REPORT

PERFORMANCE BENCHMARKING

The experiment performed is in order to benchmark the system resources. We use the virtual machine or instances provided by Amazon Web Services to obtain a standard value. This is done because the configuration of each machine differs and accordingly the values may also differ. Once the experiments are done, we run the respective standard benchmark programs to compare the values and improvise the code provided by us to the benchmark program.

CPU:

The goal of this experiment is to calculate the processor speed in GFLOPS (Giga Floating Point Operations per Second) for floating point operations or GIOPS (Giga Integer Operations per Second) for integer operations.

Design:

The approach followed here is to use the primitive operations of multiplication, addition, subtraction and division in arithmetic expressions with integer and floating point numbers. These operations are executed in a loop several times to give the CPU enough load to approximate the number of instructions per second.

The results vary based on the overall workload of the CPU and on the fact if it is a dedicated or shared resource.

The values of FLOPS and IOPS are approximated to the nearest integer for calcualations.

Observations:

The t2.micro instance is a shared resource i.e. the user is not given the complete core for utilization. Hence the number of GFLOPS/GIOPS obtained differs based on availability of the system.

Theoretical value for local system = (CPU speed in GHz) x (number of CPU cores) x (CPU instruction per cycle)

= 2.88 * 2 * 4 (assuming value comparing the configuration matching another system)

= 23 (approx.)

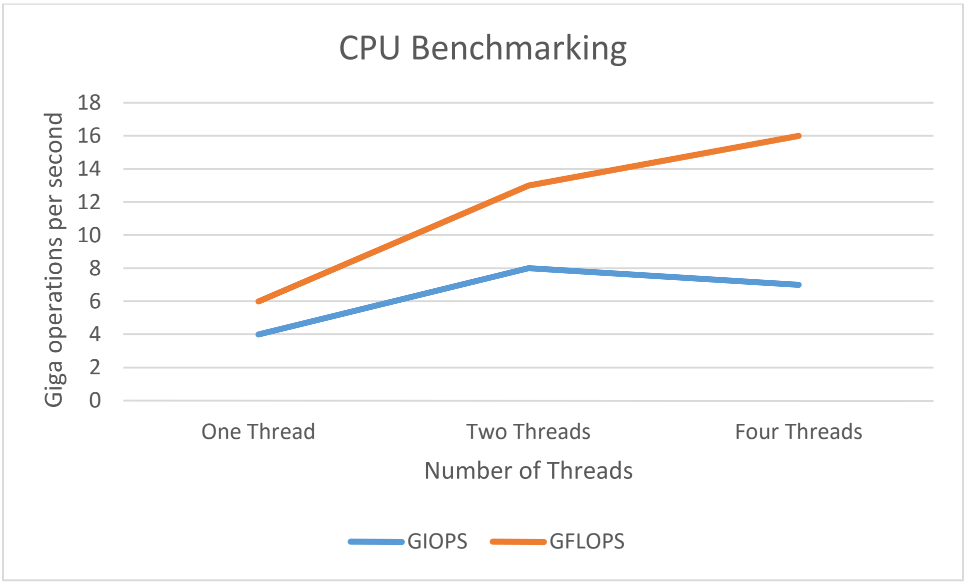
Theoretical value for t2.micro = (CPU speed in GHz) x (number of CPU cores) x (CPU instruction per cycle)

= 2.4 * 1 * 8 (assuming value comparing the configuration matching another system)

= 19(approx.)

Result:

The values obtained in t2.micro instance are:



Case	GIOPS	GFLOPS
Single Thread	4	2
Two Threads	8	5
Four Threads	7	9

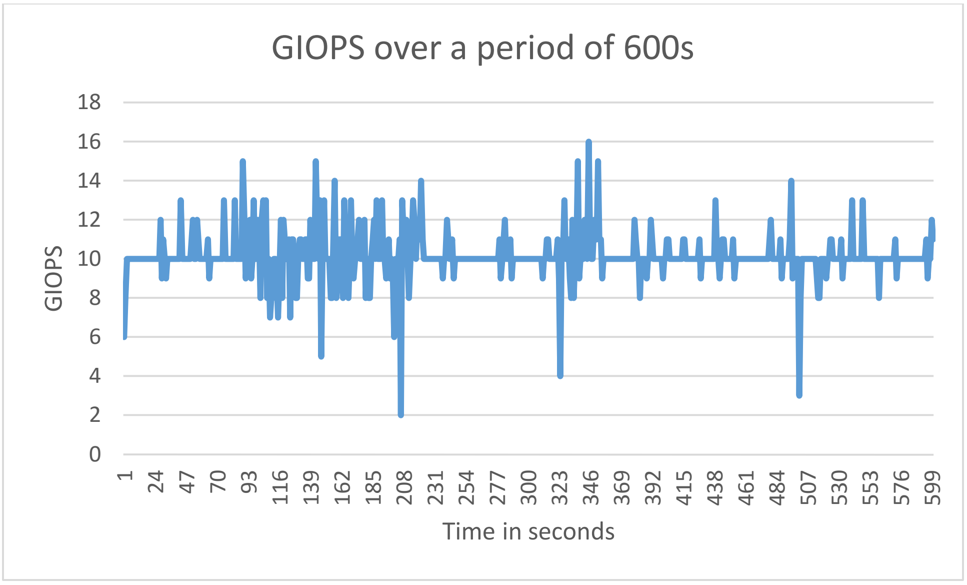
Performance:

The values here are comparatively less due to the fact that there is a lot of resource contention in the t2.micro instance and we will not be given a dedicated resource to fully utilize it. We are able to achieve about 25% – 30% of the benchmark values.

