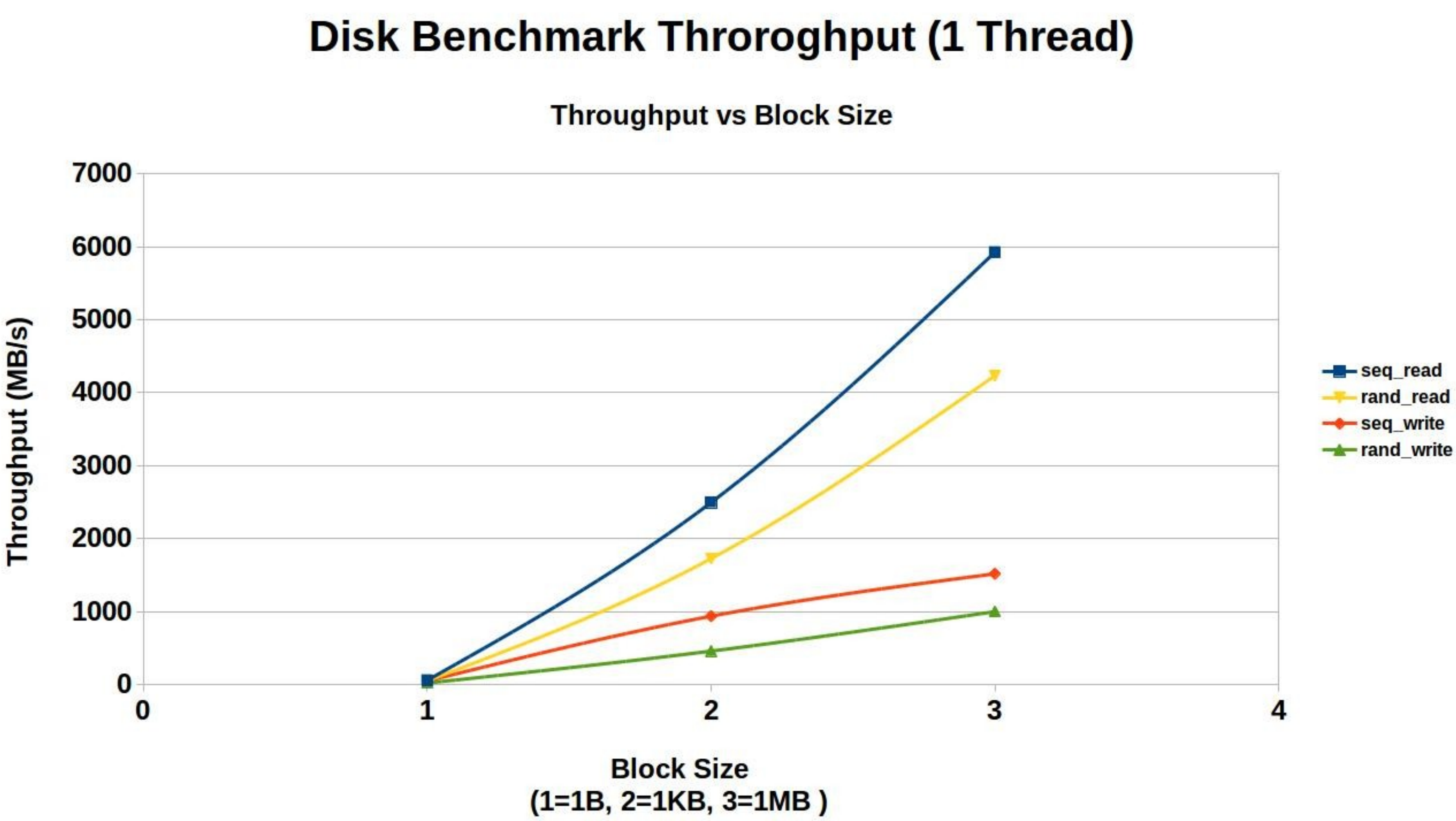
Disk performance in terms of throughput(MB/s) and latency (ms) calculated for all the experiments mentioned below in table.

