

CS 553 Cloud Computing
Programming Assignment 1
Name: Kedar Kaushikkar (A20355218)

Performance Evaluation Report

Introduction:

This report gives the performance evaluation of various system components like CPU, MEMORY, and DISK. The experiments were performed several times and average of each experiments are used to depict the final results. Each component module has been thoroughly analyzed on the parameters of System used for experiments, Graphs, Actual Results obtained, Theoretical Peak Performance, Efficiency and conclusions.

Also the report includes results from other benchmarks like Linpack, Iozone & Stream and the results of those benchmarks are compared with our benchmarks and efficiency is calculated.

System Configuration:

- Operating System: UBUNTU 14.04(Dual Boot)
- Model: Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz
- RAM: 8GB

CPU Benchmarking

This module aims to calculate the processing speed in terms of Giga FLOPS and Giga IOPS on different levels on concurrency. (1, 2 and 4).

Calculations:

$$\text{GFLOPS} = \frac{\text{Number of Operations} * \text{No of Times operations performed}}{\text{Time Elapsed} * 10^9}$$

$$\text{GIOPS} = \frac{\text{Number of Operations} * \text{No of Times operations performed}}{\text{Time Elapsed} * 10^9}$$

- The theoretical peak performance is calculated using the following parameters,
- No of Operations: Several arithmetic operations (30) are performed for large number of times. A large number is used to get accurate results. In this experiment the no of times the operation is performed is 10^9 .
- Time Elapsed: The time elapsed is the maximum time taken amongst any thread at any concurrency level to complete calculating all the operations.

Benchmarking Results:

CPU speed is calculated in terms of Giga FLOPS and Giga IOPS using threads 1, 2 and 4.
The average results obtained after performing the benchmarking an repeating it for 5 times is given below,

1 Thread:

Sr. No	GIOPS	Time	GFLOPS	Time
1 Thread				
1	12.1396332	2.635994	12.1400156	2.635911
2	12.1422958	2.635416	12.1809731	2.627048
3	12.1320305	2.637646	12.1415081	2.635587
4	12.170785	2.629247	11.3150339	2.828096
5	12.9137812	2.477973	12.8850508	2.483498
Average	12.2997051		12.1325163	

2 Threads:

Sr. No	GIOPS	Time	GFLOPS	Time
2 Thread				
1	12.1254539	5.278153	12.1036069	5.257275
2	12.0686016	5.303017	12.1730728	5.257506
3	12.1237917	5.278877	11.341507	5.642989
4	12.0975485	5.290328	12.0522575	5.275193
5	12.9072618	4.958449	12.9235964	4.952182
Average	12.2645315		12.1188081	

4 Threads:

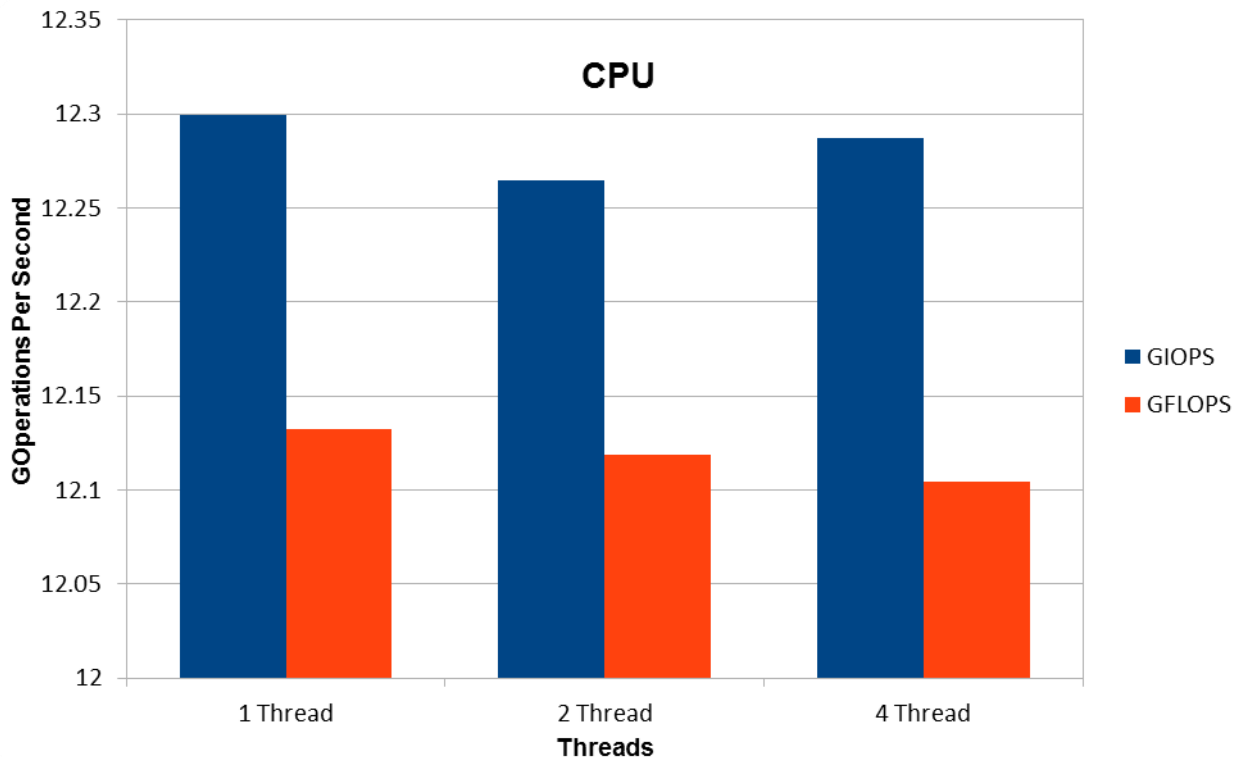
Sr. No	GIOPS	Time	GFLOPS	Time
4 Thread				
4	12.1252699	10.556466	12.0170698	10.651515
4	12.1158838	10.564644	11.6989565	10.941147
4	12.1264477	10.555441	12.1113195	10.568625
4	12.1474581	10.537184	12.1268585	10.563794
4	12.9193935	9.907586	12.5695662	9.907454
Average	12.2868906		12.1047541	

As per the result data shown above, the results obtained are

GIOPS for 1 Thread: **12.2997051**
GIOPS for 2 Threads: **12.2645315**
GIOPS for 4 Threads: **12.2868906**

GFLOPS for 1 Thread: **12.1325163**
GFLOPS for 2 Threads: **12.1188081**
GFLOPS for 4 Threads: **12.1047541**

- Hence the number of GIOPS for different concurrency levels is more than the the number of GFLOPS for all concurrency levels.
- Below shown is the data chart displaying the GIOPS and GFLOPS for 1 , 2 and 4 concurrency level.



