1.Design Documentation

This programming assignment consists of 3 parts,
1. CPU Benchmark
2. Memory Benchmark
3. Disk Benchmark

The basic designs of all the benchmarks are as follows:

(a) Functions are defined that does all the operations like IOPS, FLOPS, Read and write.
(b) The Main Block usually has two options. The 1st one is to input the kind of operation to be done and the second input is to know the number of threads to be executed. The functions defined are called according to this inputs.
(c) Before starting the thread the time is noted using wall clock function. The time is noted after the execution and the difference is the time of execution and this is used for further calculation.

The detailed design description of all benchmarks:

1. **CPU Benchmark:** This has 4 separate Programs. The first function is to compute the GFLOPS and this is done by doing multiple float division operations of numbers stored in an array. The second program is to calculate GIOPS and this is done by doing multiple integer multiplication operations of values stored in an array. The third program is to do integer operations for 10 minutes and produce 600 IOPS samples. A sample is generated every second. The fourth and the last program is to do floating operations for 10 minutes and produce 500 sample FLOPS samples. A sample is generated every second.
2. **Memory Benchmark:** The program consists of 6 functions to do Sequential and Random memory access using 1B,1KB and 1MB block sizes using memcpy () function. The main program is a menu driven program and user can input the kind of operation to be done. He can also select the number of threads. This two inputs are used to decide the number of threads and the function to be called.

3. **Disk Benchmark**: The program consists of 12 functions to do sequential and Random memory read and write operation using 1B,1KB and 1MB block sizes. The functions use fread (), fwrite () and fseek () to do this. The main program is a menu driven program and user can input the kind of operation to be done. He can also select the number of threads. These two inputs are used to decide number of threads and the function to be called

Improvements and Extensions to program:

The following improvements and extensions can be implemented

1. CPU benchmark:
i. Simple operations are used. Much more complicated operations can be used
ii. More than 4 threads can be used
iii. GUI can be implemented
iv. The functions should be done for more time and there should be graphs to show the trends of the GFLOPS and IOPS as the time progresses

2. Memory Benchmark:

i. More threads can be used.

ii. GUI can be implemented

iii. memcpy () function was used. More functions could be tried out and check which one has better output

3. Disk Benchmark:

i. More threads can be used

ii. The program was implemented in a virtual machine. So a better estimate can be obtained if used in a standalone OS

iii. Better functions than fseek (), fwrite () and fread can be used

2.Performance Evaluation

CPU Benchmark:

- The program finds the number of GIGA FLOPS and GIGA IOPS per second
- The Start time and end time is calculated
- The total time is time difference between start time and end time.
- The performance is measured using the formula

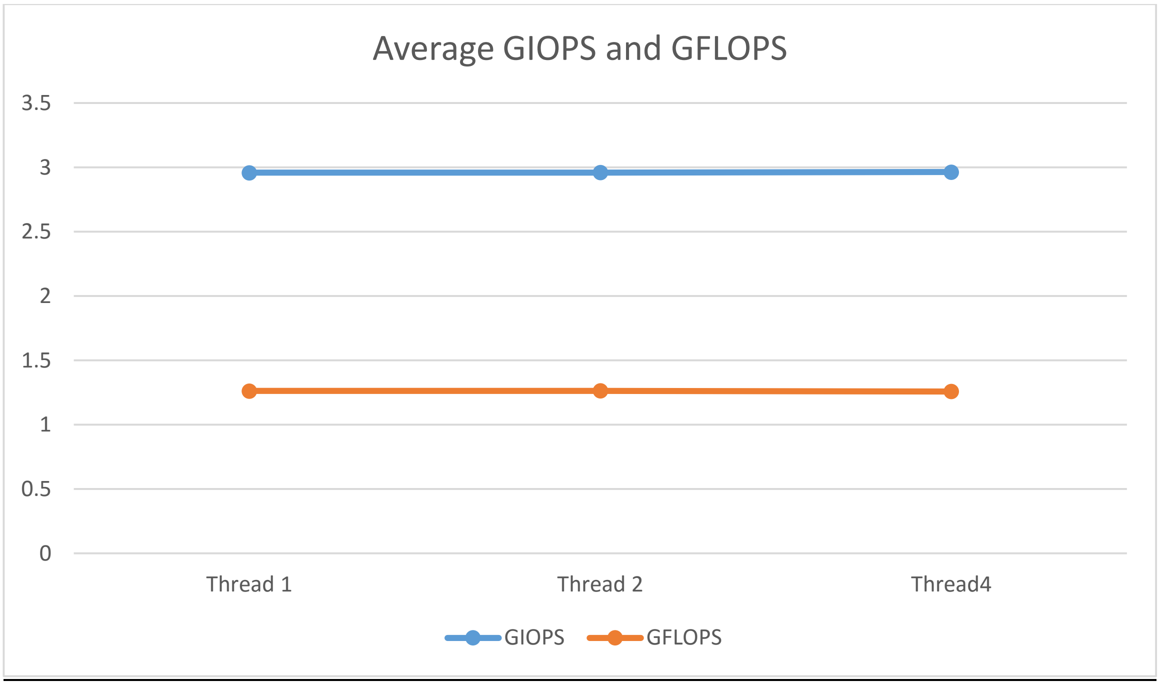
OPS = number of instructions / Total Time .