600 seconds Sample run:

Case 1: Integer Operations

The results for running the experiment for a period of 10 minutes and noting down the readings every second are as follows:

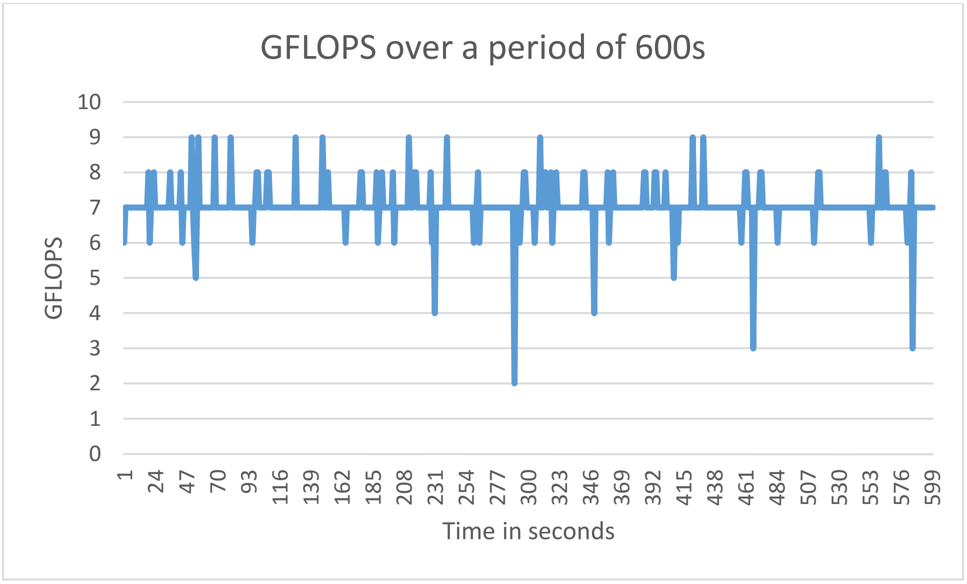


Performance:

From the graph we conclude that it varies based on the availability of the core in the micro instance to perform the task. Since the value considered is of four threads running parallel, we see peaks in the value of GIOPS when all of them are able to utilize the CPU efficiently.

Case 2: Floating point operations

The results for running the experiment for a period of 10 minutes and noting down the readings every second are as follows:



Performance:

The value of GFLOPS for four threads run in parallel varies based on the resource available to the program at any given instance of time. The efficiency reached is about 30% of the value obtained in standard benchmarks where it peaks (9 GFLOPS)

DISK:

The goal of this experiment is to find the throughput (MB/s) and latency (ms) for the disk by performing experiments involving read and write, in random and sequential order for blocks of size 1 byte, 1 KB and 1 MB.

Design:

The approach followed for this experiment is to create a file of size greater than L1 cache (say > 7 MB) and store it in disk. Then the program needs to read, write to this file on disk in units of 1 byte, 1 kilobyte and 1 megabyte. Random and sequential operations are done and each operation is repeated in a loop in order to achieve precision in the calculations.

Observations and Performance:

Since the t2.micro instance does not allocate dedicated disk space to the user, it is possible to achieve only a part of the theoretical value or value as per the benchmark.

The latency for sequential operations is higher than that of random operations as we need to seek to particular location and then traverse continuously which takes more time.

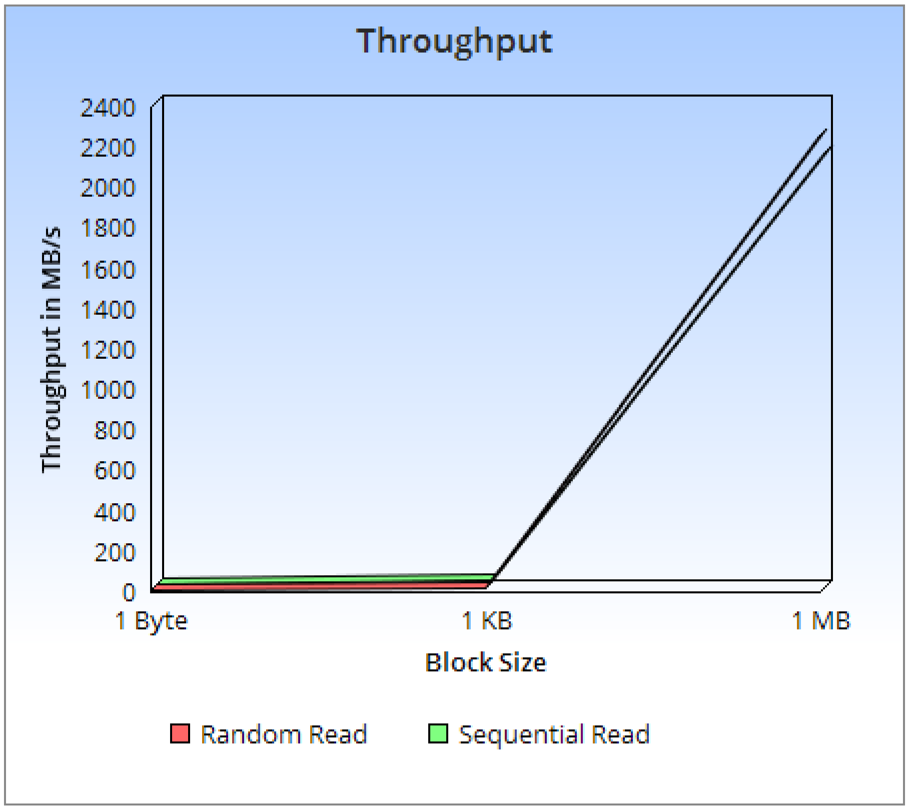
The throughput for reading 1 MB is higher than expected on the local machine, due to volume of data and configuration available to the user at the time of the operation.