Theoretical Peak Performance:

The theoretical peak performance can be calculated using the below formula

$$\text{GFLOPS} = \text{No of Cores} * \text{Clock Cycle} * \text{Instructions Per Cycle}$$

System Configuration on Amazon t2 micro instance is

Model: Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz

CPU Cores: 1

IPC: The system uses AVX2 instruction set extensions that can support 2x the FLOPS per core (16 Flops/Clock). Considering the minimum Flops = x, IPC = 8

$$\begin{aligned}\text{GFLOPS} &= 1 * 2.40 * 8 \\ &= 19.20 \text{ Instructions per second.}\end{aligned}$$

Efficiency:

The Efficiency can be measured as a comparison of actual results obtained to the theoretical performance of 30720 the system.

$$\text{Efficiency} = \frac{\text{Actual Performance}}{\text{Theoretical Peak Performance}} * 100$$

According to the result obtained, the efficiency can be calculated as (FLOPS for 1 thread / Theoretical Peak Performance) * 100

$$\begin{aligned} &= (12.13 / 19.20) * 100 \\ &= 63.17 \% \text{ Efficiency.} \end{aligned}$$

Conclusion:

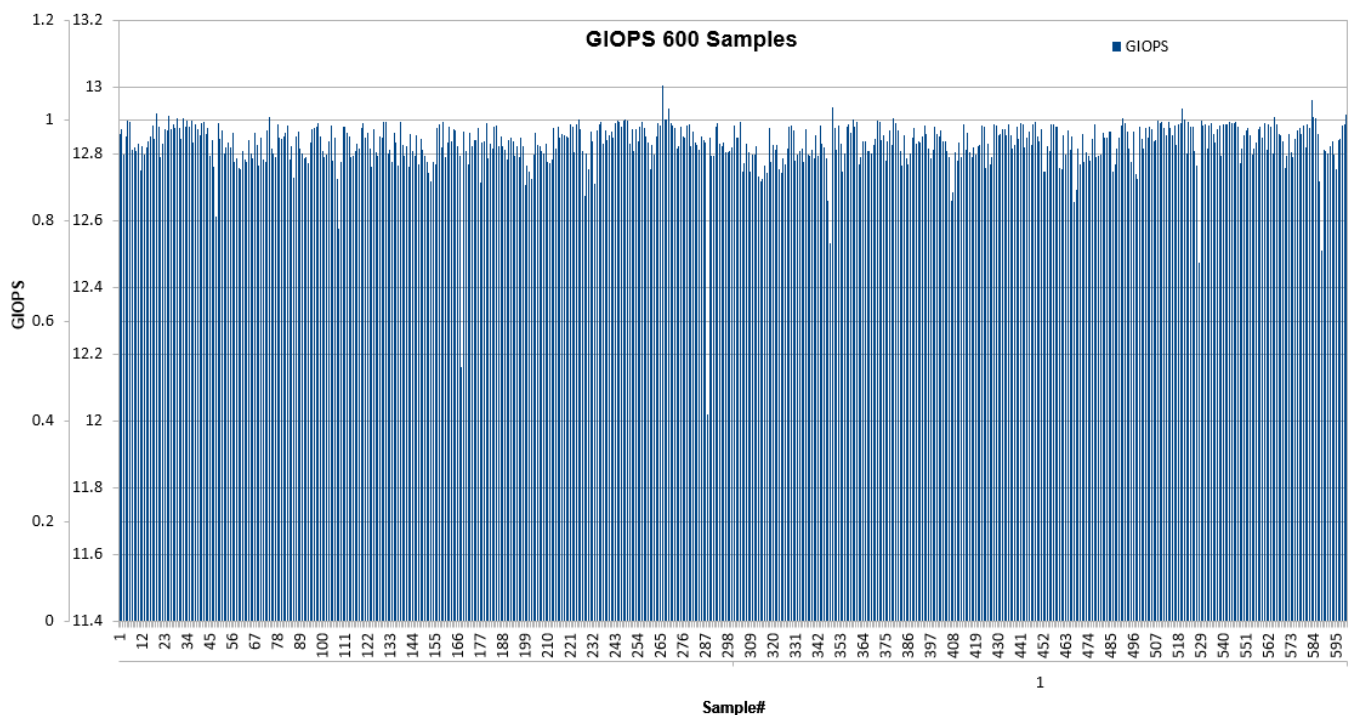
The optimal number of concurrency level for a good performance is 1. However, comparing the results with performance of other concurrency levels (2 & 4), the difference is very minimum.

The performance at concurrency level 1 is optimal since more threads bring up the overhead of thread maintenance and also time wait between switching between threads i.e. Wait time for other threads to complete their operations is also added.