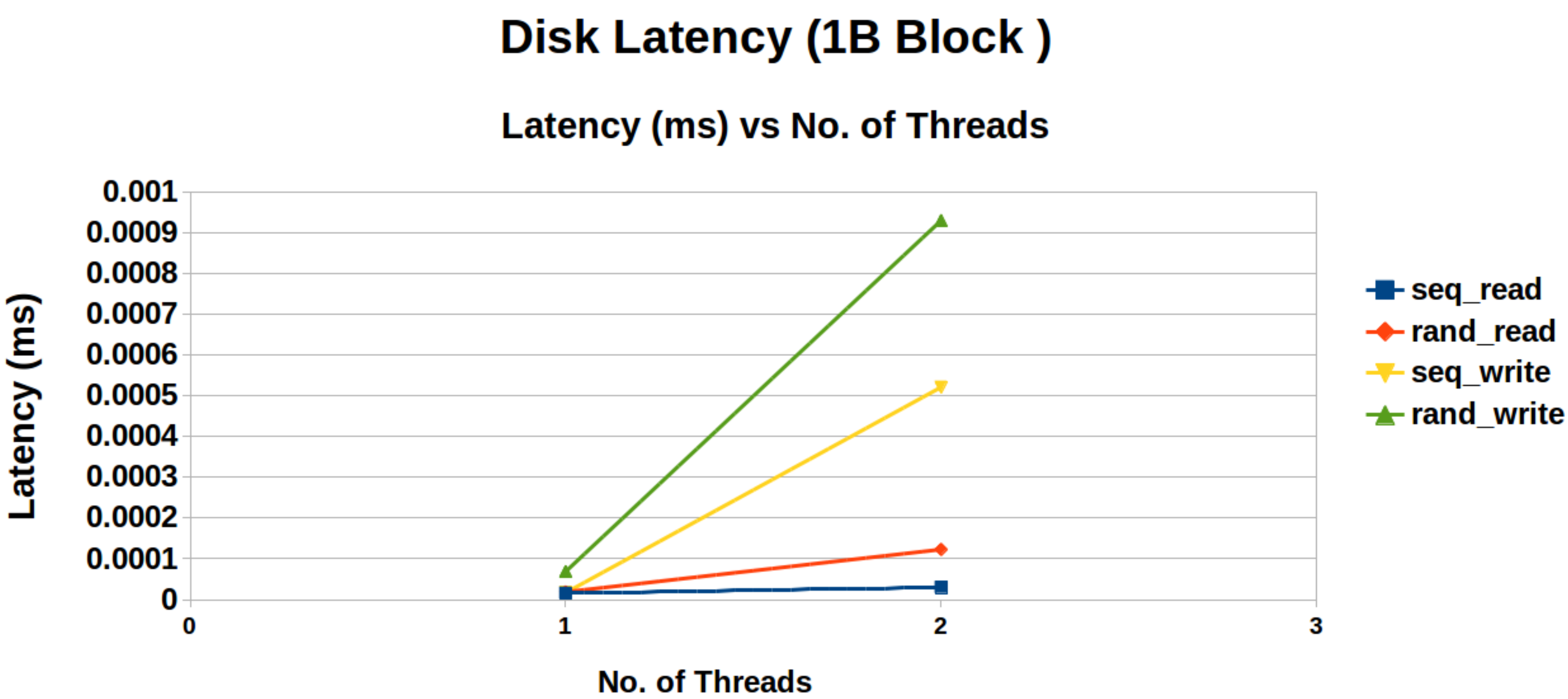
Practical throughput = **3745.318 MB/s** Practical Latency = **0.267 ms**
Theoretical Disk performance = 6.00 GB/s = 6000 MB/s (SATA V-3)
efficiency = (3754.318/6000) *100 = **62.42 %**