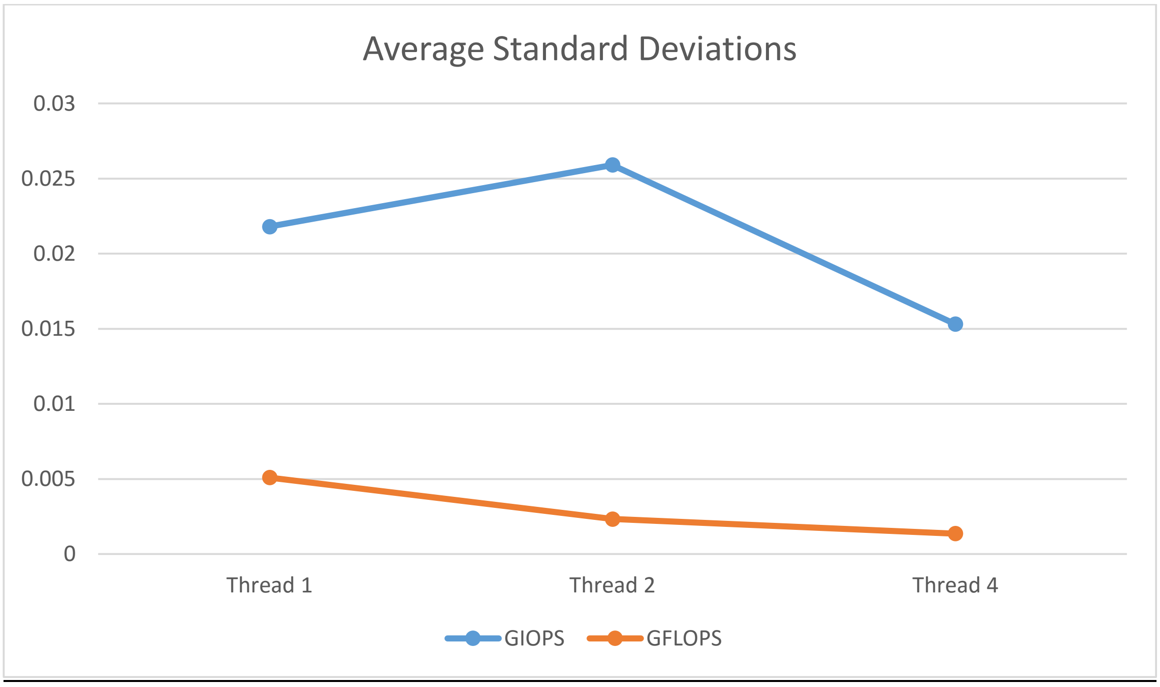
Experiment 1:

The program to calculate GIOPS and GFLOPS were run 3 times and the outputs were recorded in the table. The

Performance Table:

Operations per second	1 Thread	2 Thread	4 Thread
GFLOPS/sec exp1	1.26342	1.26723	1.25757
GFLOPS/sec exp2	1.25786	1.26428	1.2587
GFLOPS/sec exp3	1.26803	1.26263	1.26029
GFLOPS/sec Average	1.263103	1.264713	1.258853
GFLOPS/sec SD	0.005092	0.00233	0.001366
GIOPS/sec exp1	2.94118	2.9504	2.97453
GIOPS/sec exp2	2.9553	2.99065	2.97204
GIOPS/sec exp3	2.98396	2.94226	2.94686
GIOPS/sec Average	2.960147	2.961103	2.964477
GIOPS/sec SD	0.021798	0.02591	0.015307





Operations are taken on the y axis and thread is taken on the x axis

Observation:

- There is always increase in operations when the 2 threads are used
- In case of GIOPS we see a constant linear growth as the number of threads increases

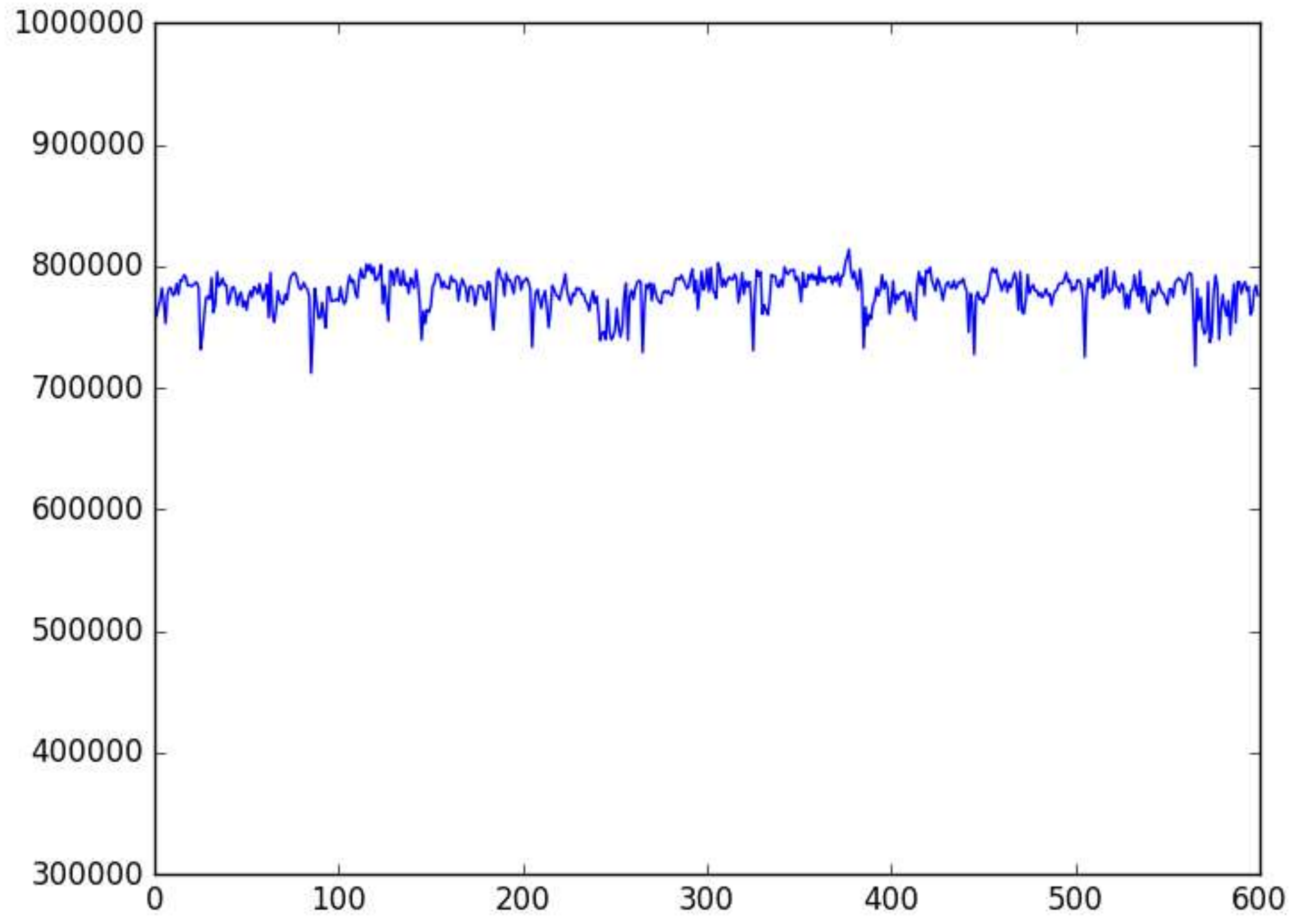
Conclusions:

- The number of IOPS and FLOPS increases with the number of threads
- The formulated values of GFLOPS and IOPS mentioned above are much less when compared to the values obtained while running Linpack.
- The algorithm used was not efficient as lot of processing is wasted on iterations and assignment operations

Experiment 2:

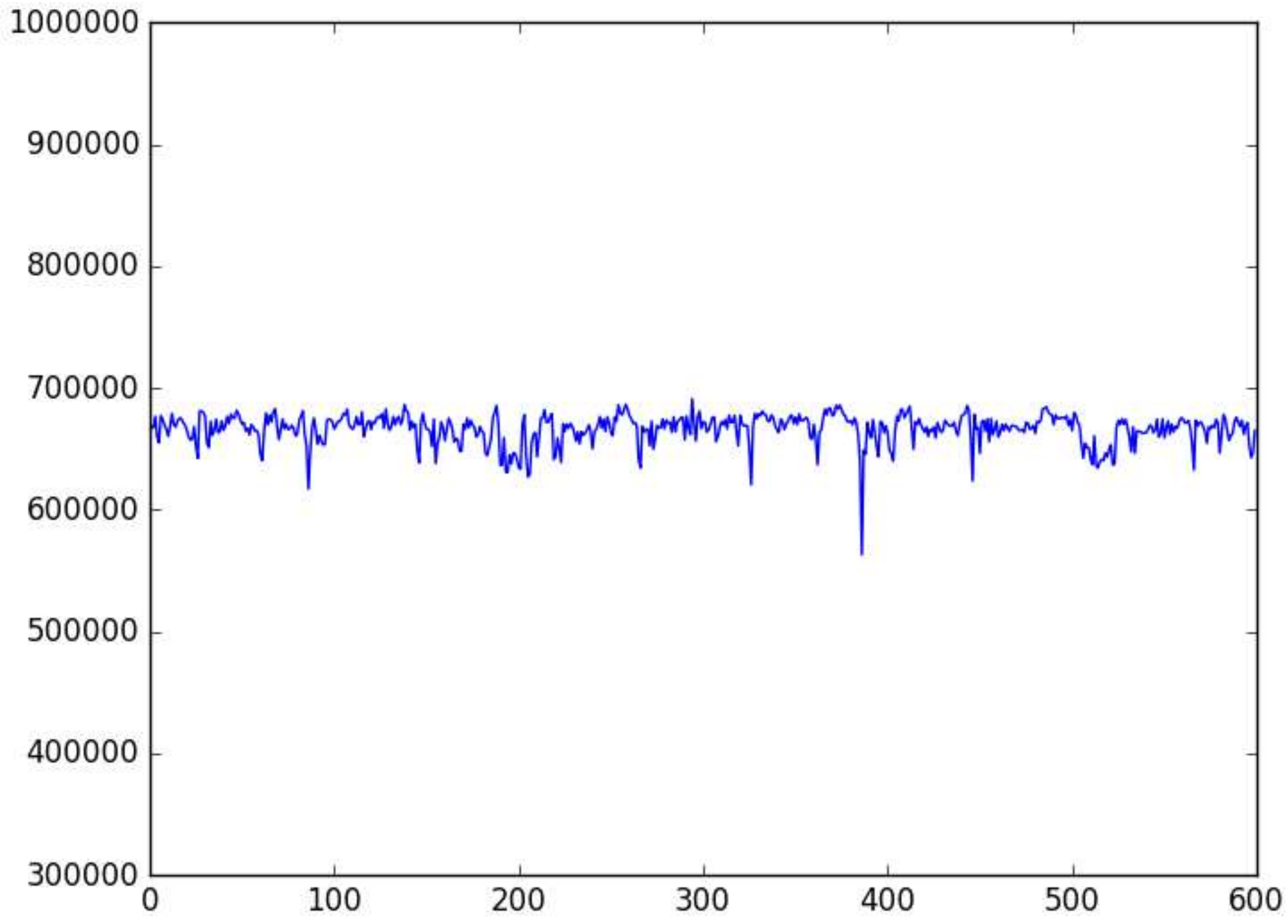
A program was run for 600 seconds and sample iops and flops were collected every second. This data was used to draw a graph. The graph is as follows