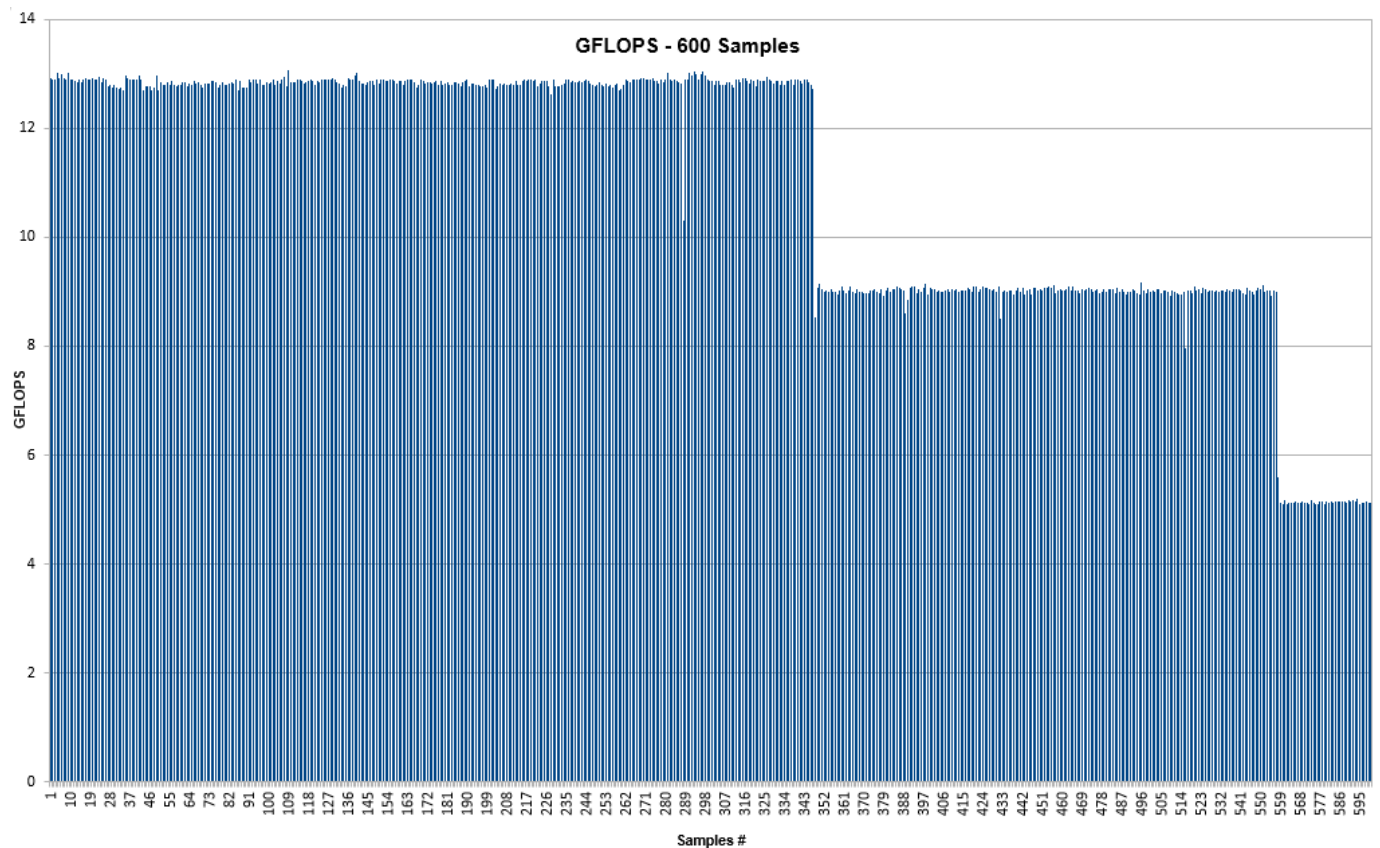
CPU Benchmarking for 600 Samples

600 Samples for GIOPS and GFLOPS were recorded for concurrency level of 4 and the result data chart for GIOPS and GFLOPS is given below,

For GIOPS, the number of instructions per second ranges from 12.1 to 13.25 GIOPS



For GFLOPS, the number of instructions per second ranges from 5.92 to 13.01 GFLOPS



Extra Credits

The Linpack benchmarking tool was used to compare the accuracy of the results with the theoretical peak value obtained through this tool.

Below screenshot displays the Linpack Benchmarking results on Amazon T2 Micro Instance.

```
ubuntu@ip-172-31-56-178:~/linpack_10.3.4/benchmarks/linpack$ ./
drwxr-xr-x 4 ubuntu ubuntu 4096 May 5 2011 ./
drwxr-xr-x 4 ubuntu ubuntu 4096 May 5 2011 ../
drwxr-xr-x 2 ubuntu ubuntu 4096 Feb 9 03:10 linpack/
drwxr-xr-x 11 ubuntu ubuntu 4096 May 5 2011 mp_linpack/
ubuntu@ip-172-31-56-178:~/linpack_10.3.4/benchmarks$ cd linpack/
ubuntu@ip-172-31-56-178:~/linpack_10.3.4/benchmarks/linpack$ ll
total 7484
drwxr-xr-x 2 ubuntu ubuntu 4096 Feb 9 03:10 ./
drwxr-xr-x 4 ubuntu ubuntu 4096 May 5 2011 ../
-rwxr-xr-x 1 ubuntu ubuntu 167 May 5 2011 help.lpk*
-rwxr-xr-x 1 ubuntu ubuntu 446 May 5 2011 lininput_xeon32*
-rwxr-xr-x 1 ubuntu ubuntu 439 May 5 2011 lininput_xeon64*
-rw-rw-r-- 1 ubuntu ubuntu 1553 Feb 9 03:12 lin_xeon64.txt
-rwxr-xr-x 1 ubuntu ubuntu 309 May 5 2011 runme_xeon32*
-rwxr-xr-x 1 ubuntu ubuntu 309 May 5 2011 runme_xeon64*
-rwxr-xr-x 1 ubuntu ubuntu 11850 May 5 2011 xhelp.lpk*
-rwxr-xr-x 1 ubuntu ubuntu 3041082 May 5 2011 xlinpack_xeon32*
-rwxr-xr-x 1 ubuntu ubuntu 4672750 May 5 2011 xlinpack_xeon64*
ubuntu@ip-172-31-56-178:~/linpack_10.3.4/benchmarks/linpack$ ./xlinpack_xeon64
Input data or print help ? Type [data]/help :
Number of equations to solve (problem size): 10000
Leading dimension of array: 10000
Number of trials to run: 4
Data alignment value (in Kbytes): 4
Current date/time: Thu Feb 11 03:20:07 2016
CPU frequency: 2.828 GHz
Number of CPUs: 1
Number of cores: 1
Number of threads: 1
Parameters are set to:
Number of tests : 1
Number of equations to solve (problem size) : 10000
Leading dimension of array : 10000
Number of trials to run : 4
Data alignment value (in Kbytes) : 4
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)
10000 10000 4 32.546 20.4898 1.124127e-10 3.963786e-02
10000 10000 4 32.808 20.3262 1.124127e-10 3.963786e-02
10000 10000 4 32.595 20.4594 1.124127e-10 3.963786e-02
10000 10000 4 32.427 20.5655 1.124127e-10 3.963786e-02
Performance Summary (GFlops)
Size LDA Align. Average Maximal
10000 10000 4 20.4602 20.5655
End of tests
ubuntu@ip-172-31-56-178:~/linpack_10.3.4/benchmarks/linpack$
```

This benchmark was initialized with input as

No of Tests = 1

No of Equations to Solve = 10000

Leading Dimensions of Array = 10000

No of Trials to Run = 4

No of CPU = 1

No of Cores = 1

No of Threads = 1

The Linpack benchmark calculated the Average Theoretical Peak GFLOPS = 20.4602 Instructions per second.

$$\begin{aligned}\text{Efficiency Achieved} &= (\text{GFLOPS for 1 Thread} / \text{Theoretical Peak Performance}) * 100 \\ &= (12.13/20.4602) * 100 \\ &= 59.28\%\end{aligned}$$

The CPU benchmark gave efficiency of 59.28% when compared to Linpack Benchmark Theoretical peak Results.

DISK Benchmarking

This module aims to calculate the disk speed in terms of read and write for various block size of 1B, 1KB, 1 MB and accessing the disk sequentially or randomly.

Calculations

- The experiments and benchmarking done for this module gives us the throughput and latency for accessing disk & performing operations for different block sizes.
- **Cache Size:** 30720.00 Kb
- The text file used for benchmarking is 40Mb. The file size used to benchmark is greater than page cache so that it will fit the file contents in any of the caches and hence will generate accurate results as only disk memory will be used.
- Disk will be accessed sequentially and randomly.
- Disk operations include reading from file on the disk and writing in the file on the disk.
- Hence the performance will be measured on the Throughput and Latency of the disk for the operations on difference file access methods.

$$\text{Throughput} = \frac{\text{No_of_operations} * \text{Num_Bytes}}{\text{Time_Elapsed} * 1024 * 1024} \text{ Mb/Sec}$$

No_of_operations: This is the total number of times the file is read or written to / from the disk.

Num_Bytes: The Bytes Read or written for each operation.

1B = 4000000 times (40Mb)

1Kb = 40000 times (40Mb)

1Mb = 40 times (40Mb)

Time Elapsed: The time taken to read or write 1B, 1Kb or 1Mb for a 40Mb file.

Latency: 1/Throughput

Benchmarking Results:

The Benchmarking result data shows the throughput for 1 Thread & 2 Threads Sequential and Random file access for 1B , 1Kb , 1Mb block size read/write operations.