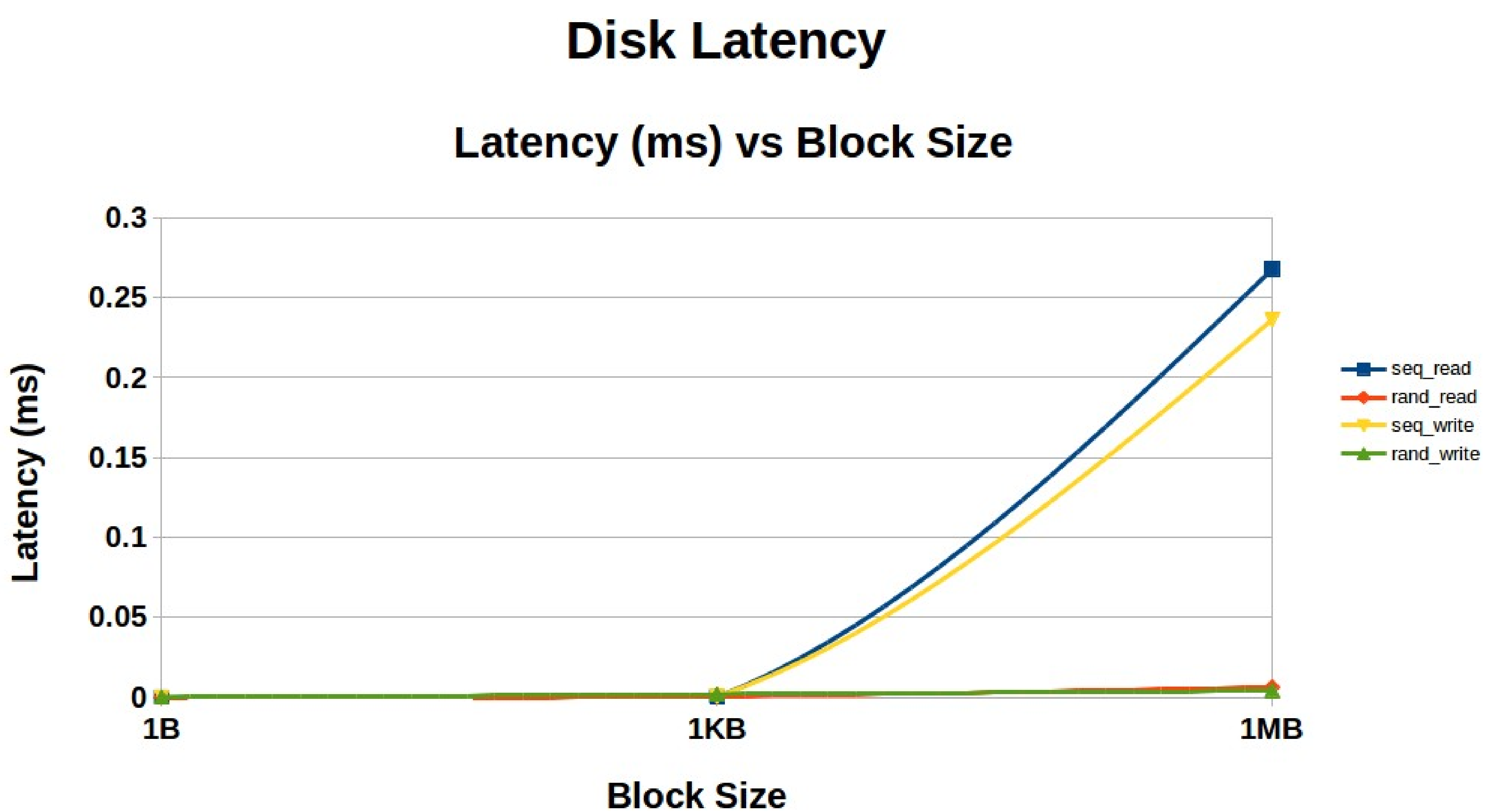
- Below graph is give the better idea about the comparison with the different experiments
- Sequential throughput always higher than Random throughput in both the case Read and Write.
- Write throughput is always less than the Read throughput and latency is reverse.
- Random access is costlier than Sequential access in terms of speed and accuracy.
- No of threads is increase than throughput is decrease in all cases because thread handling overhead.
- The lesser the latency, The Higher the performance.

- So, In terms of throughput(MB/s) and latency.






Disk Latency (1B Block)
Latency (ms) vs No. of Threads
Latency (ms)
0.001
0.0009
0.0008
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0
0
1
2
3
No. of Threads
seq_read
rand_read
seq_write
rand_write


Disk Latency
Latency (ms) vs Block Size
Latency (ms)
0.3
0.25
0.2
0.15
0.1
0.05
0
1B
1KB
1MB
Block Size
seq_read
rand_read
seq_write
rand_write

Sequential Read

Block Size	No. of Threads	Average(Throughput (MB/s))	SD(Throughput (MB/s))	Average (Latency (ms))	SD(Latency(ms))
1B	1	56.5849	1.01223	0.000016	0.000004
1B	2	32.4231	0.96992	0.000030	0.000003
1KB	1	2484.4534	1.09461	0.000183	0.000040
1KB	2	2137.2445	0.94674	0.000190	0.000010
1MB	1	5880.31311	0.95105	0.267799	0.003500
1MB	2	5180.6477	1.0991	0.274545	0.002300

Random Read

Block Size	No. of Threads	Average(Throughput (MB/s))	SD(Throughput (MB/s))	Average (Latency (ms))	SD(Latency(ms))
1B	1	52.071	0.64532	0.000019	0.000005
1B	2	30.9692	1.0403	0.000521	0.000032
1KB	1	1725.0571	0.67554	0.000798	0.000052
1KB	2	1566.4123	0.64284	0.000623	0.000076
1MB	1	4231.9685	0.51678	0.063230	0.000504
1MB	2	4023.3211	0.43572	0.167394	0.002034