FLOPS



Seconds is taken in the x axis and Flops is taken in the y axis

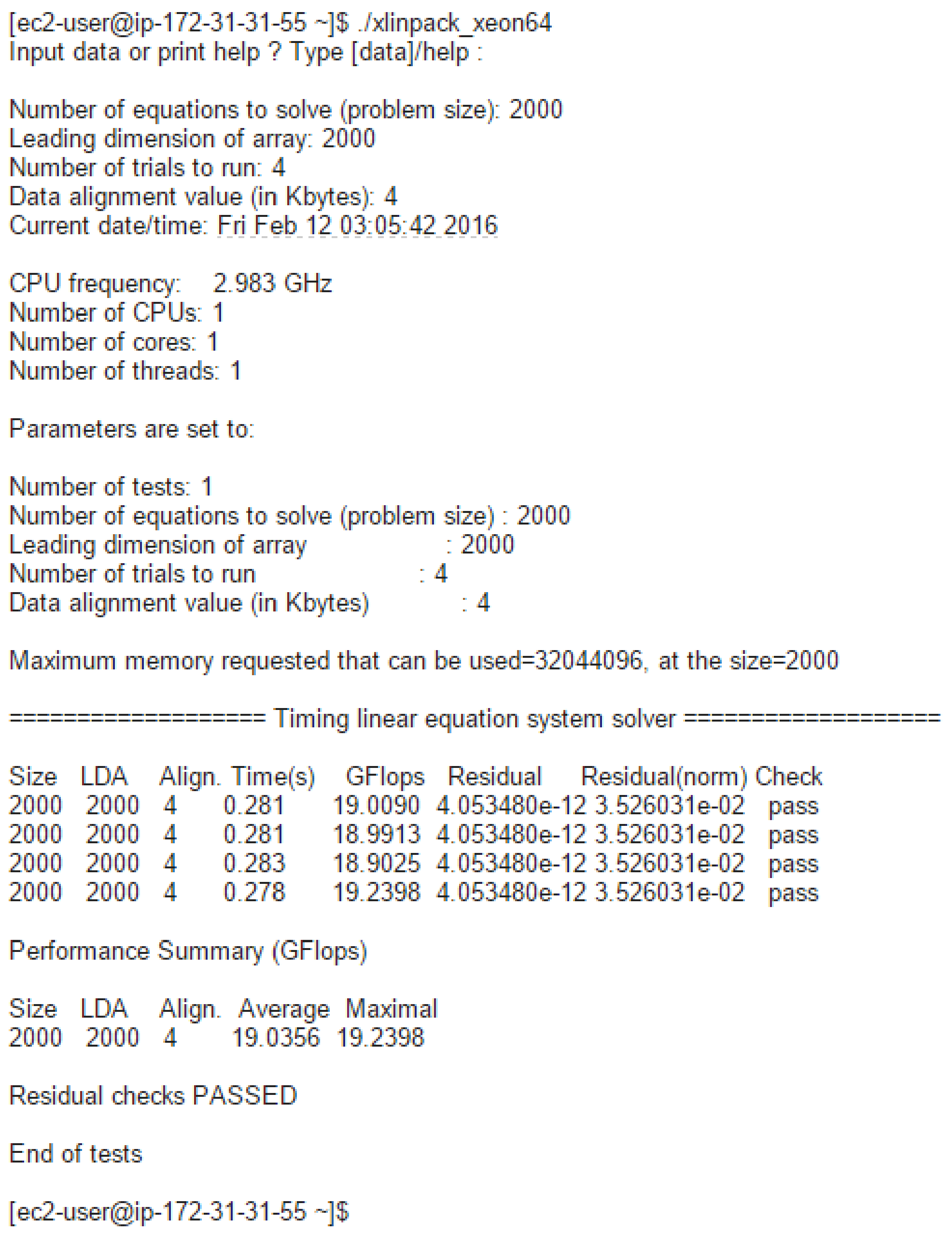
IOPS



Seconds is taken in the x axis and Iops is taken in the y axis

Experiment 3:

Linpack was run and the following results were obtained



```
[ec2-user@ip-172-31-31-55 ~]$ ./xlinpack_xeon64
Input data or print help ? Type [data]/help :

Number of equations to solve (problem size): 2000
Leading dimension of array: 2000
Number of trials to run: 4
Data alignment value (in Kbytes): 4
Current date/time: Fri Feb 12 03:05:42 2016

CPU frequency:    2.983 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) : 2000
Leading dimension of array                  : 2000
Number of trials to run                     : 4
Data alignment value (in Kbytes)            : 4

Maximum memory requested that can be used=32044096, at the size=2000

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

Size   LDA    Align. Time(s)    GFlops   Residual      Residual(norm) Check
2000   2000   4      0.281      19.0090  4.053480e-12 3.526031e-02   pass
2000   2000   4      0.281      18.9913  4.053480e-12 3.526031e-02   pass
2000   2000   4      0.283      18.9025  4.053480e-12 3.526031e-02   pass
2000   2000   4      0.278      19.2398  4.053480e-12 3.526031e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
2000   2000   4       19.0356  19.2398

Residual checks PASSED

End of tests

[ec2-user@ip-172-31-31-55 ~]$
```

Theoretical Peak Performance:

The Amazon instance has a processor: Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz

The closest to it is Intel(R) Xeon(R) CPU E5-2670 and its performance is:

Processor Number	Frequency Type	Clock	CTP	GFLOP	APP 1-way	APP 2-way	APP 4-way
E5-2670	Base	2.6	204533	166.4	0.04992	0.09984	0.19968
	Single Core Max Turbo	3.3	259600	211	0.06336	0.12672	0.25344
	GPU ONLY	N/A	N/A	N/A	N/A	N/A	N/A

Performance Table:

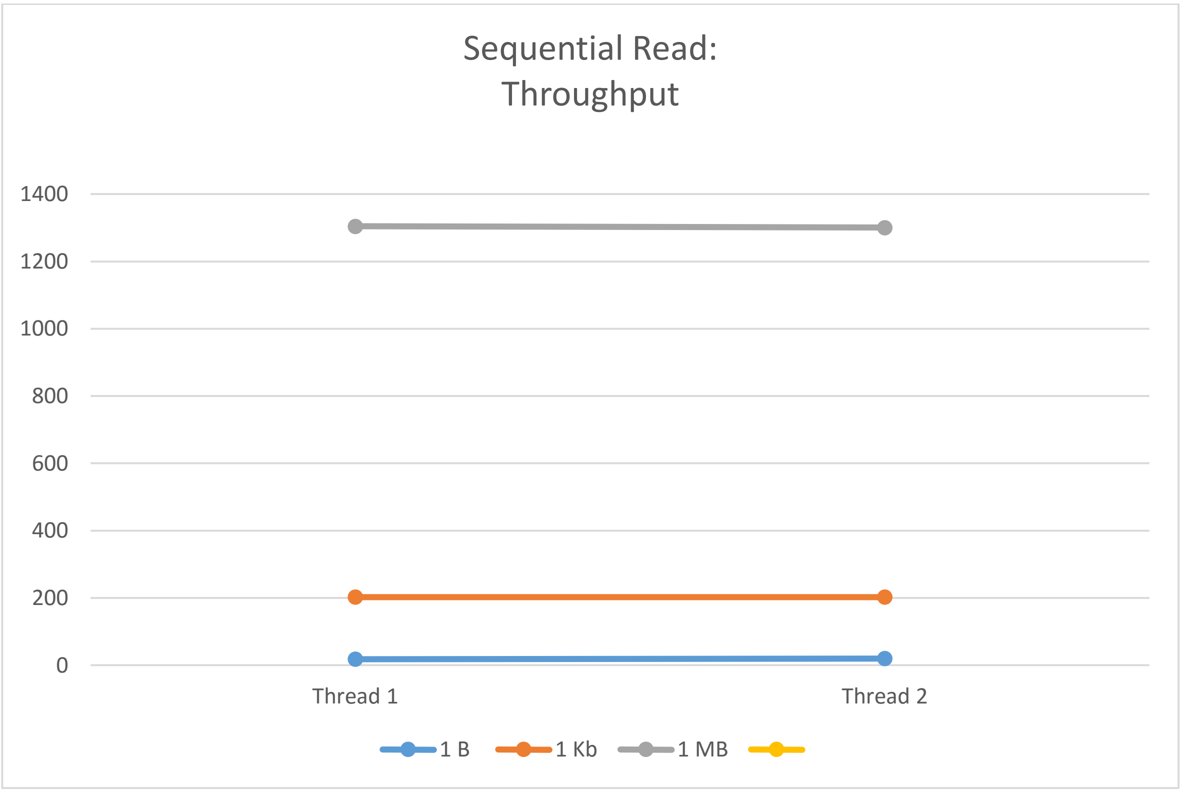
Operation per second	1 Thread (Average)	Standard Deviation	2 Thread (Average)	Standard Deviations	4 Thread (Average)	Standard Deviations
GFLOPS/sec	0.26861	0.00042	0.536252	0	1.07542	0.00767
GIOPS/sec	0.667033	0	0.571743	0	0.615723	0