1 Thread

	1B Read	1B Write	1Kb Read	1Kb Write	1Mb Read	1Mb Write
Sequential	35.120689	10.51446	4136.64258	1794.09119	7083.17822	3165.57227
	35.02203	9.933911	4047.13086	1860.82434	6755.47266	3311.20557
	35.751953	10.517959	4296.42822	1899.66028	4857.64014	3411.52881
Average	35.4116563	10.3710723	4194.15747	1863.55902	5888.48279	3324.95886

	1B Read	1B Write	1Kb Read	1Kb Write	1Mb Read	1Mb Write
Random	1.402331	0.529213	1191.65894	564.517822	7668.8833	3613.67651
	1.41225	0.501983	1221.92053	567.430908	8329.88281	3880.56079
	0.5127	0.5127	593.466248	593.466248	8001.34277	4514.12988
Average	1.10909367	0.514632	1002.34857	575.138326	8000.0363	4002.78906

2 Thread

	1B Read	1B Write	1Kb Read	1Kb Write	1Mb Read	1Mb Write
Sequential	17.595249	5.270917	2215.37842	997.157166	3736.78174	1925.87451
	17.266323	5.208666	2292.35669	929.584229	6745.14746	2012.29407
	17.386471	5.176753	1036.37891	932.081299	5496.95801	1738.10437
Average	17.4160143	5.21877867	1848.038	952.940898	5326.29574	1892.09098

	1B Read	1B Write	1Kb Read	1Kb Write	1Mb Read	1Mb Write
Random	0.687568	0.261782	671.5896	296.367371	5273.70801	2446.34473
	0.698517	0.202971	672.381592	304.878632	5035.48047	2033.0293
	0.720017	0.263469	658.052612	305.822723	4811.75391	2437.30908
Average	0.702034	0.24274067	667.341268	302.356242	5040.31413	2305.56104

- Three result sets were recorded for disk benchmarking and average of those results is calculated.
- The above results are shown for disk accessed sequentially and randomly for two different concurrency level for variable block sizes.

Based on the result sheets obtained, the average data chart is plotted for read /write operations.

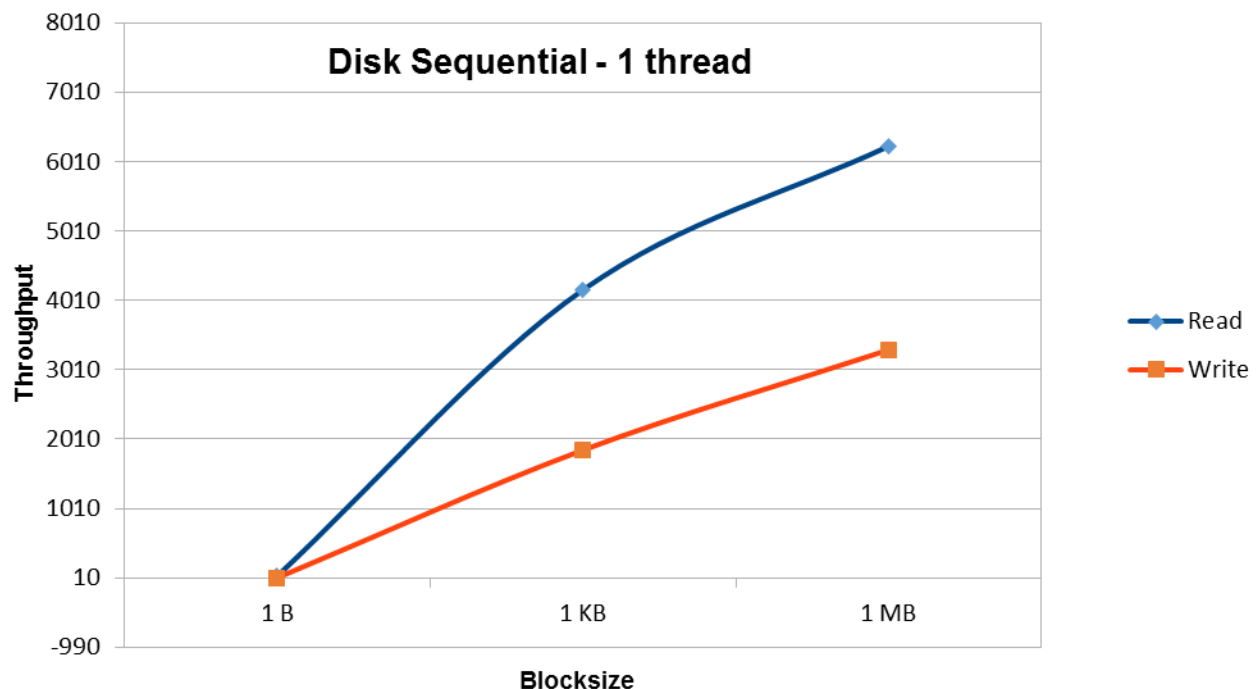


Fig. : Read/Write Sequential Disk access with 1 Thread.