Sequential Write

Block Size	No. of Threads	Average(Throughput (MB/s))	SD(Throughput (MB/s))	Average (Latency (ms))	SD(Latency(ms))
1B	1	49.0506	0.6180	0.000020	0.000008
1B	2	30.8317	0.88293	0.000123	0.000042
1KB	1	935.9977	0.9232	0.000566	0.000048
1KB	2	690.2321	1.0022	0.000587	0.000072
1MB	1	1224.2568	1.2133	0.063200	0.004030
1MB	2	1023.3121	0.8992	0.083211	0.000342

Random Write

Block Size	No. of Threads	Average(Throughput (MB/s))	SD(Throughput (MB/s))	Average (Latency (ms))	SD(Latency(ms))
1B	1	20.7958	0.9012	0.000069	0.000008
1B	2	10.7612	0.7812	0.000930	0.000078
1KB	1	508.9865	0.8233	0.001919	0.000204
1KB	2	412.3123	1.2110	0.001318	0.000123
1MB	1	998.5963	0.91212	0.042300	0.003279
1MB	2	732.1232	1.01311	0.303231	0.004321

Note

- Average and Standard Deviation of throughput and latency gives the better and accurate performance of disk.
- Average and Standard Deviation are calculated for each and every experiments thrice.
- For Disk Performance in every experiments **File Size** is 100 MB.

Practical Performance of Disk Speed on IOZONE Benchmark Tool:

- I used **iozone3_434** for benchmarking of disk of AWS_EC2 Cloud.
- Comparison Between Iozone result and My program Practical result for 1MB block size.

Access Type	**Iozone**	**My Practical Results**
Sequential Read	9485.232	5880.31311
Random Read	10230.845	4231.9685
Sequential Write	2192.645	1224.2568
Random Write	4282.950	998.5963

```
[ec2-user@ip-172-31-49-170 current]$ ./iozone -a -i 0 -i 1 -i 2 -s 1024
        Iozone: Performance Test of File I/O
                Version $Revision: 3.434 $
                Compiled for 64 bit mode.
                Build: linux

        Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
                     Al Slater, Scott Rhine, Mike Wisner, Ken Goss
                     Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
                     Randy Dunlap, Mark Montague, Dan Million, Gavin Brebner,
                     Jean-Marc Zucconi, Jeff Blomberg, Benny Halevy, Dave Boone,
                     Erik Habbinga, Kris Strecker, Walter Wong, Joshua Root,
                     Fabrice Bacchella, Zhenghua Xue, Qin Li, Darren Sawyer,
                     Vangel Bojaxhi, Ben England, Vikentsi Lapa,
                     Alexey Skidanov.

        Run began: Fri Feb 12 21:39:08 2016

        Auto Mode
        File size set to 1024 kB
        Command line used: ./iozone -a -i 0 -i 1 -i 2 -s 1024
        Output is in kBytes/sec
        Time Resolution = 0.000001 seconds.
        Processor cache size set to 1024 kBytes.
        Processor cache line size set to 32 bytes.
        File stride size set to 17 * record size.
                                                              random    random     bkwd    record    stride
              kB  reclen    write  rewrite    read    reread    read     write     read   rewrite      read   fwrite frewrite    fread  freread
            1024       4  1735476  4146499 11249091 12456195  9485232  4282950
            1024       8  2443368  5169641 13472053 15516181 10454984  5226256
            1024      16  2516377  4995276 12944224 14044758 12944224  5042191
            1024      32  2716948  4995276 12037272 13998981 13102174  5885083
            1024      64  2349794  5169641  9569770 15027577 13686709  6093831
            1024     128  2192645  4451639  9485232 13998981 13264026  6059442
            1024     256  2250080  4195100  9402175 12173747 11249091  5821271
            1024     512  2254805  4614246 10039528 11903823 11249091  5630486
            1024    1024  2056183  4898425 11614118 10230845 10662628  5593820

iozone test complete.
```

Fig : Iozone – 1MB multiple experiment

Theoritical Performance of Disk

```
[ec2-user@ip-172-31-49-170 ~]$ sudo hdparm -tT /dev/sda

/dev/sda:
 Timing cached reads:   21204 MB in  2.00 seconds = 10613.19 MB/sec
 Timing buffered disk reads: 244 MB in  3.00 seconds =  81.24 MB/sec
[ec2-user@ip-172-31-49-170 ~]$
```