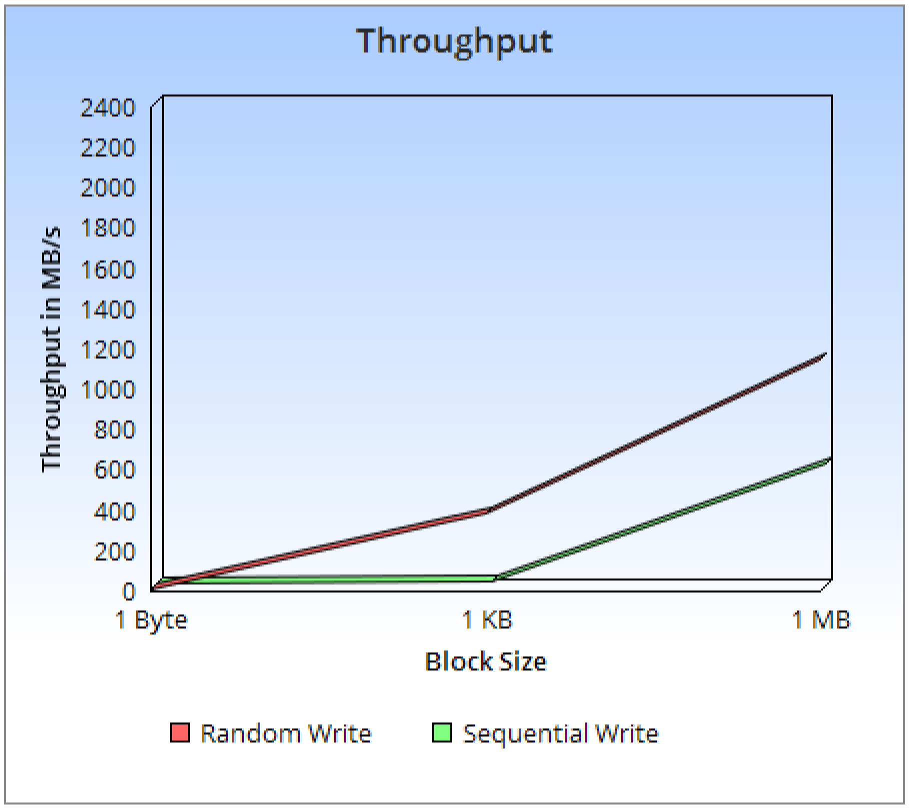
Theoretical disk bandwidth for local system (HP EliteBook 8470p) = 300MB/s (as given by the provider)

Result:

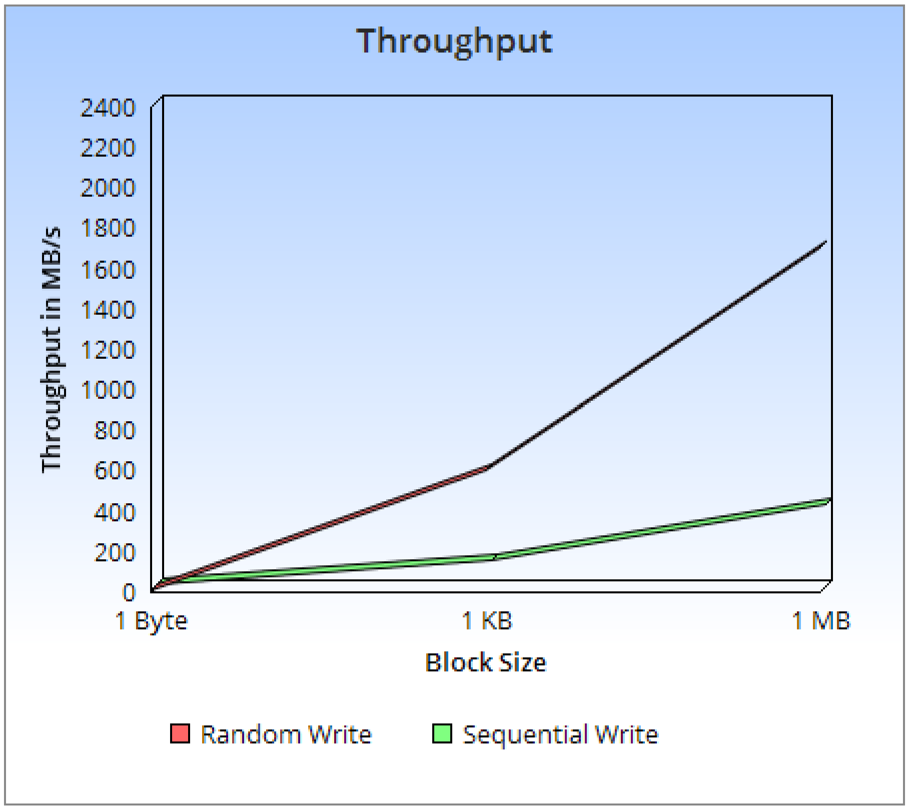
The following are the results obtained by running on t2.micro instance:

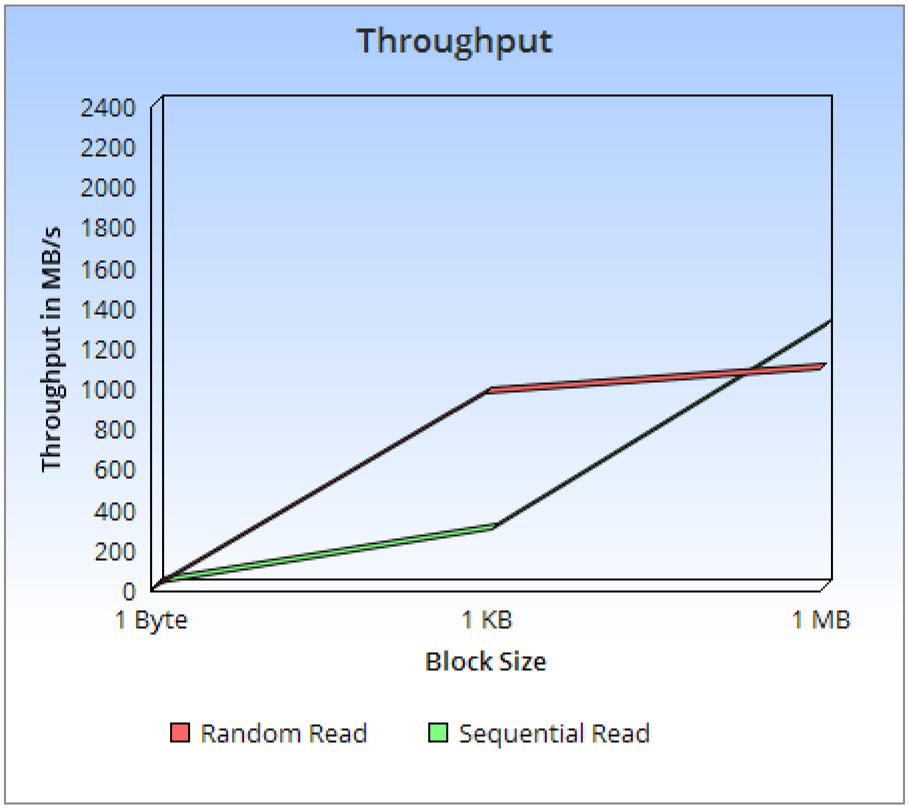
Single Thread:




Throughput
Throughput in MB/s
2400
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
0
1 Byte
1 KB
1 MB
Block Size
🟥 Random Write
🟩 Sequential Write

Two Threads:

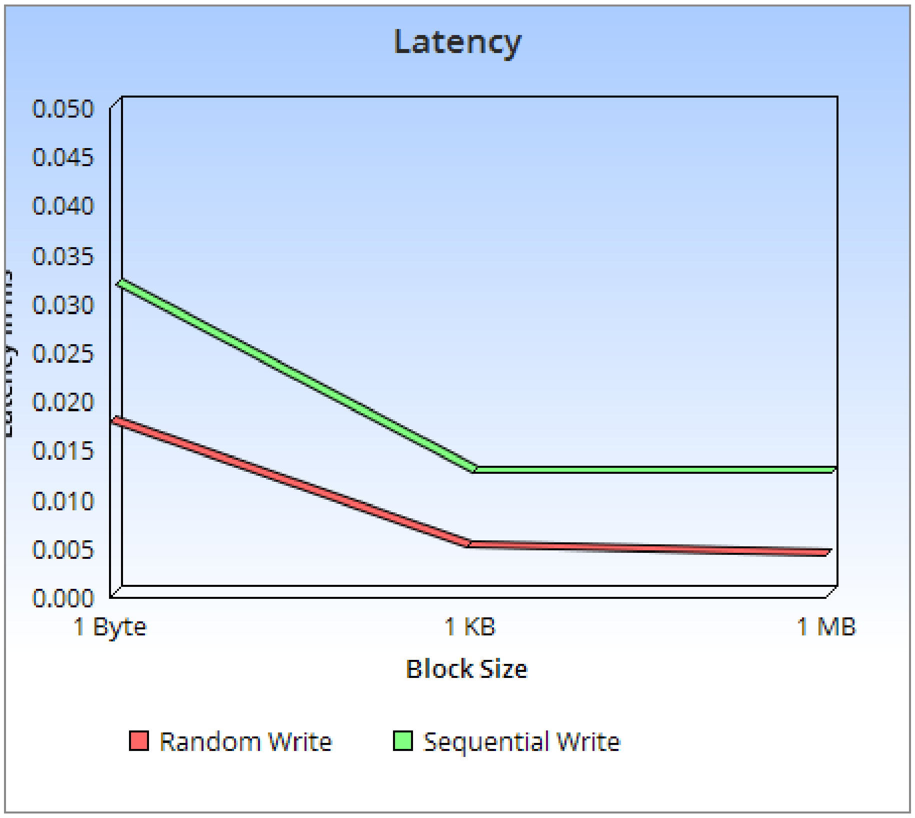

Throughput
Throughput in MB/s
2400
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
0
1 Byte
1 KB
1 MB
Block Size
🟥 Random Write
🟩 Sequential Write