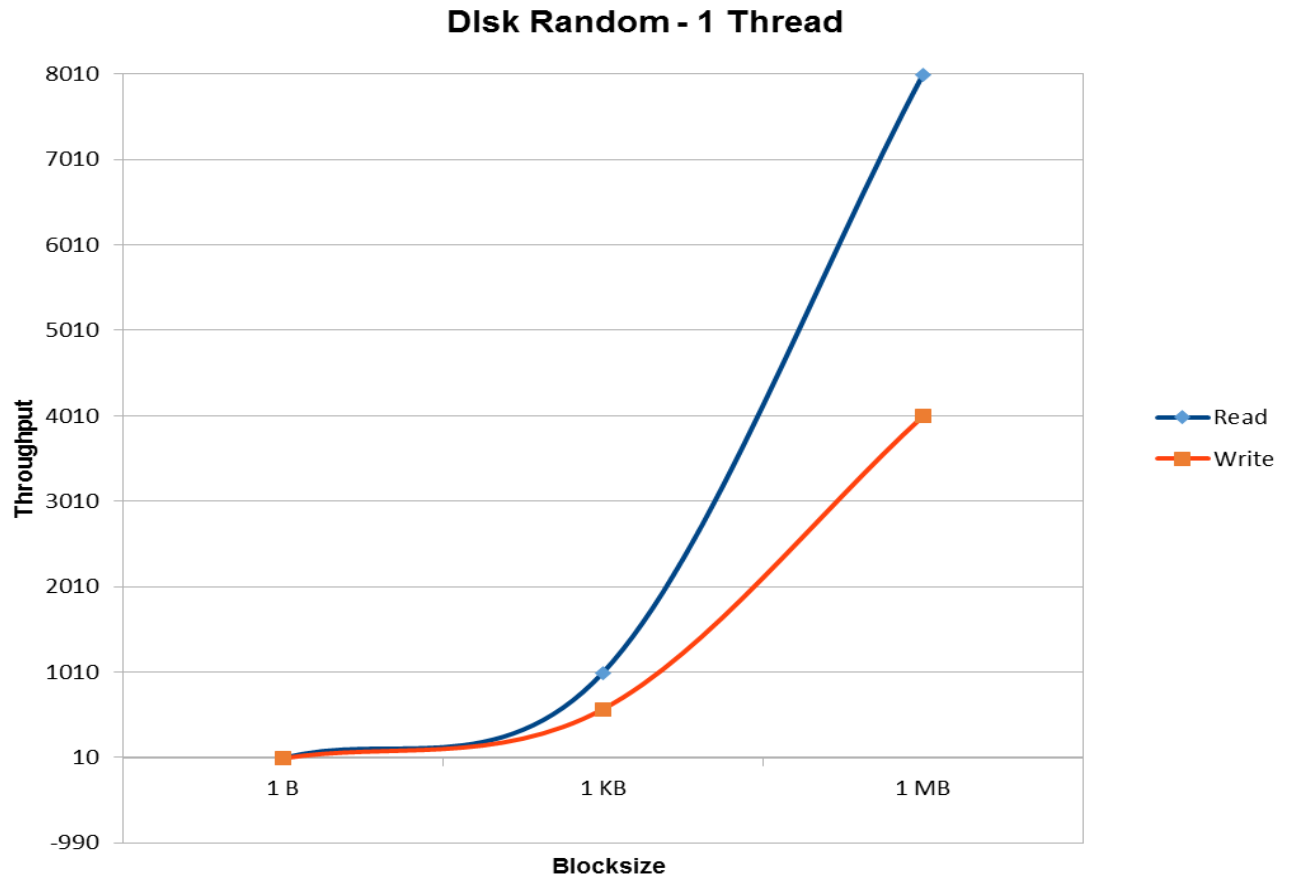


Fig. : Read/Write Random Disk access with 1 Thread.

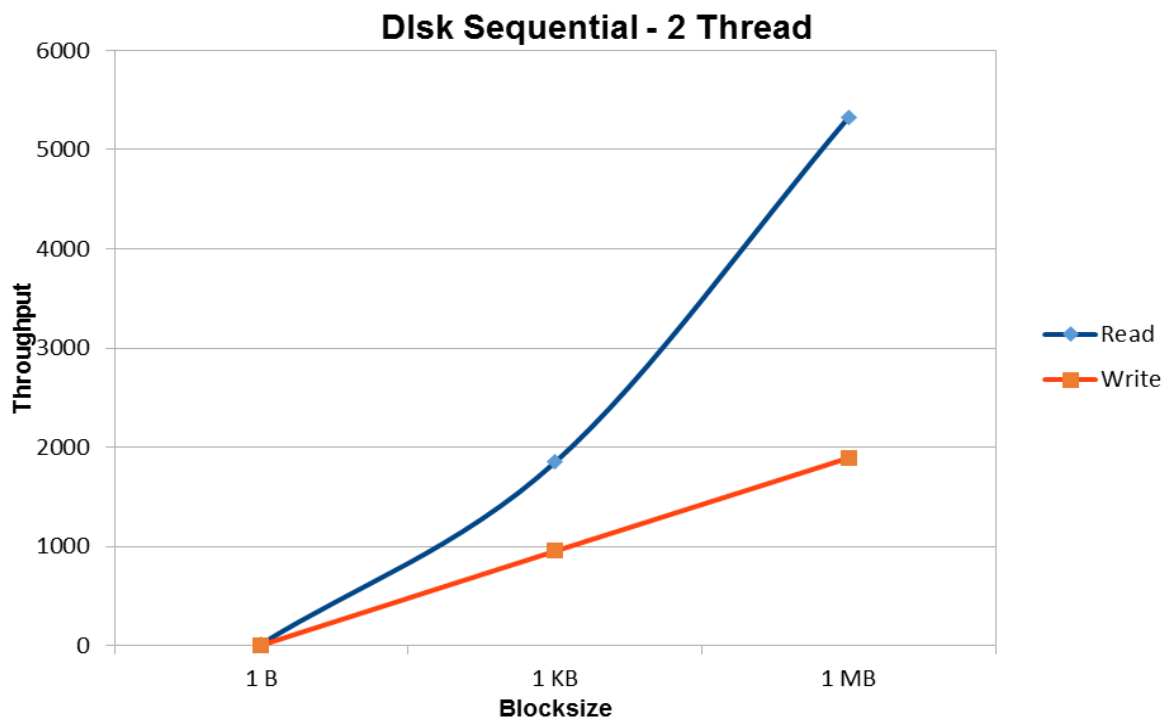


Fig. : Read/Write Sequential Disk access with 2 Thread.

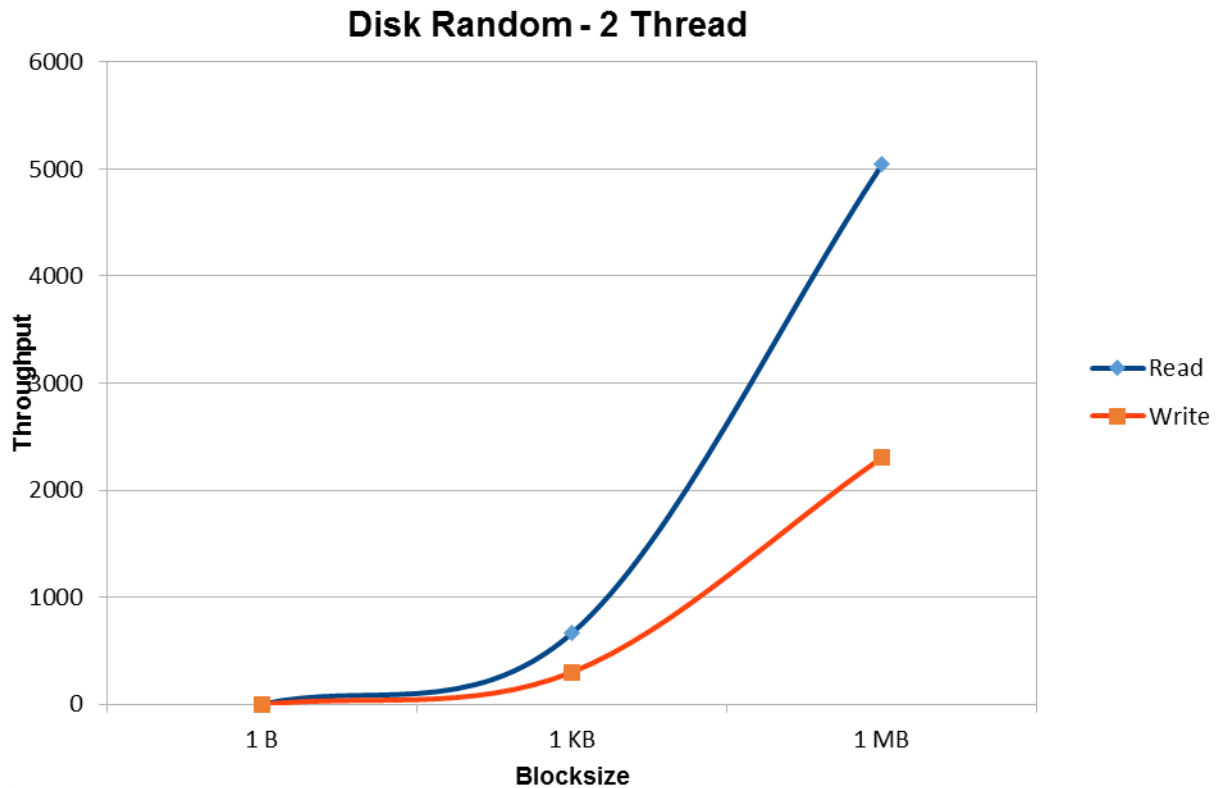


Fig. : Read/Write Random Disk access with 2 Thread.

