CLOUD COMPUTING

Performance Evaluation Document

Programming Assignment 1

Vinit Shah

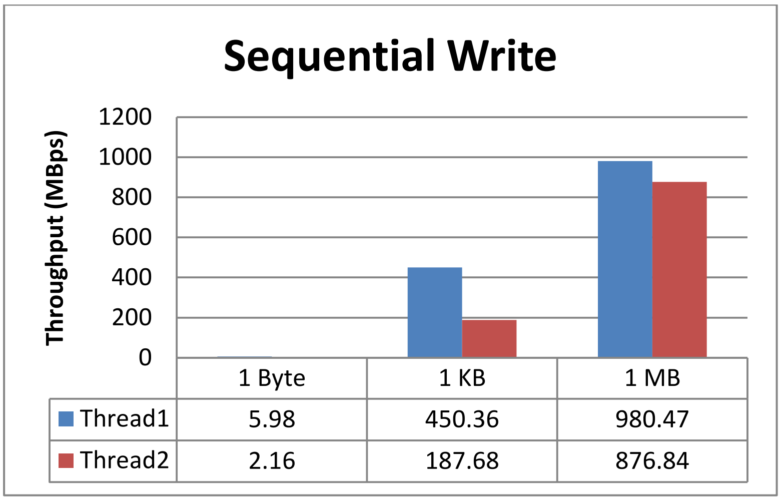
A20350453

Disk Benchmarking

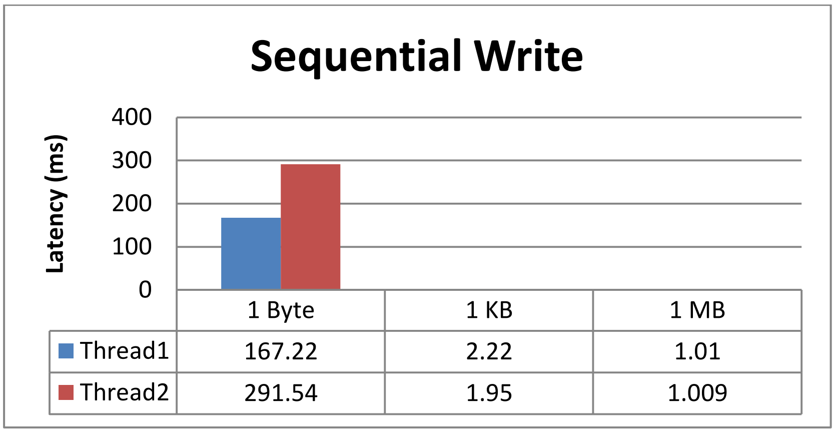
Sr no	Operation	Data Size	Throughput(MBPS)	Latency(ms)	Number of Threads
1	Write Sequential	1B	5.98	167.22	1
2	Write Sequential	1KB	450.36	2.220	1
3	Write Sequential	1MB	980.47	1.01	1
4	Read Sequential	1B	3.87	258.39	1
5	Read Sequential	1KB	850.90	1.1752	1
6	Read Sequential	1MB	2021	0.49	1
7	write Random	1B	2.57	389.10	1
8	write Random	1KB	410.84	2.43	1
9	write Random	1MB	870.58	1.14	1
10	Read Random	1B	0.84	1190.47	1
11	Read Random	1KB	170.85	5.88	1
12	Read Random	1MB	870.87	1.148	1
13	write Sequential	1B	3.43	291.54	2
14	write Sequential	1KB	510.24	1.95	2
15	write Sequential	1MB	990.74	1.009	2
16	Read Sequential	1B	5.86	170.64	2
17	Read Sequential	1KB	810.89	1.23	2
18	Read Sequential	1MB	2947.74	0.339	2
19	Write Random	1B	2.16	262.96	2
20	Write Random	1KB	187.68	5.32	2
21	Write Random	1MB	876.84	1.14	2
22	Read Random	1B	1.35	1130.85	2
23	Read Random	1KB	170.84	5.85	2
24	Read Random	1MB	994.87	1.005	2

Disk benchmarking is one of the techniques to find out how fast the Disk is while reading and writing data to and fro. Based on set of input size and required number of threads, we have analyzed the disk performance for throughput and latency. Through the experiments we can conclude that the time required to read and write sequentially is less than the time required to read or write randomly. With varying number of threads the capability of disk to read and write is generally constant as we can analyze from the graph.

Sequential Write:

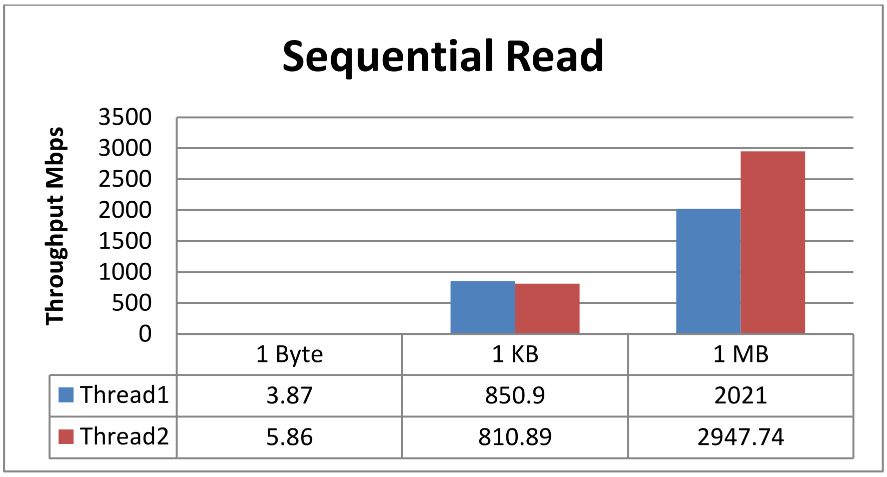


	1 Byte	1 KB	1 MB
🟦 Thread1	5.98	450.36	980.47
🟥 Thread2	2.16	187.68	876.84

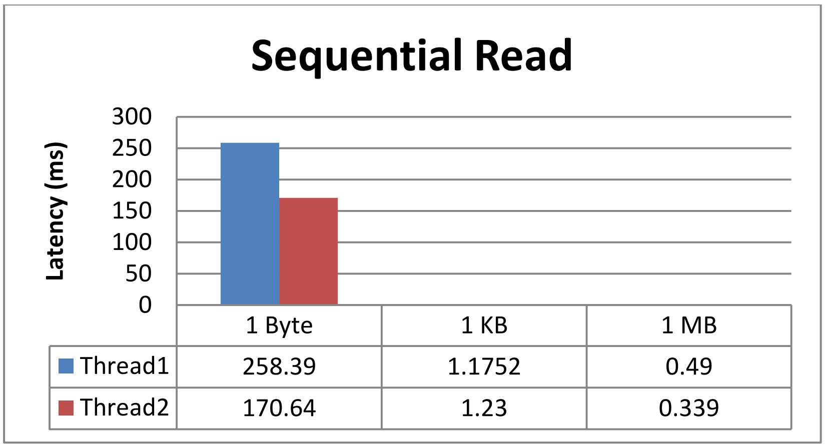


	1 Byte	1 KB	1 MB
🟦 Thread1	167.22	2.22	1.01
🟥 Thread2	291.54	1.95	1.009

Sequential Read:

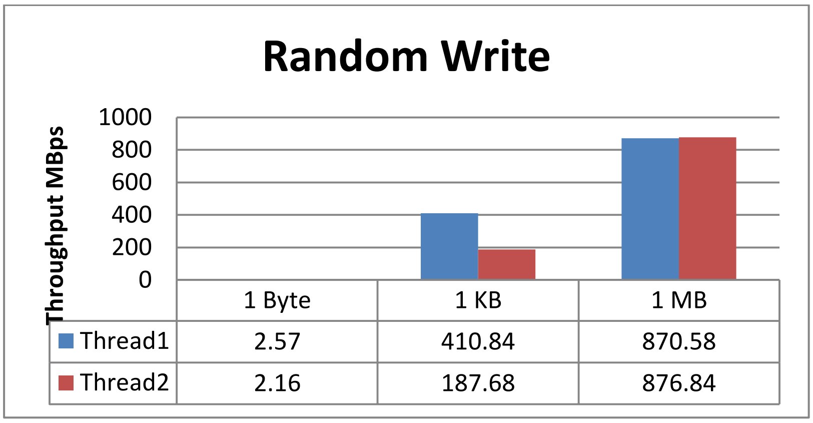


	1 Byte	1 KB	1 MB
🟦 Thread1	3.87	850.9	2021
🟥 Thread2	5.86	810.89	2947.74

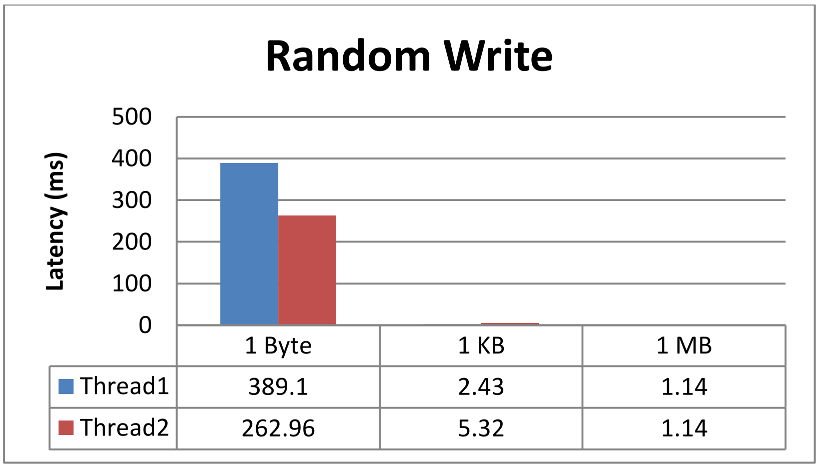


	1 Byte	1 KB	1 MB
🟦 Thread1	258.39	1.1752	0.49
🟥 Thread2	170.64	1.23	0.339

Random Write:

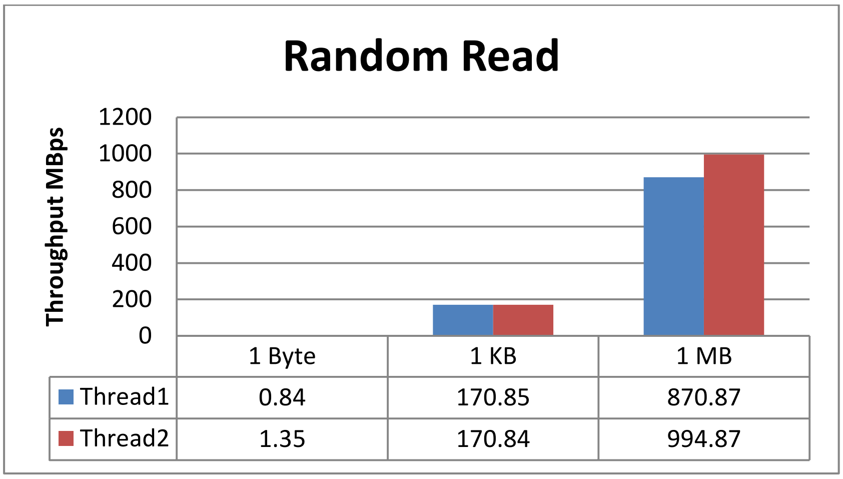


	1 Byte	1 KB	1 MB
🟦 Thread1	2.57	410.84	870.58
🟥 Thread2	2.16	187.68	876.84

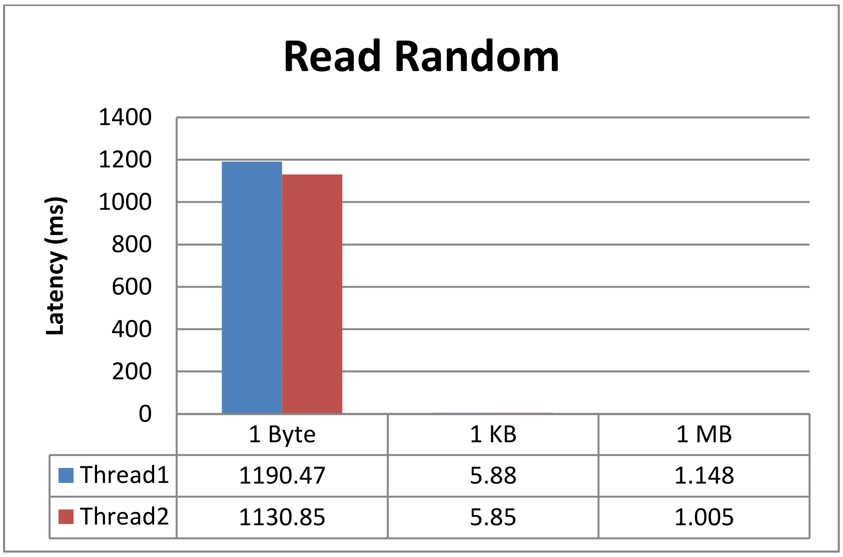


	1 Byte	1 KB	1 MB
🟦 Thread1	389.1	2.43	1.14
🟥 Thread2	262.96	5.32	1.14

Random Read:

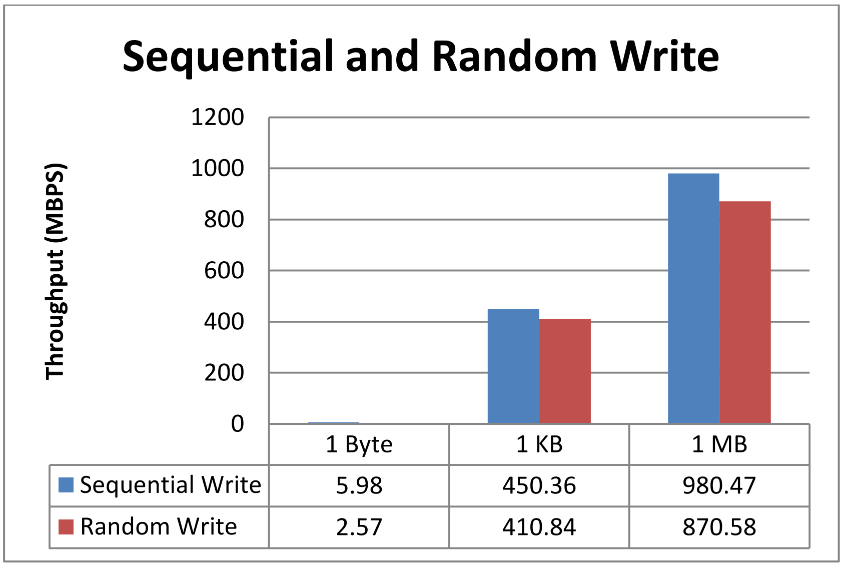


	1 Byte	1 KB	1 MB
🟦 Thread1	0.84	170.85	870.87
🟥 Thread2	1.35	170.84	994.87

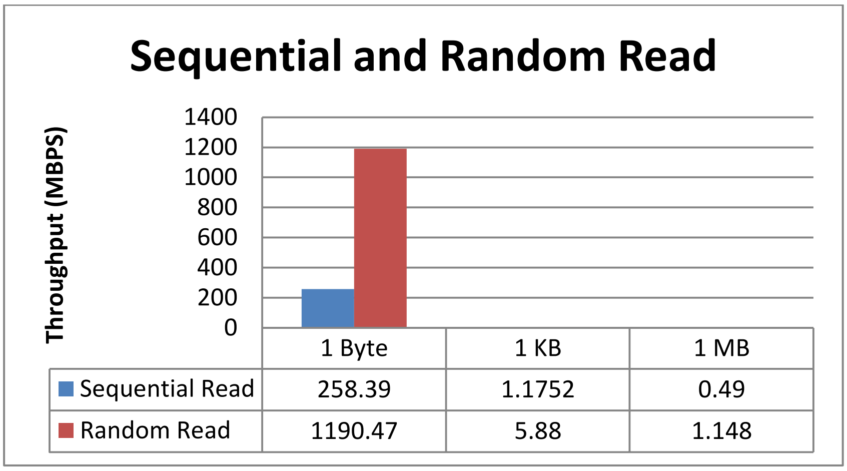


	1 Byte	1 KB	1 MB
🟦 Thread1	1190.47	5.88	1.148
🟥 Thread2	1130.85	5.85	1.005

Comparing Sequential and Random Access:



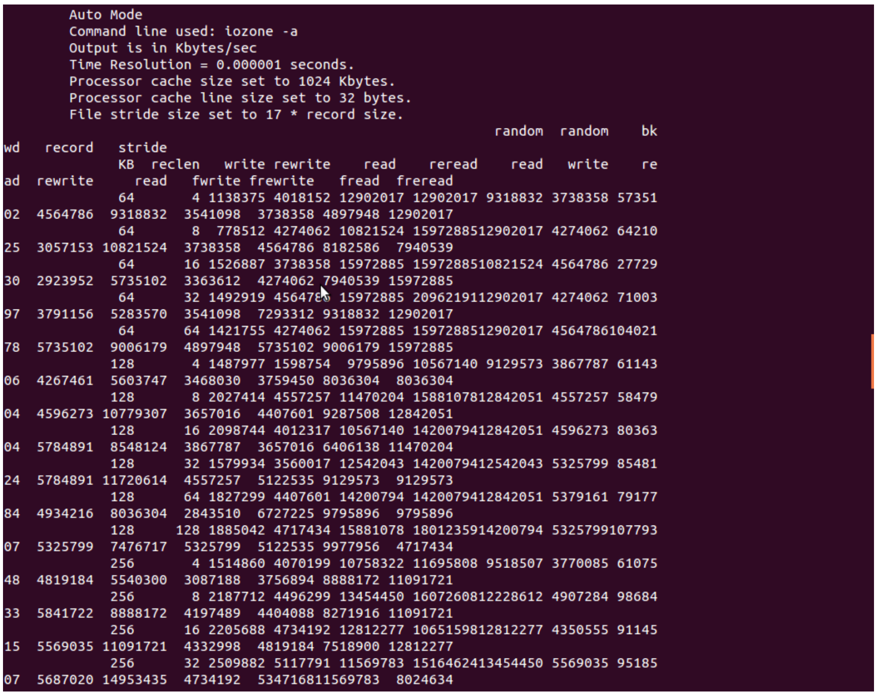
	1 Byte	1 KB	1 MB
🟦 Sequential Write	5.98	450.36	980.47
🟥 Random Write	2.57	410.84	870.58



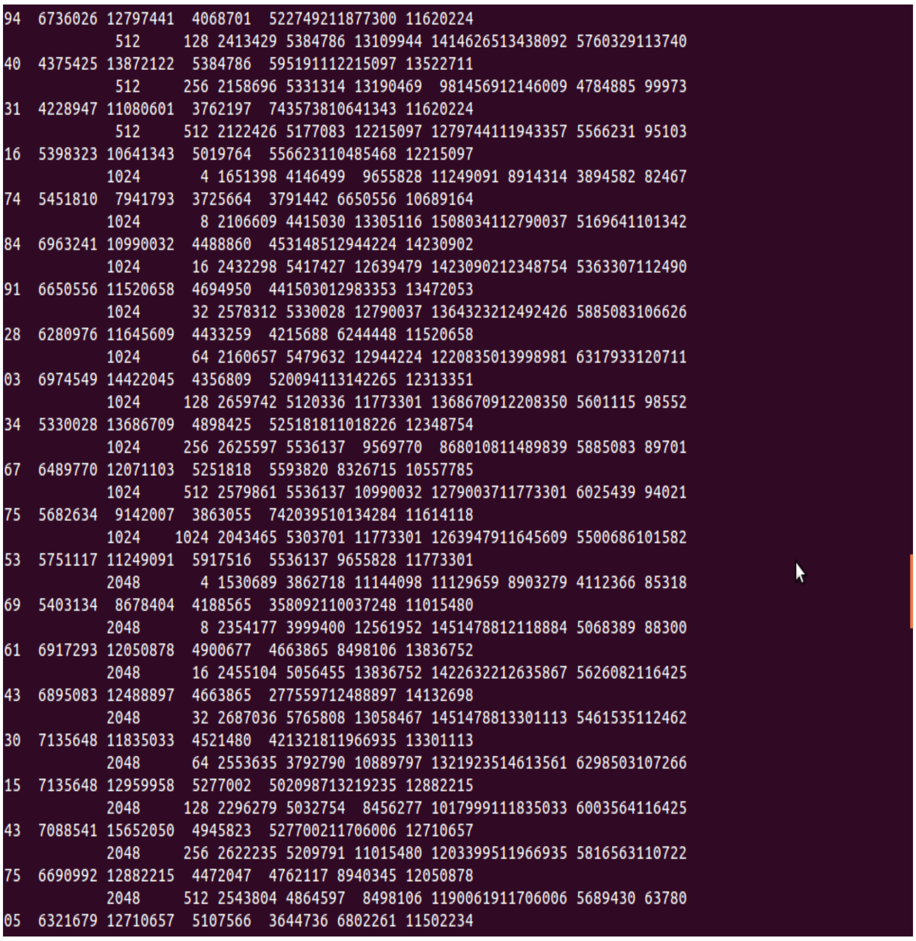
	1 Byte	1 KB	1 MB
🟦 Sequential Read	258.39	1.1752	0.49
🟥 Random Read	1190.47	5.88	1.148

IOZONE:

IOZONE is the Disk bench mark which is a standard used to verify the disk capability to transfer data sequentially and randomly, based on different data size.



```
        Auto Mode
        Command line used: iozone -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.
                                                            random  random    bk
wd   record   stride
              KB  reclen   write rewrite    read    reread    read   write    re
ad  rewrite     read   fwrite frewrite   fread  freread
              64       4 1138375 4018152 12902017 12902017 9318832 3738358 57351
02  4564786  9318832  3541098  3738358 4897948 12902017
              64       8  778512 4274062 10821524 1597288512902017 4274062 64210
25  3057153 10821524  3738358  4564786 8182586  7940539
              64      16 1526887 3738358 15972885 1597288510821524 4564786 27729
30  2923952  5735102  3363612  4274062 7940539 15972885
              64      32 1492919 4564786 15972885 2096219112902017 4274062 71003
97  3791156  5283570  3541098  7293312 9318832 12902017
              64      64 1421755 4274062 15972885 1597288512902017 4564786104021
78  5735102  9006179  4897948  5735102 9006179 15972885
             128       4 1487977 1598754  9795896 10567140 9129573 3867787 61143
06  4267461  5603747  3468030  3759450 8036304  8036304
             128       8 2027414 4557257 11470204 1588107812842051 4557257 58479
04  4596273 10779307  3657016  4407601 9287508 12842051
             128      16 2098744 4012317 10567140 1420079412842051 4596273 80363
04  5784891  8548124  3867787  3657016 6406138 11470204
             128      32 1579934 3560017 12542043 1420079412542043 5325799 85481
24  5784891 11720614  4557257  5122535 9129573  9129573
             128      64 1827299 4407601 14200794 1420079412842051 5379161 79177
84  4934216  8036304  2843510  6727225 9795896  9795896
             128     128 1885042 4717434 15881078 1801235914200794 5325799107793
07  5325799  7476717  5325799  5122535 9977956  4717434
             256       4 1514860 4070199 10758322 11695808 9518507 3770085 61075
48  4819184  5540300  3087188  3756894 8888172 11091721
             256       8 2187712 4496299 13454450 1607260812228612 4907284 98684
33  5841722  8888172  4197489  4404088 8271916 11091721
             256      16 2205688 4734192 12812277 1065159812812277 4350555 91145
15  5569035 11091721  4332998  4819184 7518900 12812277
             256      32 2509882 5117791 11569783 1516462413454450 5569035 95185
07  5687020 14953435  4734192  534716811569783  8024634
```



```
94  6736026 12797441  4068701  522749211877300 11620224
             512      128 2413429 5384786 13109944 1414626513438092 5760329113740
40  4375425 13872122  5384786  595191112215097 13522711
             512      256 2158696 5331314 13190469  981456912146009 4784885 99973
31  4228947 11080601  3762197  743573810641343 11620224
             512      512 2122426 5177083 12215097 1279744111943357 5566231 95103
16  5398323 10641343  5019764  556623110485468 12215097
            1024        4 1651398 4146499  9655828 11249091 8914314 3894582 82467
74  5451810  7941793  3725664  3791442 6650556 10689164
            1024        8 2106609 4415030 13305116 1508034112790037 5169641101342
84  6963241 10990032  4488860  453148512944224 14230902
            1024       16 2432298 5417427 12639479 1423090212348754 5363307112490
91  6650556 11520658  4694950  441503012983353 13472053
            1024       32 2578312 5330028 12790037 1364323212492426 5885083106626
28  6280976 11645609  4433259  4215688 6244448 11520658
            1024       64 2160657 5479632 12944224 1220835013998981 6317933120711
03  6974549 14422045  4356809  520094113142265 12313351
            1024      128 2659742 5120336 11773301 1368670912208350 5601115 98552
34  5330028 13686709  4898425  525181811018226 12348754
            1024      256 2625597 5536137  9569770  868010811489839 5885083 89701
67  6489770 12071103  5251818  5593820 8326715 10557785
            1024      512 2579861 5536137 10990032 1279003711773301 6025439 94021
75  5682634  9142007  3863055  742039510134284 11614118
            1024     1024 2043465 5303701 11773301 1263947911645609 5500686101582
53  5751117 11249091  5917516  5536137 9655828 11773301
            2048        4 1530689 3862718 11144098 11129659 8903279 4112366 85318
69  5403134  8678404  4188565  358092110037248 11015480
            2048        8 2354177 3999400 12561952 1451478812118884 5068389 88300
61  6917293 12050878  4900677  4663865 8498106 13836752
            2048       16 2455104 5056455 13836752 1422632212635867 5626082116425
43  6895083 12488897  4663865  277559712488897 14132698
            2048       32 2687036 5765808 13058467 1451478813301113 5461535112462
30  7135648 11835033  4521480  421321811966935 13301113
            2048       64 2553635 3792790 10889797 1321923514613561 6298503107266
15  7135648 12959958  5277002  502098713219235 12882215
            2048      128 2296279 5032754  8456277 1017999111835033 6003564116425
43  7088541 15652050  4945823  527700211706006 12710657
            2048      256 2622235 5209791 11015480 1203399511966935 5816563110722
75  6690992 12882215  4472047  4762117 8940345 12050878
            2048      512 2543804 4864597  8498106 1190061911706006 5689430 63780
05  6321679 12710657  5107566  3644736 6802261 11502234
```

Network Benchmarking

a,b,c:

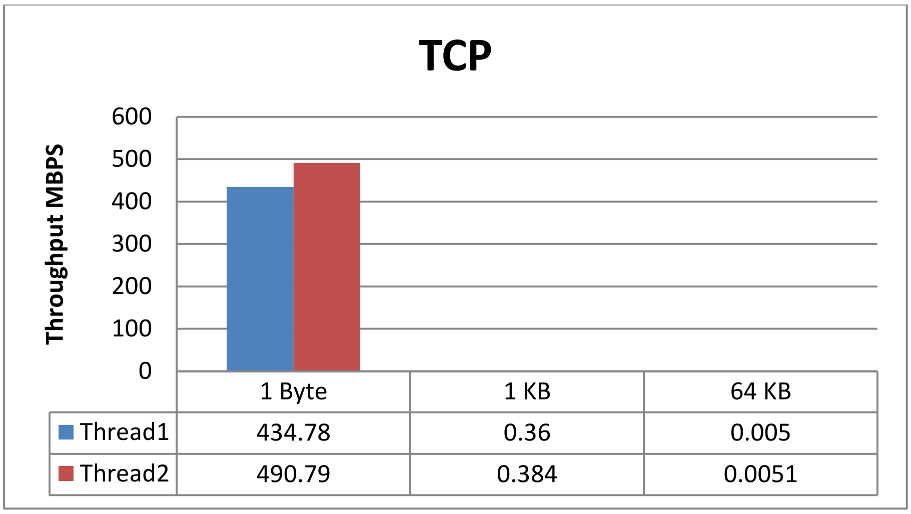
Sr no	Operation	Data Size	Throughput(MBPS)	Throughput (Mbits per sec)	Latency(ms)	Number of Threads
1	TCP	1 Byte	0.00165	0.0132	0.00204	1
2	TCP	1 KB	1.6975	13.58	1.746	1
3	TCP	64KB	94.5925	756.74	440.78	1
4	UDP	1Byte	0.0023	0.0184	0.739	1
5	UDP	1KB	2.72	21.76	1.10	1
6	UDP	64KB	199.533	1596.264	370.98	1
7	TCP	1 Byte	0.001875	0.0150	0.00216	2
8	TCP	1 KB	1.7625	14.10	1.84	2
9	TCP	64KB	98.8425	790.74	450.41	2
10	UDP	1Byte	0.0020375	0.0163	0.024	2
11	UDP	1KB	2.6	20.8	1.05	2
12	UDP	64KB	194.55	1556.4	390.80	2

Benchmarking of Network, to calculate bandwidth of network. As we can analyze through readings and graph for both TCP and UDP Protocols.

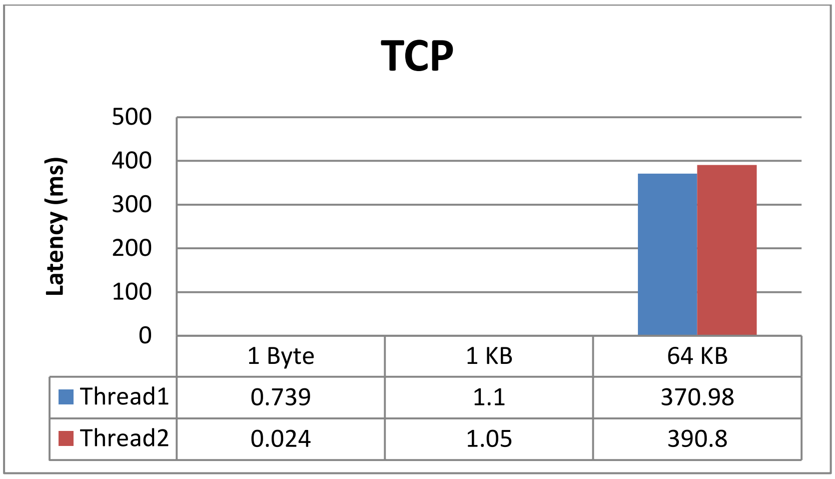
Multithreaded sever is able to handle request from multiple client at the same time with the capability to serve each request.

TCP:

Now as we can analyze from graph of TCP, as it’s a connection oriented protocol and each buffer of data is transferred in byte stream from client to server and server to client. The time taken for transferring the data in buffer of 1 Byte, 1 KB and 64 KB to travel from client to server and from server to client (Round-trip time) with an acknowledgement from both parties , is usually more as compared to the UDP, as data are transferred in stream of bytes in TCP. Now throughput of Network is almost the same for server with one client and server with two clients. This indicates servers’ capability to handle multiple client requests and serving with equal, as it will seems like there is a dedicated server for each client. The same performance is achieved by 1 thread and 2 thread clients is because the bandwidth is being shared by both the threads which gives the almost equal throughput and latency. As the size of Buffer to be transferred increases, we can see that the throughput of network is also increases and latency decreases.



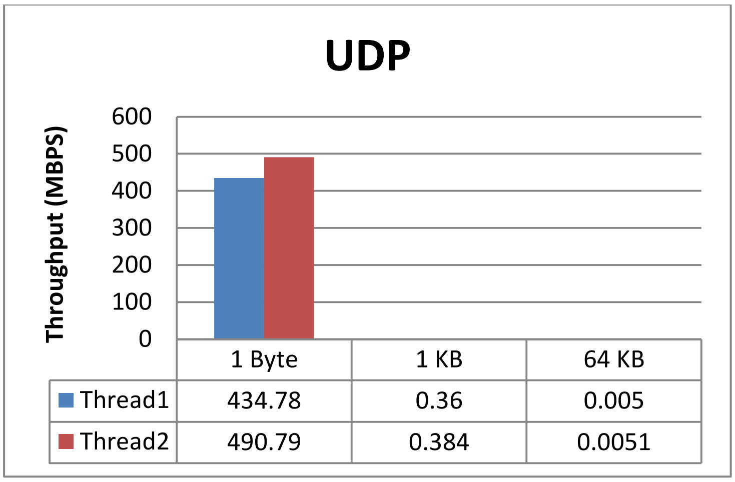
	1 Byte	1 KB	64 KB
🟦 Thread1	434.78	0.36	0.005
🟥 Thread2	490.79	0.384	0.0051