Throughput
Throughput in MB/s
2400
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
0
1 Byte
1 KB
1 MB
Block Size
🟥 Random Read
🟩 Sequential Read

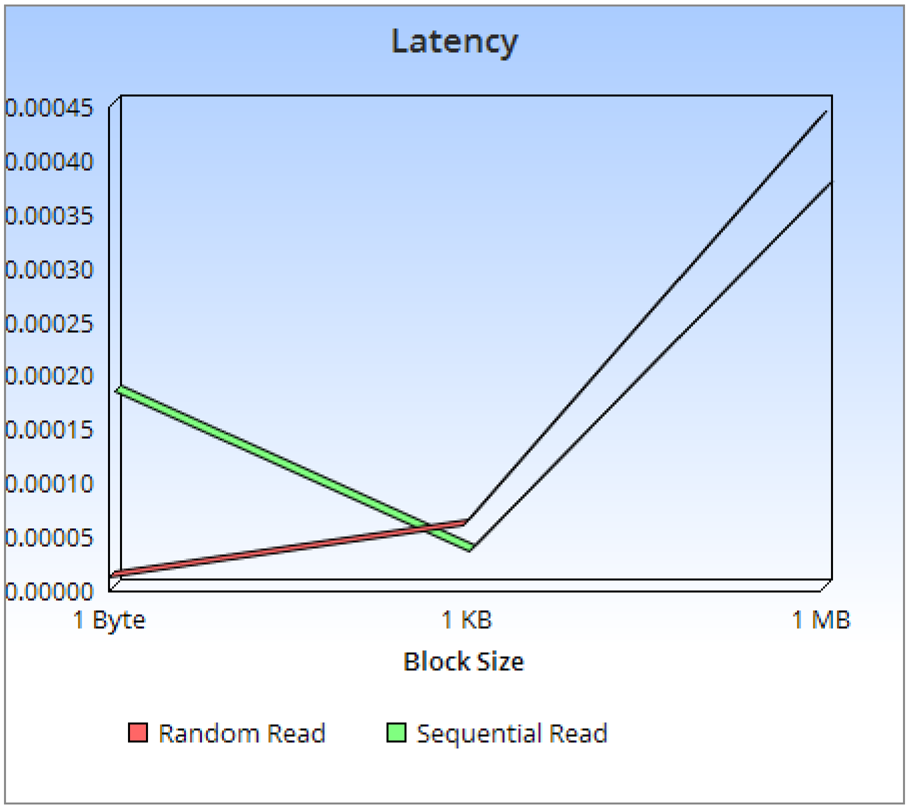
Case	Throughput_random (MB/s)			Throughput_ sequential(MB/s)		
	1b	1KB	1MB	1b	1KB	1MB
Single Thread	0.0839 – read 0.0855 - write	15.211 -read 381.91 - write	2256.20 – read 1147.62 - write	0.00502 – read 0.00003 - write	25.91 - read 15.01 - write	2146.34- read 598.63- write
Two Threads	0.178 – read 0.09 - write	973.72- read 604.14- write	1096.92-read 1700.19- write	0.0198 - read 0.0178 - write	272.41- read 126.77- write	1284.78 - read 400.44 -write

Single Thread:

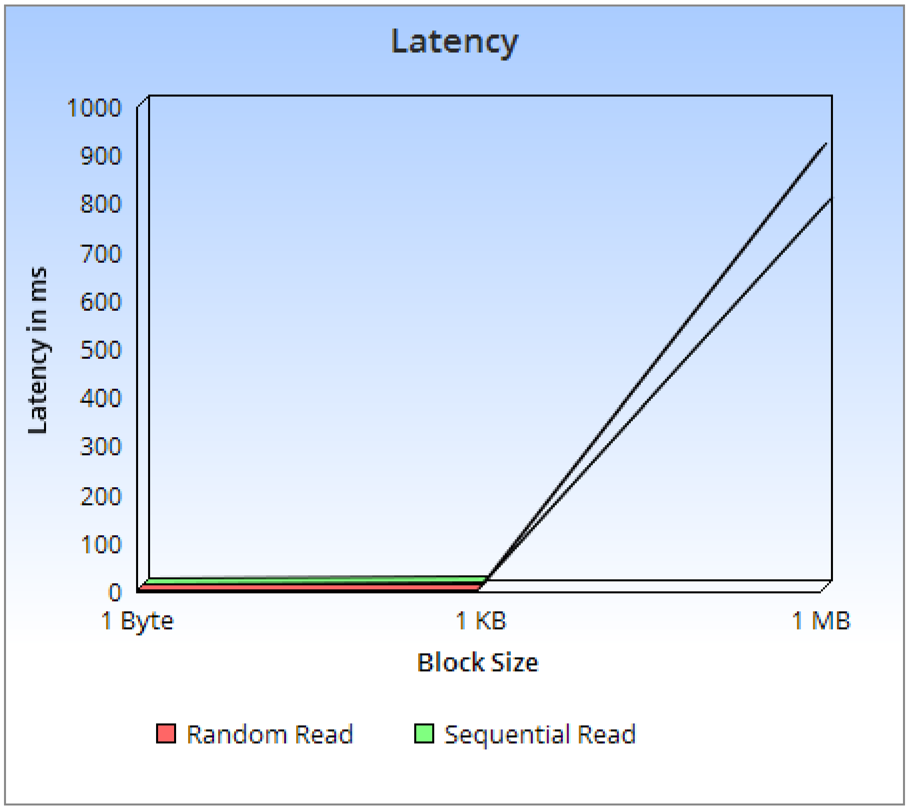
Latency in ms

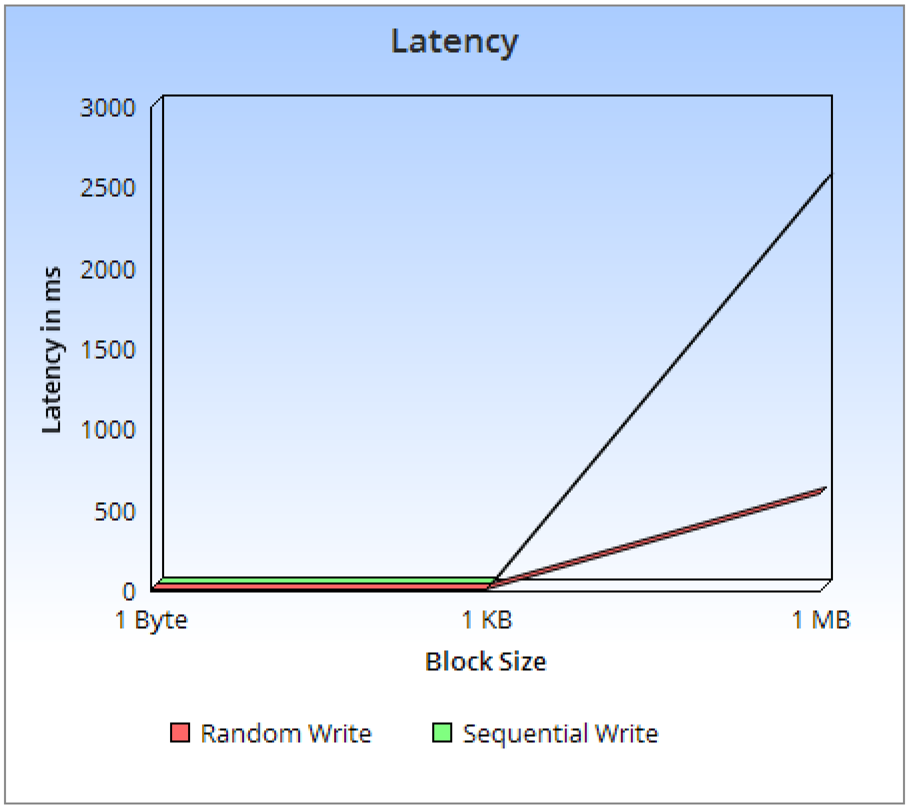


Latency in ms



Two Threads:


Latency
Latency in ms
1000
900
800
700
600
500
400
300
200
100
0
1 Byte
1 KB
1 MB
Block Size
Random Read
Sequential Read


Latency
Latency in ms
3000
2500
2000
1500
1000
500
0
1 Byte
1 KB
1 MB
Block Size
Random Write
Sequential Write

Case	Latency_random (ms)			Latency_sequential(ms)		
	1b	1KB	1MB	1b	1KB	1MB
Single Thread	0.01796- write 0.000013– read	0. 005144– write 0. 00006– read	0. 0042 – write 0.00044 - read	0.0312- write 0.00018 – read	0.012- write 0.00003 - read	0.0120- write 0.00037 - read
Two Threads	0.2424- read 0.0038 - write	1.012- read 1.625 – write	911.87 – read 599.19 - write	0.122 – read 0.026 – write	6.55 – read 8.38- write	788.56-read 2507.95 - write

MEMORY:

The goal of this experiment is to find the throughput (MB/s) and latency (ms) of memory using random and sequential read and write operations. To implement this a fixed chunk of memory is read and written into using built in functions.

Design:

We use the function memcpy API in C library to do the read and write operations. The code is executed for 3 different block sizes namely 1 byte,1 KB and 1 MB. We do the operations sequentially and by seeking into random locations.

The same operation is repeated in a loop several times in order to get an accurate result.

Observations and Performance:

The latency for sequential operations is much higher than that for random operations. Throughput is thus higher for random operations as compared to sequential operations. Due to the resource sharing constraints in t2.micro it is possible to obtain values which are about 30% - 40% of that in the benchmarks.