Conclusion:

- The optimal performance achieved is for concurrency level 1. The average throughput and Latency shown in the results and graphs shows the optimal performance at concurrency level 1.
- Accessing the file randomly for 1Mb block size and performing read/write operations on concurrency level 1 gives optimal performance as less blocks are accessed since the block size is high.
- Also as per the data charts the random access is slower as compared to sequential since there is additional overhead of seeking the file pointer to any random location across the disk and then perform read/write operations on that location.

Extra Credits

IOzone Benchmark is used to find the best performance of the system and this performance is compared with the result set that we receive through this benchmarking program.

```

ubuntu@lsp-172-31-56-170: ~
Al Slater, Scott Rhine, Mike Misner, Ken Goss
Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CVR,
Randy Dunlap, Dan Montague, Dan Millon, Gavin Brehner,
Jean-Marc Zucconi, Jeff Blomberg, Benny Halevy, Dave Boone,
Eric Habbington, Kris Strecker, Walter Wong, Joshua Root,
Fabrice Bacchella, Zhenghua Xue, Qin Li, Darren Sawyer,
Vangel Bojxhi, Ben England, Vikentii Lapa.

Run began: Thu Feb 11 23:31:24 2016

Auto Mode
Command line used: lozone -a
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.

KB   reclen   write  rewrite   read   reread   read   write   read   write   stride
64   4 1143223 3363612 12902017 12902017 9060179 4018152 2772930 4274062 7100397 3022727 3588436 4897948 7100397
128 4 170088 4274062 15972885 12902017 12902017 3363612 7100397 3022727 5800307 3541098 4643754 79404539 15972885 3022727
64 16 1648808 4207070 12902017 12902017 12310336 3363612 4564786 4274062 12902017 4018152 4897948 79404539 9060179
64 32 127886 4988978 15972885 15972885 12902017 4897948 6271021 3363612 5800307 3541098 4018152 5389653 9318832
64 64 1421755 4643754 20962191 20962191 12902017 4988978 6421025 3541098 7100397 4274062 79404539 15972885
128 4 1520043 3759450 7548891 1720614 9129573 3124872 7582312 5122535 6114306 3053774 3124872 7470716 8548124
128 8 183330 426114 1400974 15810781 1720614 4717434 6272225 4717434 9129573 7835961 4572575 9795890 40779307
128 16 1939552 4596273 14200794 15810781 2542043 5325799 8036384 5603747 11720614 4596273 4759253 9795890 11720614
128 32 2166499 5325799 16365173 15810781 4200794 5379161 7917784 4717434 8548124 4934216 2985839 7582312 12842051
128 64 2698744 4934216 1581078 15810781 4200794 5545860 9129573 4889281 8036304 3445772 5379161 10567140 11720614
128 128 827299 4596273 1581078 15810781 2842051 4717434 4889281 3785961 12842051 5784891 5603747 9977956 10567140
256 4 1578419 426114 1400974 1228612 9434848 3039291 571923 5022844 6719046 3310006 3939521 5320671 8004634
256 8 1694418 4553582 11091721 15164624 2661199 4332998 8534922 5810112 10651598 4422226 3285566 9192546 13454450
256 16 1523457 5117791 14164395 15164624 3454450 5320671 7791788 5428265 11091721 4572895 4840911 11091721 13454450
256 32 2223932 5569035 14535374 14953435 31625180 5117791 10148242 6249745 10651598 4572895 51275915 569783 1453744
256 64 1649865 5022044 14953435 16072608 4135374 5320671 10245071 5810112 9868433 4496299 6107548 8271916 14164395
256 128 249330 5117791 13454450 15164624 11095808 5320671 9778562 6249745 11569703 3970045 7735574 9114515 13454450
256 256 1985448 4929815 12812277 13454450 81212277 5022044 9868433 5687020 10245071 5687020 5810112 11207494 12812277
512 4 1684557 4395250 12215097 12146009 9468384 3970896 8665997 5384786 7871843 3816200 4068701 6571132 10235583
512 8 1592458 4228947 9145790 14628067 6571132 4195896 9468384 6652558 9145790 3905895 4195896 8808176 13872122
512 16 2337255 5019704 10325583 14240609 3152711 45228681 1138071 6652558 11138071 4973263 5019704 14249940 1034519
512 32 2485320 603935 14240609 124994901 397122 4660200 5624545 6652558 13872122 5007142 52788923 1522711 15037001
512 64 2708713 5278892 13872122 154711491 4628067 5886580 1215097 6632011 15037801 5331314 5227492 8528336 14628067
512 128 4846072 5227492 12797441 14628067 13783088 58228041 1374040 6395018 13872122 5495016 5989595 11877300 14146265
512 256 6259509 5910762 10641343 12797441 12797441 5278892 10284602 5886655 11374040 3296661 621022 10694340 8665997
512 512 2099500 5067142 11374040 124994901 1246009 5384786 9145790 5068495 10911094 5684095 471003 7309196 12499490
1024 4 193279 426114 1400974 1089510 965538 3369115 7273794 5021115 11808150 1089510 6918376 441509 35480997 1400974
1024 8 2315588 4265934 11773001 1422045 10309552 5071963 1348754 7208671 11498939 4810640 4044965 11645609 14230902
1024 16 2645000 4555516 10280636 15516181 13818817 4820689 9045736 6650556 11773001 5105690 4433259 12348754 14422045
1024 32 2708381 4762630 13863422 14820616 14044758 5885083 7755368 6830356 12983353 4984245 4140497 8137402 13643232
1024 64 2932998 5751117 14044758 15516181 14820161 5144870 9832672 6874084 12983353 4014240 4783849 14044758 15295157
1024 128 212335 588433 14044758 14999991 6093833 14989839 6093833 13863422 5090934 4589593 1279037 14044758
1024 256 2708381 5310258 11773001 12348754 152208350 5660167 8061038 5885083 11903823 5356618 4451639 9220512 12494246
1024 512 2612819 5120336 9464330 10557785 12
```

As shown in the screenshot, iозone 3 is run on the amazon t2 micro instance. The Output is in the format of **Kb/Sec**.

Theoretical Peak Performance

As per Iozone 3, Write 1Mb with Record Length of 1024,
Iozone Throughput fwrite = 5917.516Mb/sec
Benchmark fwrite = 3324.9588

$$\begin{aligned}\text{Efficiency (write)} &= (\text{Benchmark Result} / \text{Theoretical Result}) * 100 \\ &= (3324.9588/5917.516) * 100 \\ &= 56.18 \%\end{aligned}$$

Iozone Throughput fread = 8750.850Mb/sec
Benchmark fread = 5888.4827

$$\begin{aligned}\text{Efficiency (read)} &= (\text{Benchmark Result} / \text{Theoretical Result}) * 100 \\ &= (5888.4827/8750.850) * 100 \\ &= 67.29\%\end{aligned}$$

Memory Benchmarking

This module aims to calculate the performance of the memory speed on the parameter of throughput and latency. Memory speed is being tested by performing read write operations and accessing the memory sequentially and randomly for different block size of 1B, 1Kb, 1Mb for different concurrency level of 1 and 2.