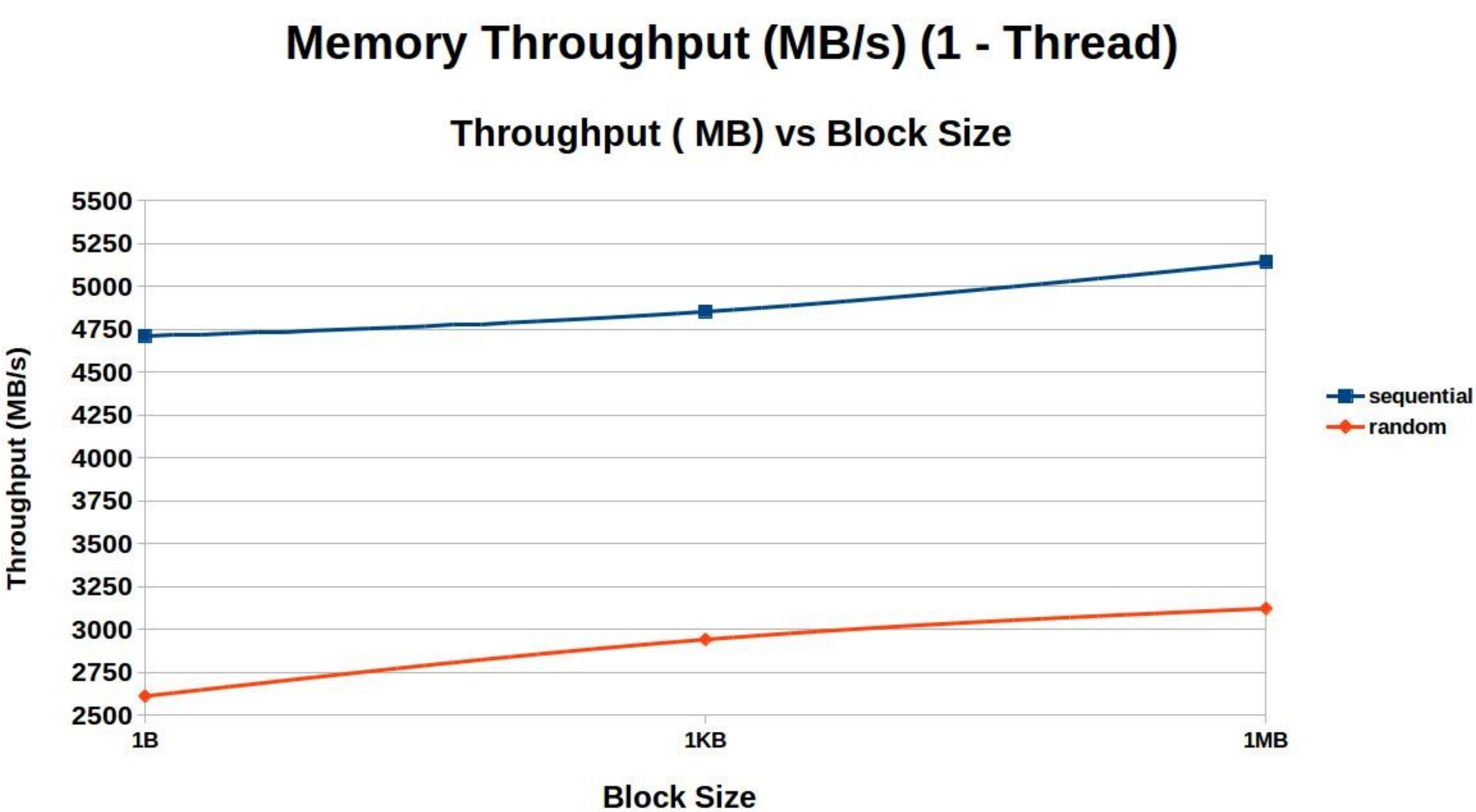
EFFICIENCY:

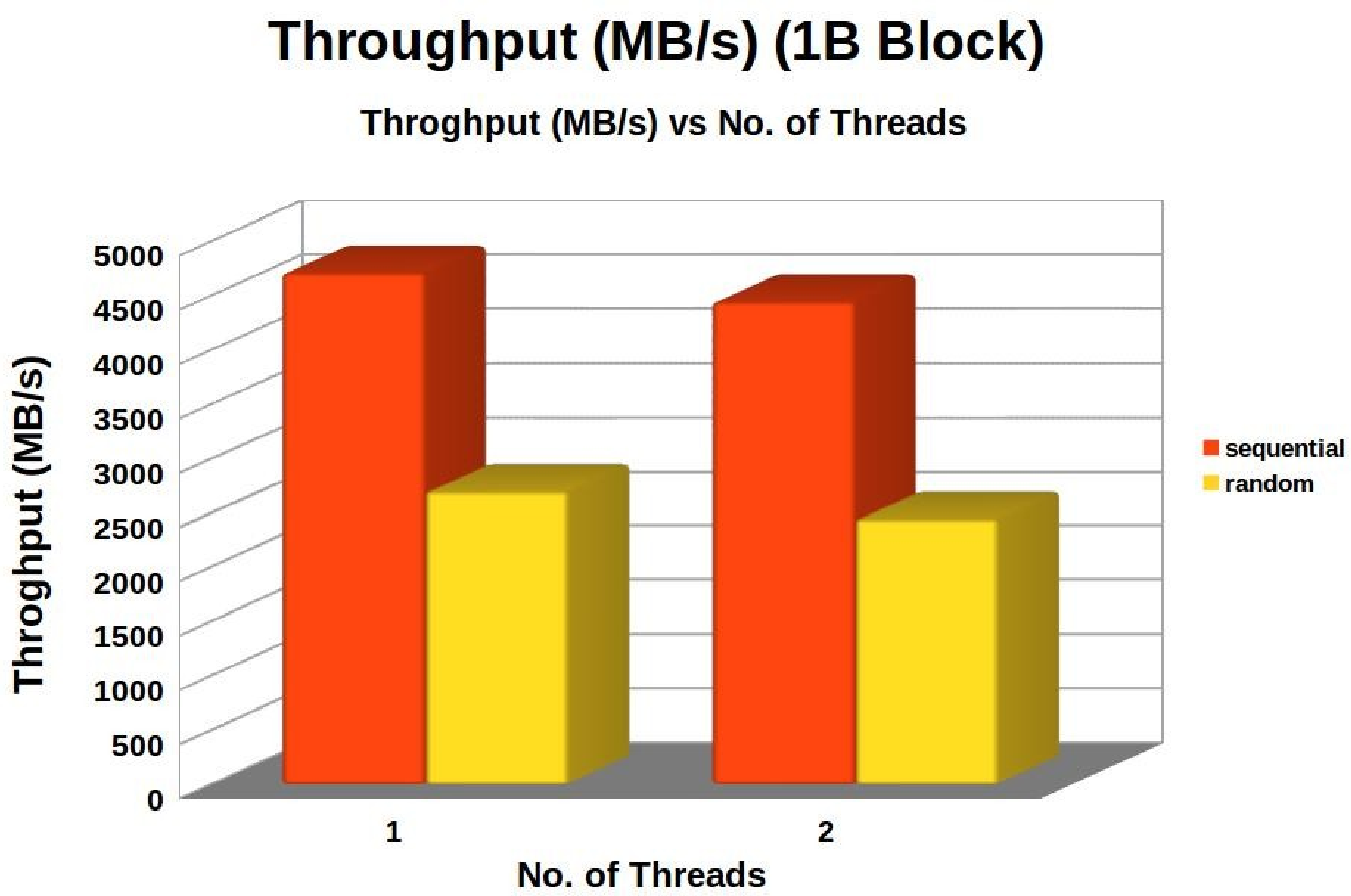
Efficiency of Iozone to Theoritical = (9485.232/10613.19) * 100
= **89.38 %**

2.3 Memory Benchmark

- Memory benchmark is for Sequential and random Access for different data size of 1B/1KB/1MB.
- All different experiment is done thrice and took Average and Standard Deviation for better accuracy and consistency.
- Thread Effect on Memory Throughput

  a.) As we increase the no of block size Sequential and Random throughput is increase
  b) As we increase the no of threads then Sequential and random throughput is decrease because thread overhead.
  c.) For all experiment Sequential throughput is always higher than Random Throughput.