The value of throughput obtained on the benchmark is around 11,000 – 12,000 MB/s for 64 KB of data. So with the constraints

Theoretical value of the memory bandwidth for local system (HP Elitebook 8470p) = Base DRAM clock frequency * Number of data transfers per clock * Memory bus (interface) width * Number of interfaces

The configuration of local system memory is 1600 MHz, DDR3 SDRAM.

Hence:

Base DRAM clock frequency= 1600 * 1000000 clocks per second

Number of data transfers per clock: Two, in the case of "double data rate" (DDR, DDR2, DDR3, DDR4) memory.

Memory bus (interface) width: Each DDR, DDR2, or DDR3 memory interface is 64 bits wide.

Number of interfaces: Modern personal computers typically use two memory interfaces (dual-channel mode).

So memory bandwidth = (1600000000 * 2 * 64 * 2) / 8

= 51200000000 bytes per second

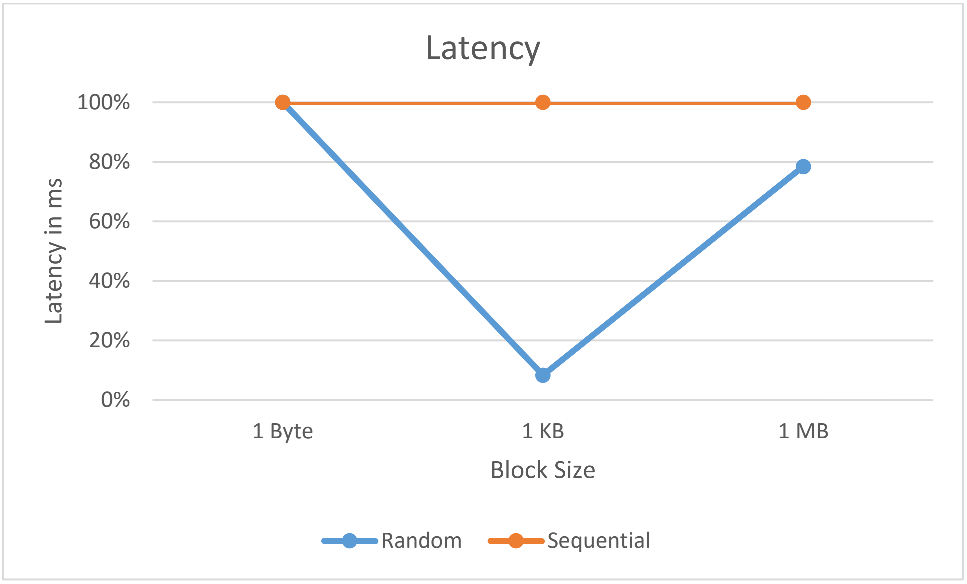
= 51200 MB/s

Result:

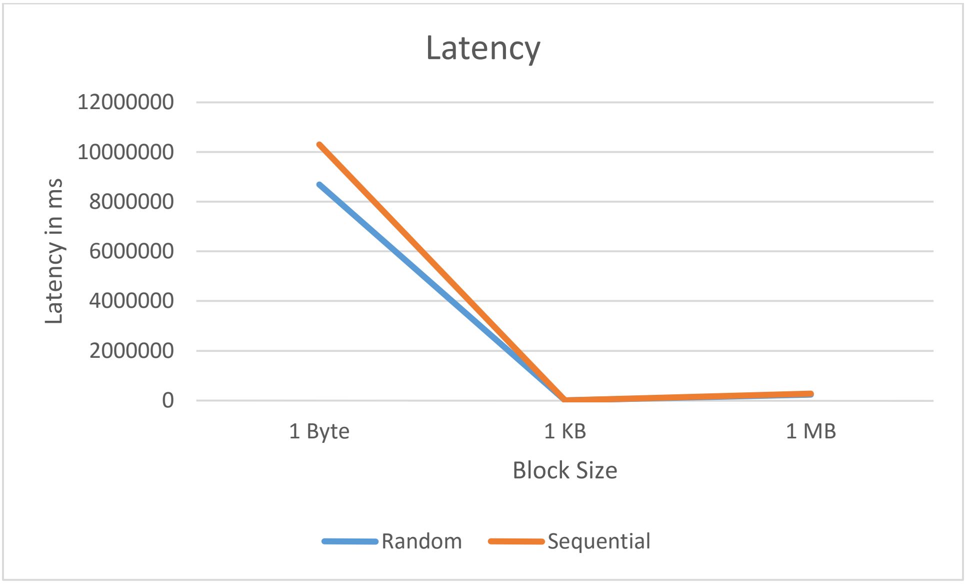
The following are the results obtained by running on t2.micro instance:

Latency:

Single Thread:



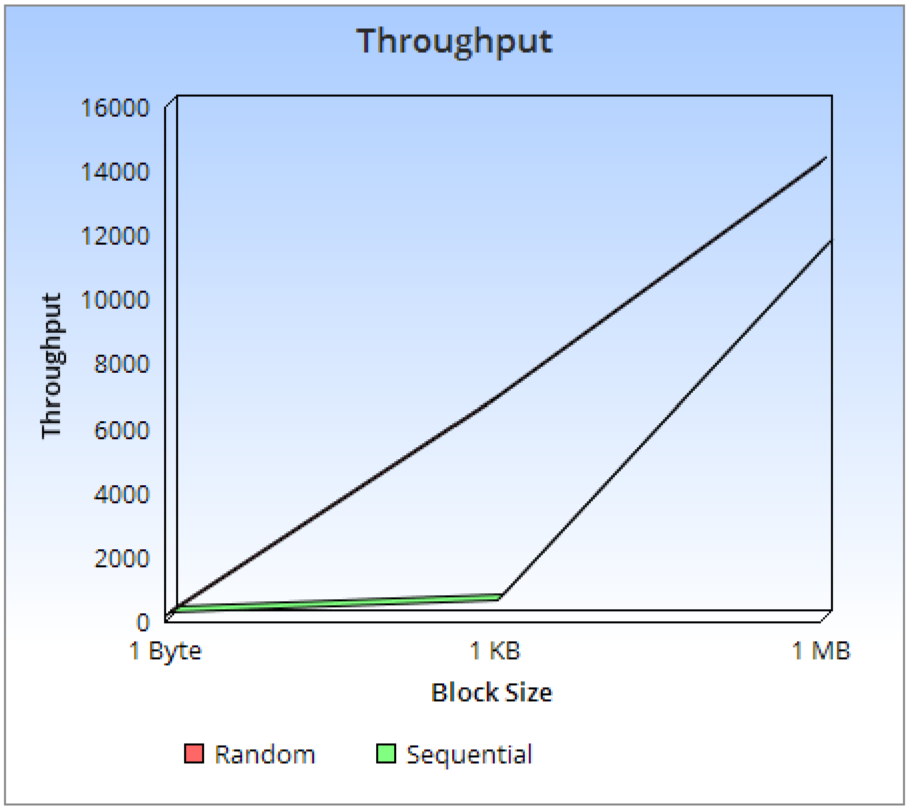
Two Threads:



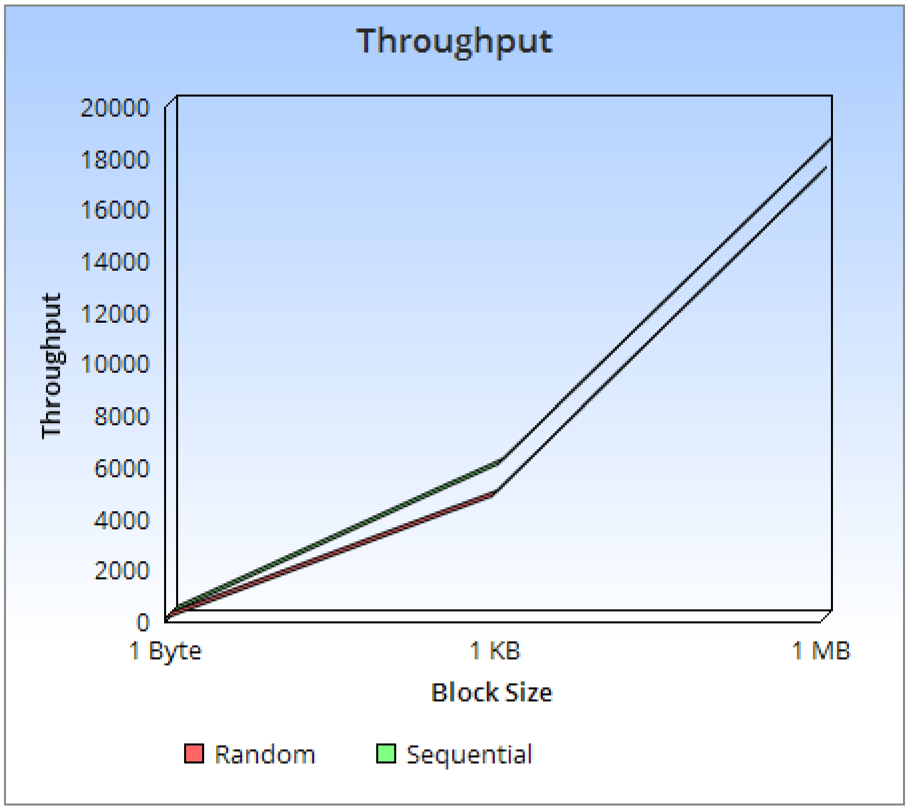
Case	Latency_random(ms)			Latency_sequential(ms)		
	1b	1KB	1 MB	1b	1KB	1MB
Single Thread	85348000	1000	632000	0.008583	11000	174000
Two Thread	8696500	1000	230500	10310500	1000	273000

Throughput:

Single Thread:



Two Threads:



Case	Throughput_random(MB/s)			Throughput_sequential(MB/s)		
	1b	1KB	1MB	1b	1KB	1MB
Single Thread	175.75	6835.93	14240.50	111.11	443.89	11494
Two Thread	115.068229	4882.812500	17404.764602	100.681216	5859.375000	18331.821084

References

ChartGo.com,. "Chartgo Create Graphs Online.". N.p., 2016. Web. 13 Feb. 2016.

Stackoverflow.com,. "Stack Overflow". N.p., 2016. Web. 11 Feb. 2016.

Tom's Hardware,. "Tom's Hardware: Hardware News, Tests And Reviews". N.p., 2016. Web. 11 Feb. 2016.

Wikipedia,. "Wikipedia, The Free Encyclopedia". N.p., 2016. Web. 11 Feb. 2016.