DISK Benchmark :

- Disk Performance is measured in terms of Latency and Throughput. Latency is measured in seconds and Throughput is measured in MB/s

Sequential and Read :

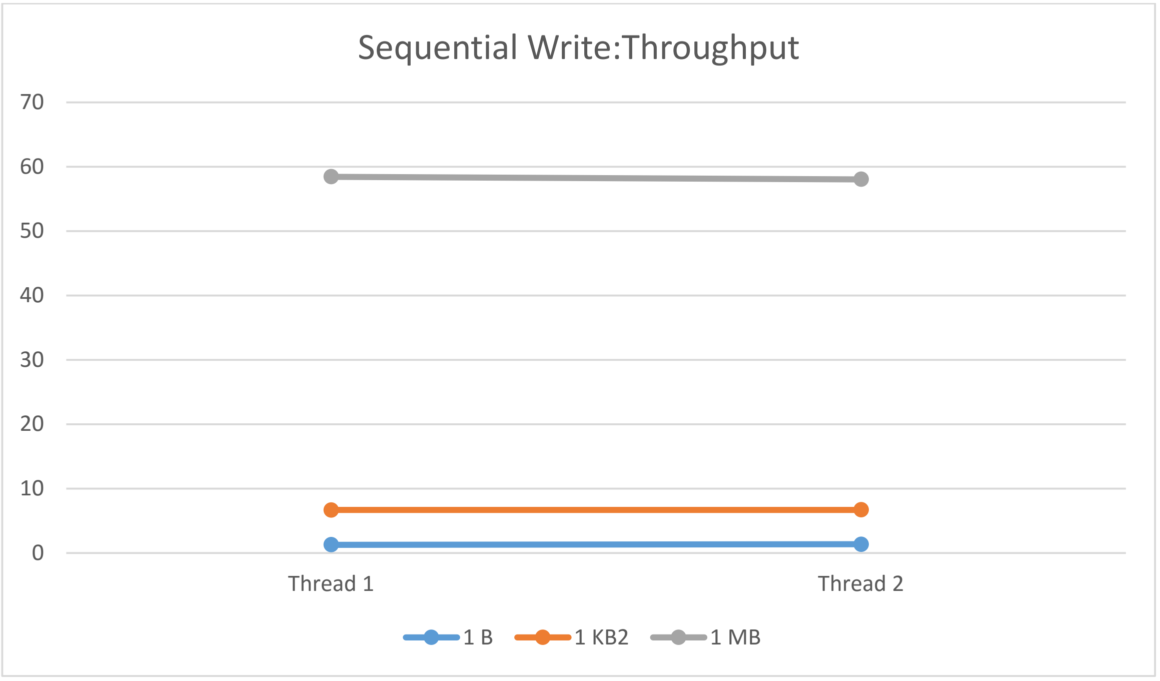
Threads	1B	1KB	1MB	
1	18.2556	202.742	1304.47	Throughput
2	20.1699	202.837	1300.72	Throughput
1	4.81445e-08	4.814e-06	0.0001386	Latency
2	4.59961e-08	4.629e-06	0.000139	Latency



Thread is taken in the x axis and Throughput in MB/s is taken in the y axis

Sequential and Write :