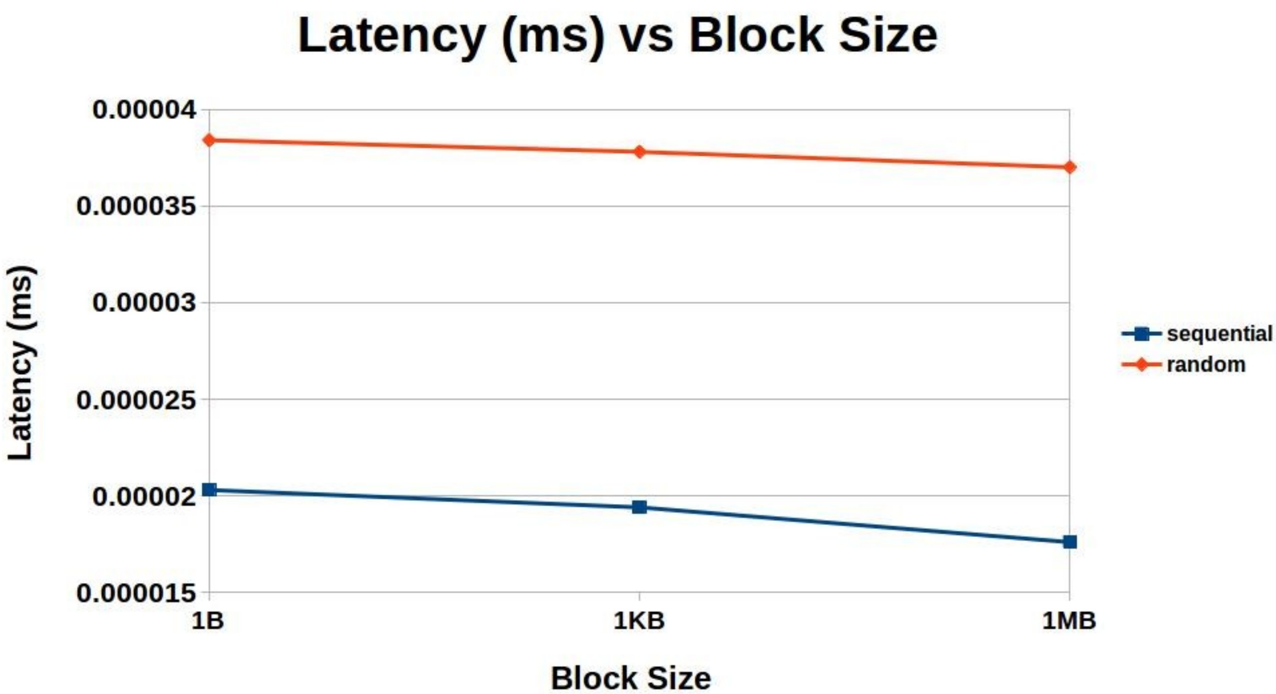


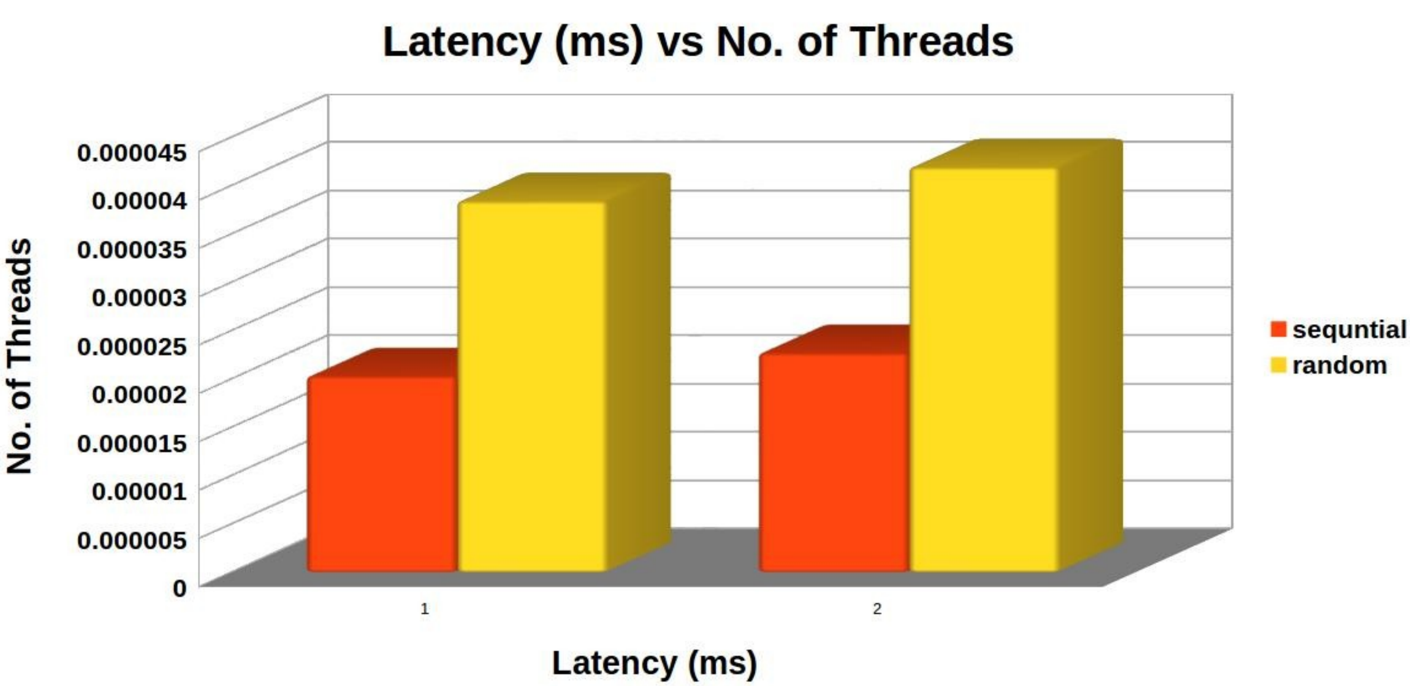
- Thread Effect on Memory Latency

  a.) As we increase the no of block size Sequential and Random Latency is decrease

  b) As we increase the no of threads then Sequential and Random latency is increase because thread overhead.

  c.) For all experiment Random Latency is always higher than Sequential Latency.


Latency (ms) vs Block Size
0.00004
0.000035
0.00003
0.000025
0.00002
0.000015
Latency (ms)
1B
1KB
1MB
Block Size
sequential
random


Latency (ms) vs No. of Threads
0.000045
0.00004
0.000035
0.00003
0.000025
0.00002
0.000015
0.00001
0.000005
0
No. of Threads
1
2
Latency (ms)
sequntial
random

Calculation

- Theoretical Throughput = **10600 MB/s** , Latency = **0.320 ms**
- The Optimal no of threads is 1, 1B Block size.
- Practical Throughput = **4710.211 MB/s** and Latency = **0.0000203 ms**
- Efficiency = **44.45 %**

Sequential Access

Block Size	No. of Threads	Average(Throughput (MB/s))	SD(Throughput (MB/s))	Average (Latency (ms))	SD(Latency(ms))
1B	1	4710.211	1.01223	0.0000203	0.000004
1B	2	4444.312	0.96992	0.0000227	0.000003
1KB	1	4853.131	1.09461	0.0000194	0.000040
1KB	2	4564.502	0.94674	0.0000224	0.000010
1MB	1	5143.322	0.95105	0.0000176	0.003500
1MB	2	4670.34	1.0991	0.0000215	0.002300

Random Access

Block Size	No. of Threads	Average(Throughput (MB/s))	SD(Throughput (MB/s))	Average (Latency (ms))	SD(Latency(ms))