Calculations:

For Memory Benchmarking, the system will perform 2 different read write operations on the memory for different block sizes at different concurrency levels while accessing the memory sequentially and randomly.

$$\text{Throughput} = \frac{\text{No_of_times operations performed} * (\text{No of Reads/Writes}) * \text{Num_Bytes}}{\text{Time_Elapsed} * 1024 * 1024} \quad \text{Mb/Sec}$$

No_of_operations: This is the total number of times the file is read or written to / from the disk.

No of Read/Writes: The Benchmarking is done using memory operations like memmove(Read and Write) and memcpy(Read and Write).

Num_Bytes: The Bytes Read or written for each operation.

1B = 100000000 times (100Mb)

1Kb = 100000 times (100Mb)

1Mb = 100 times (100Mb)

Time Elapsed: The time taken to read or write 1B, 1Kb or 1Mb of memory.

Latency: 1/Throughput

Benchmarking Results:

1 Thread

	1B	1Kb	1Mb
Sequential	305.828522	4909.44629	5327.79297
	320.768677	5693.5542	5582.09717
	322.572968	5695.57324	5672.60449
Average	316.390056	5432.85791	5527.49821

	1B of your memory	1Kb	1Mb
Random	14.270962	3356.45532	5118.37549
	14.557113	3504.68897	5095.80225
	14.770414	3429.17896	5033.09424
Average	14.5328297	3430.10775	5082.42399

2 Thread

	1B	1Kb	1Mb
Sequential	86.274536	2835.85596	2787.92383
	162.887741	2704.50244	2621.56055
	161.612534	2813.43018	2551.84351
Average	136.924937	2784.59619	2653.77596

	1B	1Kb	1Mb
Random	7.334429	1766.55347	2570.69824
	7.31721	1708.13989	2578.58594
	7.331349	1666.70789	2484.93359
Average	7.32766267	1713.80042	2544.73926

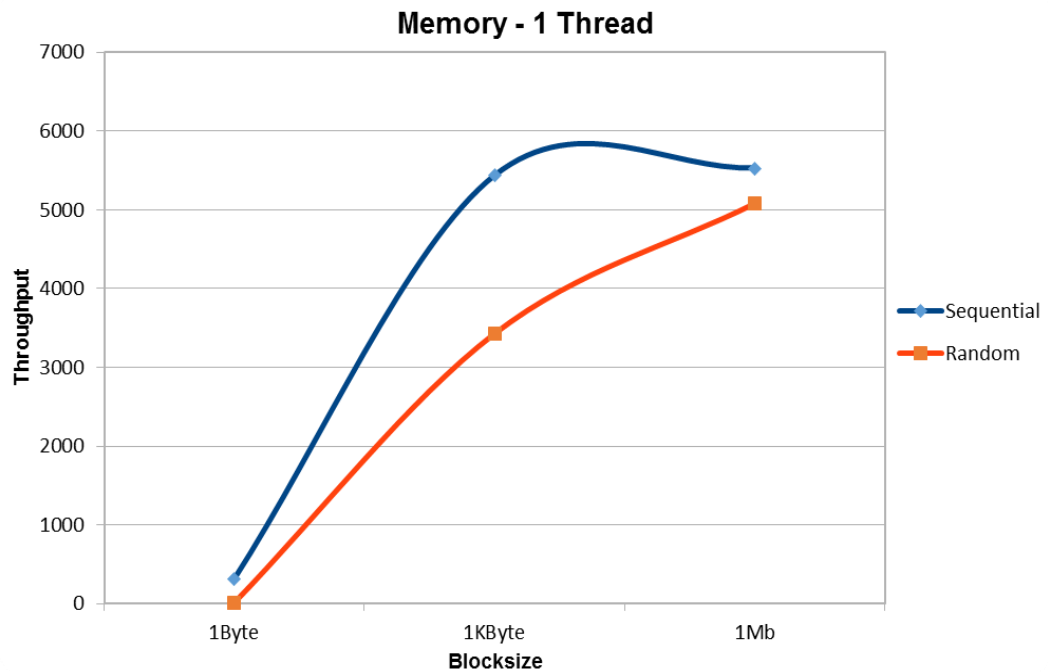


Fig. Sequential/ Random access for Memory 1 Thread for varying Block Size.

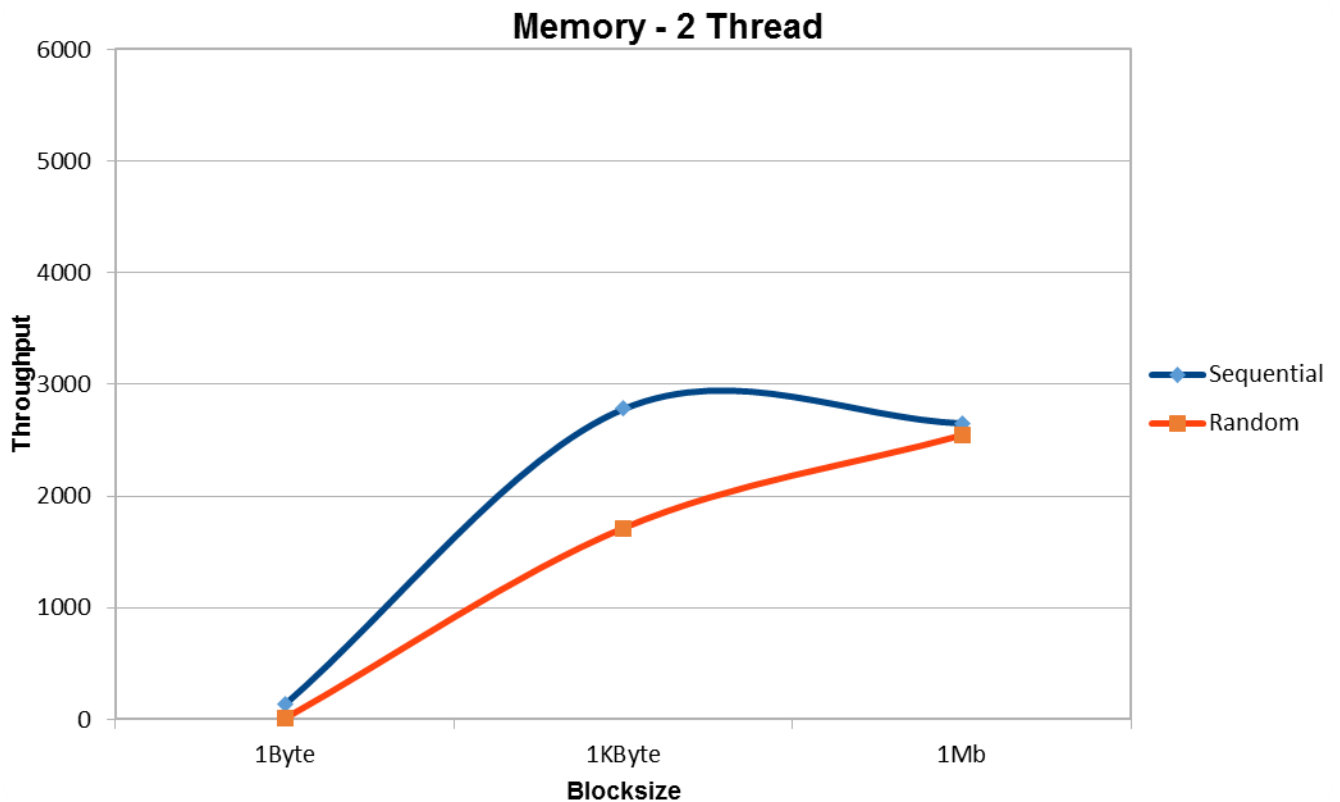


Fig. Sequential/ Random access for Memory 2 Thread for varying Block Size.