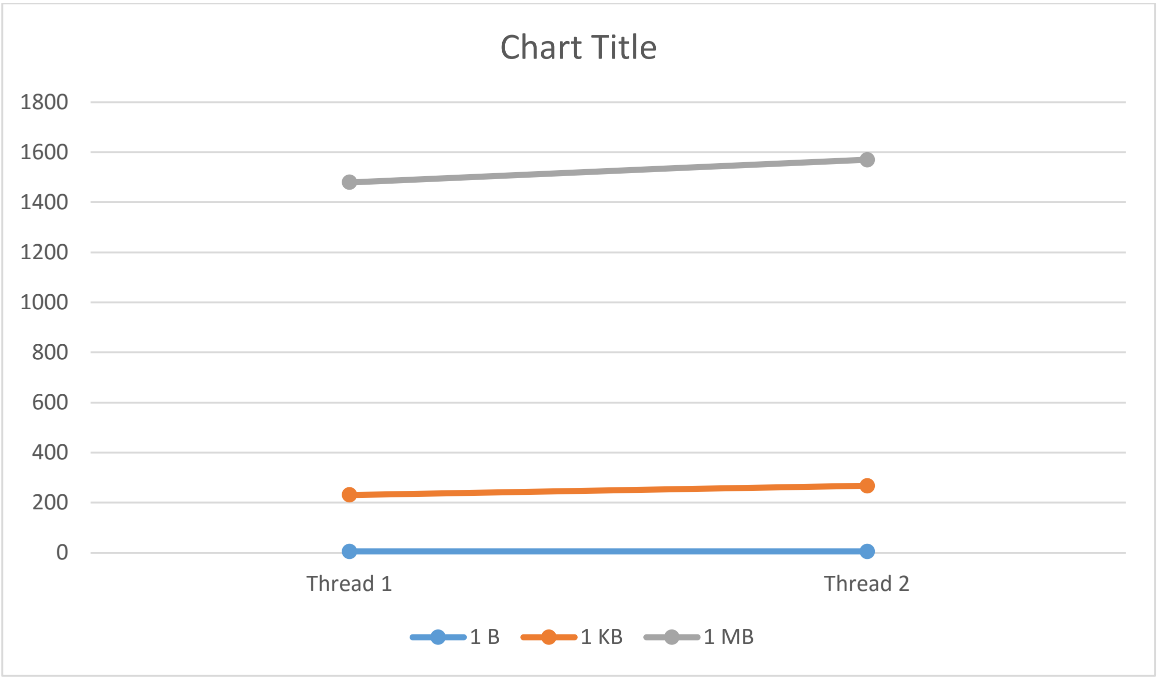
Threads	1B	1KB	1MB	
1	1.26778	6.65615	58.4249	Throughput
2	1.3478	6.6968	58.0349	Throughput
1	6.93262e-07	0.00014573	0.017116	Latency
2	6.8833e-07	0.00010476	0.017231	Latency



Thread is taken in the x axis and Throughput in MB/s is taken in the y axis

Random and Read:

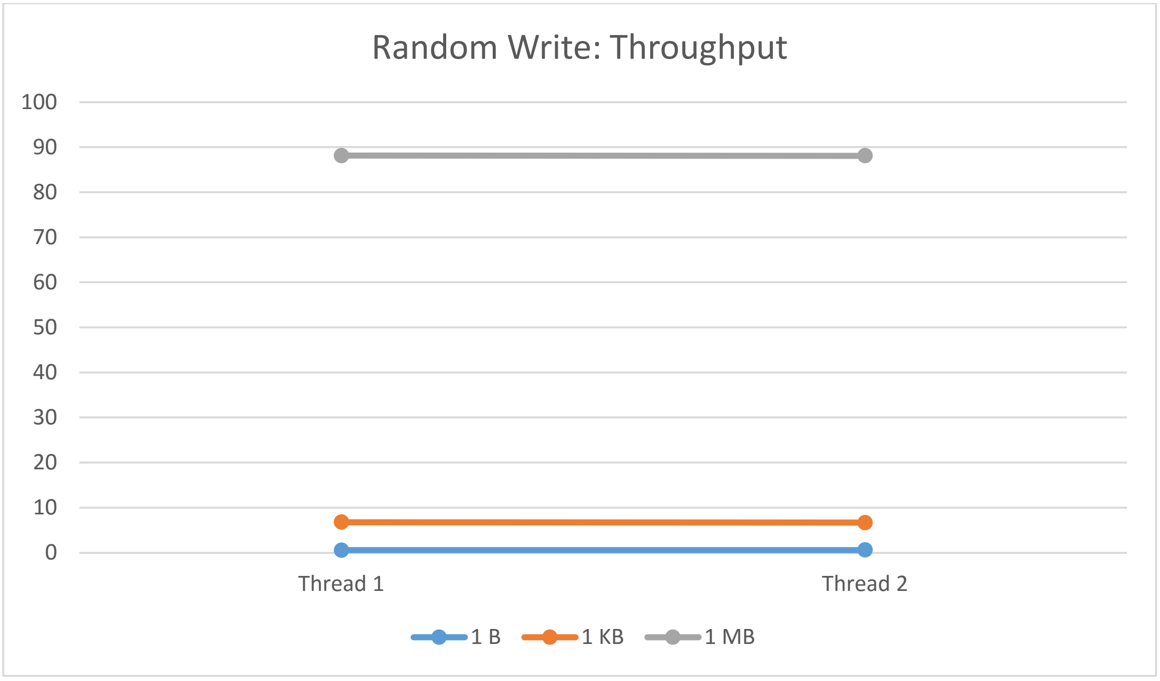
Threads	1B	1KB	1MB	
1	4.80256	230.951	1479.54	Throughput
2	5.08429	267.105	1569.44	Throughput
1	1.83008e-07	4.226e-06	0.0001222	Latency
2	1.82471e-07	3.654e-06	0.0001152	Latency