1B	1	2698.231	0.64532	0.0000384	0.000005
1B	2	2444.98	1.0403	0.0000419	0.000032
1KB	1	2745.131	0.67554	0.0000378	0.000052
1KB	2	2493.05	0.64284	0.0000410	0.000076
1MB	1	2796.321	0.51678	0.0000370	0.000504
1MB	2	2570.56	0.43572	0.0000402	0.002034

STREAM Benchmark Result Analysis

- **I** run Stream Benchmark and result is given in the below screen shot.
- Theoretical Throughput = **10600 MB/s** , Latency = **0.320 ms**
- While practical Throughput **= 6714.45 MB/sec and** Latency **= 0.03023 ms**
- Efficiency for Throughput **= 63.35 %**
- Efficiency for Latenc**y = 0.95 %**

```
[ec2-user@ip-172-31-49-170 Memory]$ gcc stream.c
[ec2-user@ip-172-31-49-170 Memory]$ ./a.out
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 10000000 (elements), Offset = 0 (elements)
Memory per array = 76.3 MiB (= 0.1 GiB).
Total memory required = 228.9 MiB (= 0.2 GiB).
Each kernel will be executed 10 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 28764 microseconds.
   (= 28764 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    Best Rate MB/s  Avg time     Min time     Max time
Copy:            5411.4     0.029769     0.029567     0.030050
Scale:           5315.3     0.030266     0.030102     0.030448
Add:             7699.2     0.031382     0.031172     0.031707
Triad:           7231.3     0.033374     0.033189     0.033571
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
[ec2-user@ip-172-31-49-170 Memory]$
```