Theoretical Peak Performance:

The Theoretical is calculated using the references for finding clock speed and bitrate for DDR3_SDRAM. of your memory

System Configuration on T2.Mirco Instance:

Architecture: x86_64

Clock Speed: 2394.926 MHz

The theoretical peak performance can be calculated using the below formula

$$\text{Throughput} = \frac{\text{Bitrate (Bits per clock)} * \text{Clock Speed}}{8} \quad \text{Mb/Sec}$$

$$= 19159.408 \text{ Mb/Sec}$$

Efficiency:

The Efficiency of the performance for benchmarking on this system can be calculated by comparing the results with the theoretical peak performance obtained.

$$\text{Efficiency} = \frac{\text{Actual Performance}}{\text{Theoretical Peak Performance}} * 100 \quad \text{Mb/sec}$$

$$= (5527.49821 / 19159.408) * 100$$

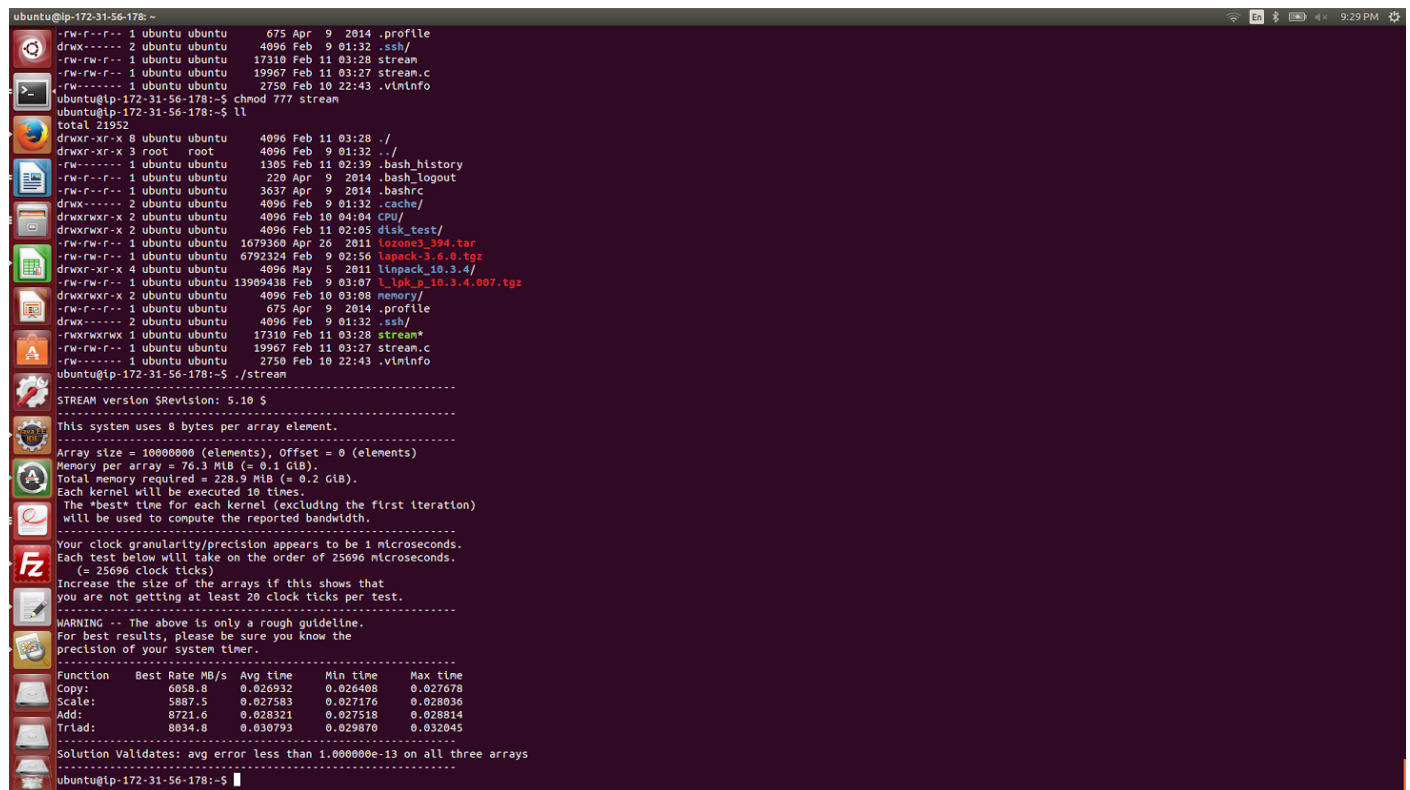
$$= \mathbf{28.85 \%}$$

Conclusion:

- The optimal performance achieved is for concurrency level 1. The average throughput and Latency shown in the results and graphs shows the optimal performance at concurrency level 1.
- Here again the sequential access is faster than the random access because the overhead of accessing any random memory and seeking the file pointer to that random location is avoided.

Extra Credits:

Stream Benchmark is used to find best performance of the Memory. The results of stream benchmark are mentioned in the below screenshot,



```
ubuntu@ip-172-31-56-178: ~$ dd if=/dev/zero of=/dev/null bs=1M count=21952
21952+0 records in
21952+0 records out
21952 KiB transferred by 1048576 KiB in 3m40s

ubuntu@ip-172-31-56-178: ~$ ./stream
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 = 228.0 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 25096 microseconds.
(= 25096 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: 6058.8 0.026932 0.026408 0.027678
Scale: 5887.5 0.027583 0.027176 0.028036
Add: 8721.6 0.028321 0.027518 0.028814
Triad: 8034.8 0.030793 0.029870 0.032045
-----
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-----
ubuntu@ip-172-31-56-178: ~$
```

Efficiency:

Throughput in Stream Benchmark (Mb/sec) = 6058.80

Result Throughput (Mb/Sec) = 5527.49821

$$\begin{aligned} \text{Efficiency} &= \frac{\text{Actual Performance}}{\text{Theoretical Peak Performance}} * 100 \text{ Mb/sec} \\ &= (5527.4982 / 6058.80) * 100 \\ &= 91.23\% \end{aligned}$$