Thread is taken in the x axis and Throughput in MB/s is taken in the y axis

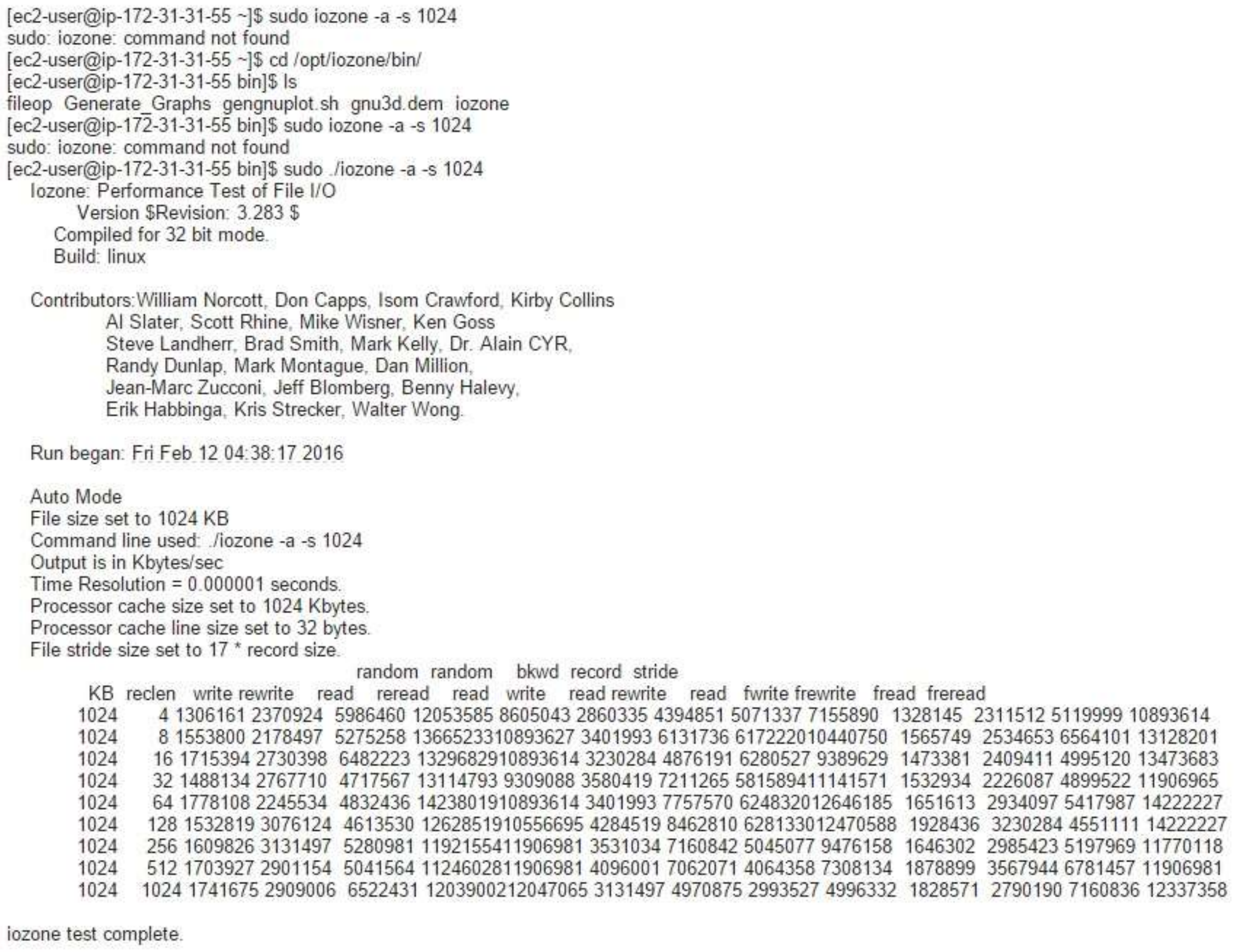
Random and Write:

Threads	1B	1KB	1MB	
1	0.555144	6.77895	88.1368	Throughput
2	0.585228	6.68244	88.1018	Throughput
1	1.5832e-06	0.00014366	0.011346	Latency
2	1.58525e-06	0.00014309	0.0113505	Latency



Experiment 2

iozone was run to estimate the disk performance



```
[ec2-user@ip-172-31-31-55 ~]$ sudo iozone -a -s 1024
sudo: iozone: command not found
[ec2-user@ip-172-31-31-55 ~]$ cd /opt/iozone/bin/
[ec2-user@ip-172-31-31-55 bin]$ ls
fileop  Generate_Graphs  gengnuplot.sh  gnu3d.dem  iozone
[ec2-user@ip-172-31-31-55 bin]$ sudo iozone -a -s 1024
sudo: iozone: command not found
[ec2-user@ip-172-31-31-55 bin]$ sudo ./iozone -a -s 1024
    Iozone: Performance Test of File I/O
            Version $Revision: 3.283 $
        Compiled for 32 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,
                 Jean-Marc Zucconi, Jeff Blomberg, Benny Halevy,
                 Erik Habbinga, Kris Strecker, Walter Wong.

    Run began: Fri Feb 12 04:38:17 2016

    Auto Mode
    File size set to 1024 KB
    Command line used: ./iozone -a -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 1306161 2370924  5986460 12053585 8605043 2860335 4394851 5071337 7155890  1328145  2311512 5119999 10893614
        1024       8 1553800 2178497  5275258 1366523310893627 3401993 6131736 617222010440750  1565749  2534653 6564101 13128201
        1024      16 1715394 2730398  6482223 1329682910893614 3230284 4876191 6280527 9389629  1473381  2409411 4995120 13473683
        1024      32 1488134 2767710  4717567 13114793 9309088 3580419 7211265 581589411141571  1532934  2226087 4899522 11906965
        1024      64 1778108 2245534  4832436 1423801910893614 3401993 7757570 624832012646185  1651613  2934097 5417987 14222227
        1024     128 1532819 3076124  4613530 1262851910556695 4284519 8462810 628133012470588  1928436  3230284 4551111 14222227
        1024     256 1609826 3131497  5280981 1192155411906981 3531034 7160842 5045077 9476158  1646302  2985423 5197969 11770118
        1024     512 1703927 2901154  5041564 1124602811906981 4096001 7062071 4064358 7308134  1878899  3567944 6781457 11906981
        1024    1024 1741675 2909006  6522431 1203900212047065 3131497 4970875 2993527 4996332  1828571  2790190 7160836 12337358

iozone test complete.
```

Conclusion

- As the threads increases the speed of transfer also increases
- Reading is always faster than writing
- As the block size increases the speed of transfer increases
- Sequential Read/write is always faster than Random Read/write

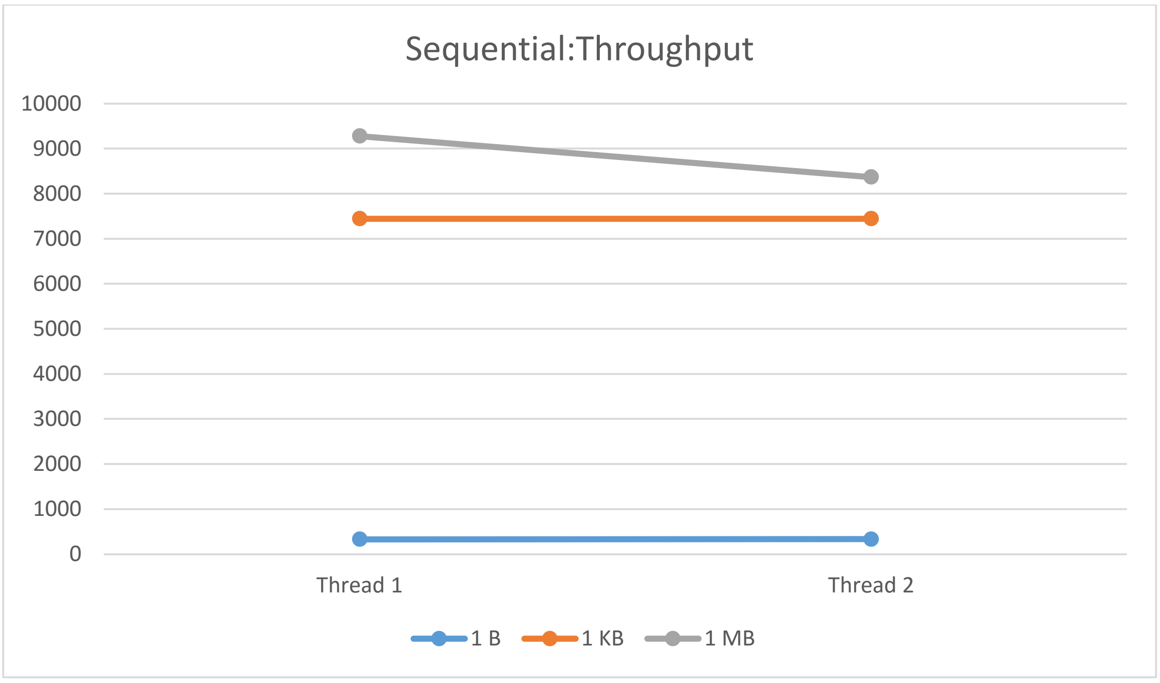
Memory Benchmark

Aim - To determine the performance of the memory.

The memory speed is measured in Latency and Throughput. Latency is measured in seconds and Throughput is measured in MB/s

Sequential:

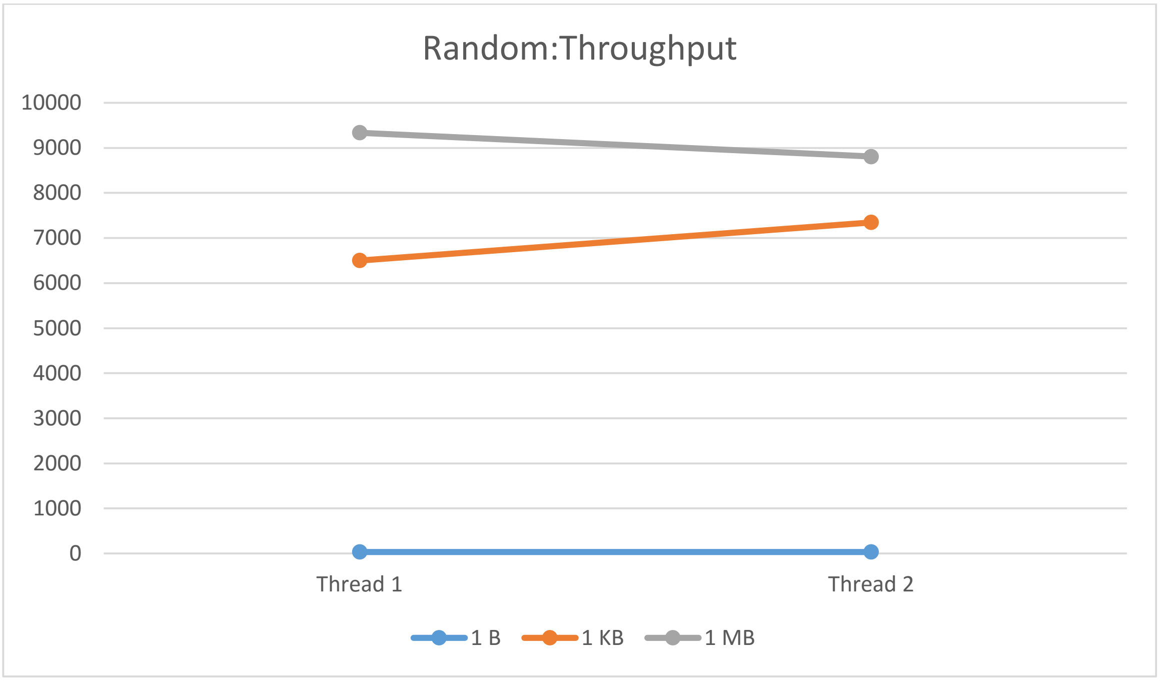
Threads	1B	1KB	1MB	
1	330.142	7443.25	9276.44	Throughput
2	331.51	7445.56	8368.2	Throughput
1	2.88868e-09	1.31201e-07	0.0001078	Latency
2	2.87676e-09	1.31001e-07	0.0001195	Latency



Thread is taken in the x axis and Throughput in MB/s is taken in the y axis

<u>Random:</u>

Threads	1B	1KB	1MB	
1	33.1455	6501.95	9337.07	Throughput
2	35.205	7347.54	8806.69	Throughput
1	2.87724e-08	1.50195e-07	0.0001071	Latency
2	2.87676e-09s	1.31001e-07	0.00011355	Latency



Thread is taken in the x axis and Throughput in MB/s is taken in the y axis

Experiment 2

steam was run to estimate the memory benchmarking

```
-------------------------------------------------------------
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 25488 microseconds.
   (= 25488 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:           6130.1     0.026749     0.026101     0.027534
Scale:          5987.2     0.027532     0.026724     0.028531
Add:            8531.2     0.028552     0.028132     0.029243
Triad:          7968.9     0.031052     0.030117     0.031613
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
```

Conclusion:

- As the threads increases the speed of transfer also increases
- Reading is always faster than writing
- As the block size increases the speed of transfer increases
- Sequential Read/write is always faster than Random Read/write
- Memory is faster than disk

System Information:

All the experiments were carried out in Amazon AWS EC2 system. The configuration is as follows

```
[ec2-user@ip-172-31-31-55 ~]$ 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:                 62
Model name:            Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
Stepping:              4
CPU MHz:               2494.056
BogoMIPS:              4988.11
Hypervisor vendor:     Xen
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              25600K
NUMA node0 CPU(s):     0
```