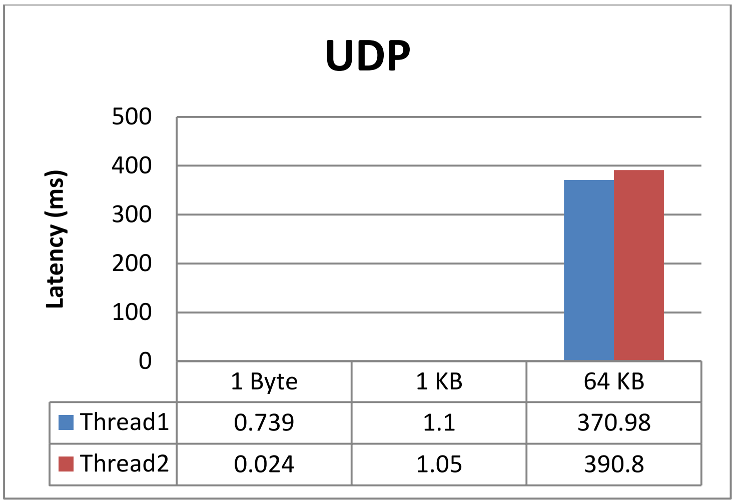
	1 Byte	1 KB	64 KB
🟦 Thread1	0.739	1.1	370.98
🟥 Thread2	0.024	1.05	390.8

UDP:

UDP is the connection less protocol, so each data travels individually from client to server and receives out of order and also there is no acknowledgment from server about reception of data unlike TCP, and data might get lost in the network. Now in UDP to transfer data from client to server the data sent in Datagram Packet. Throughput achieved through UDP in multithreaded environment is generally high as compared to TCP, because for each datagram packet to be sent from client there is a make and break of connection with server. Server also serves multiple clients at same time. In the experiment 2 threads and 1 thread performance of UDP is almost the same as the network bandwidth is being shared by threads, so that we can conclude that UDP is much faster than TCP because 1 Byte, 1KB and 64KB of data is transferred in the one datagram packet, so throughput is generally high and latency to travel data gram packet in round-trip is usually slow.



	1 Byte	1 KB	64 KB
🟦 Thread1	434.78	0.36	0.005
🟥 Thread2	490.79	0.384	0.0051



	1 Byte	1 KB	64 KB
🟦 Thread1	0.739	1.1	370.98
🟥 Thread2	0.024	1.05	390.8

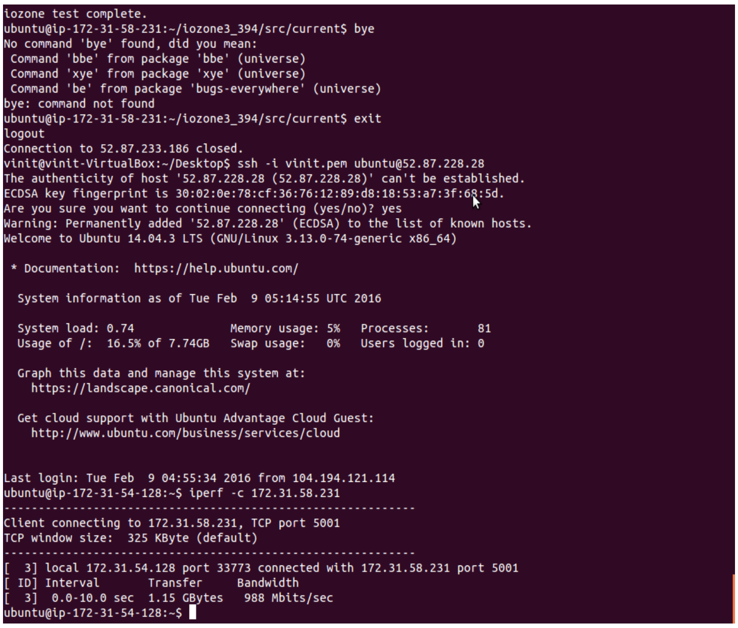
d. IPerf benchmark:-

Running the Iperf benchmark of T2 Micro, IPerf is the network bandwidth analysing tool, which measures the network throughput by for TCP and UDP protocol, as it’s a standard by calculating the round trip time of packet.

After running the IPerf on T2 Micro, for TCP and UDP we can fairly analyse that the throughput achieved through TCP is less as compared to UDP.

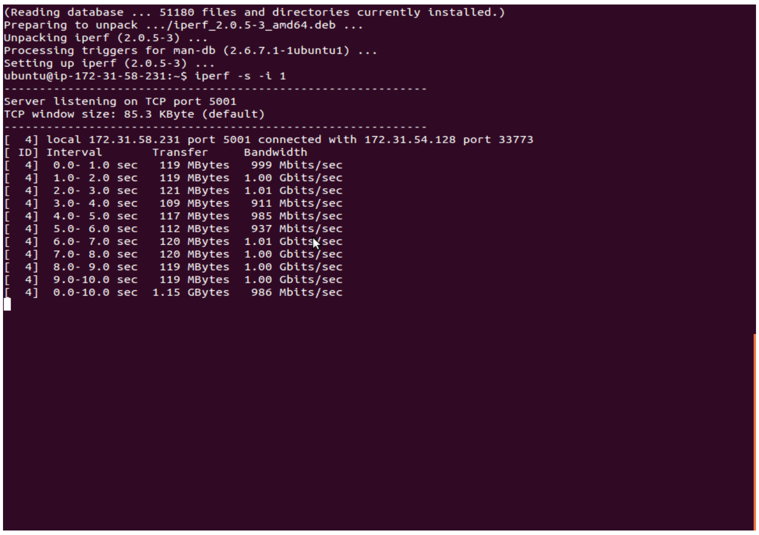
Snapshots of IPerf for TCP and UDP:

IPerf TCP Client:



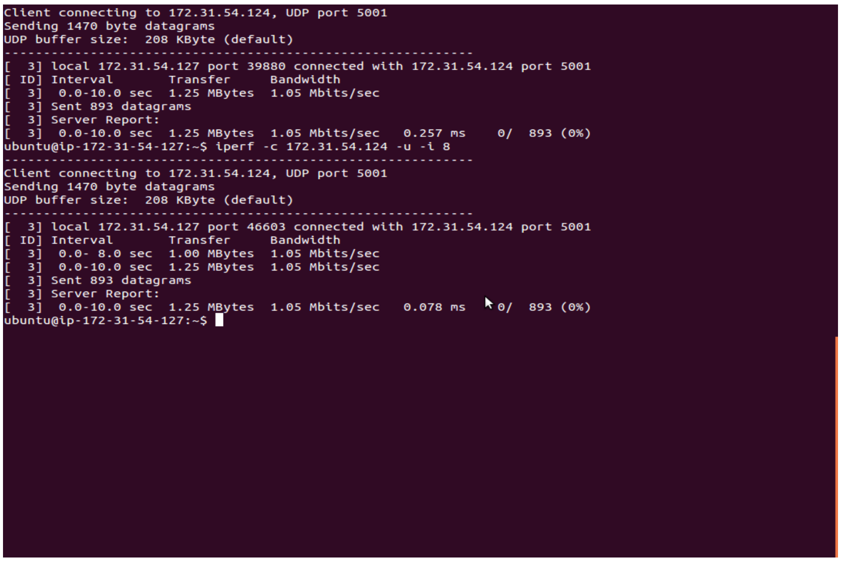
```
iozone test complete.
ubuntu@ip-172-31-58-231:~/iozone3_394/src/current$ bye
No command 'bye' found, did you mean:
 Command 'bbe' from package 'bbe' (universe)
 Command 'xye' from package 'xye' (universe)
 Command 'be' from package 'bugs-everywhere' (universe)
bye: command not found
ubuntu@ip-172-31-58-231:~/iozone3_394/src/current$ exit
logout
Connection to 52.87.233.186 closed.
vinit@vinit-VirtualBox:~/Desktop$ ssh -i vinit.pem ubuntu@52.87.228.28
The authenticity of host '52.87.228.28 (52.87.228.28)' can't be established.
ECDSA key fingerprint is 30:02:0e:78:cf:36:76:12:89:d8:18:53:a7:3f:68:5d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '52.87.228.28' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.13.0-74-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Tue Feb  9 05:14:55 UTC 2016

  System load: 0.74              Memory usage: 5%   Processes:       81
  Usage of /:  16.5% of 7.74GB   Swap usage:   0%   Users logged in: 0

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud


Last login: Tue Feb  9 04:55:34 2016 from 104.194.121.114
ubuntu@ip-172-31-54-128:~$ iperf -c 172.31.58.231
------------------------------------------------------------
Client connecting to 172.31.58.231, TCP port 5001
TCP window size:  325 KByte (default)
------------------------------------------------------------
[  3] local 172.31.54.128 port 33773 connected with 172.31.58.231 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.15 GBytes   988 Mbits/sec
ubuntu@ip-172-31-54-128:~$
```

IPerf TCP Server:



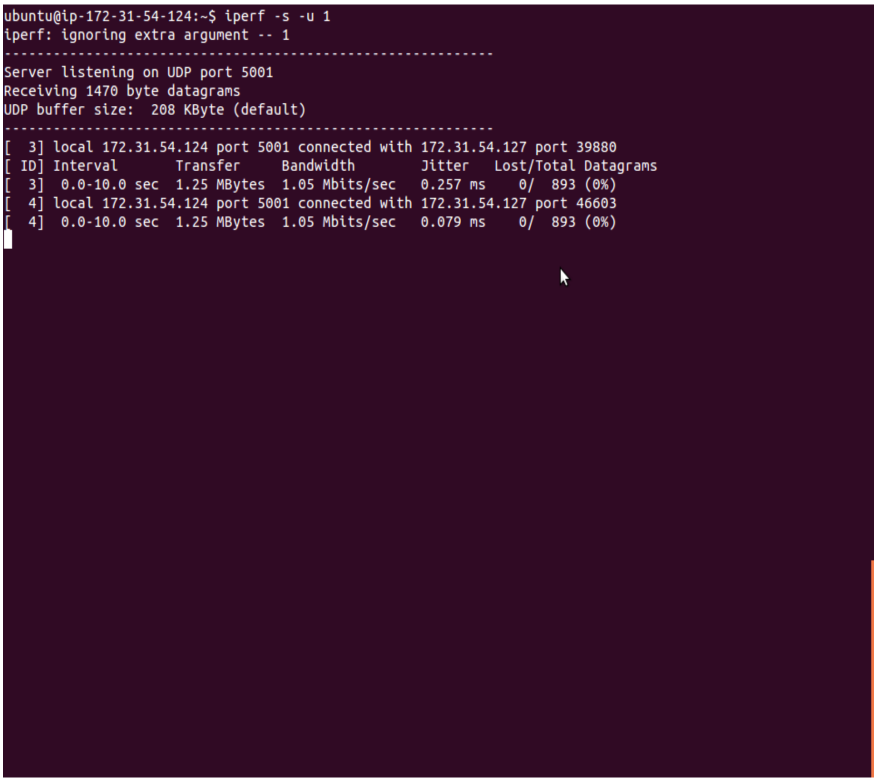
```
(Reading database ... 51180 files and directories currently installed.)
Preparing to unpack .../iperf_2.0.5-3_amd64.deb ...
Unpacking iperf (2.0.5-3) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up iperf (2.0.5-3) ...
ubuntu@ip-172-31-58-231:~$ iperf -s -i 1
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 172.31.58.231 port 5001 connected with 172.31.54.128 port 33773
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0- 1.0 sec   119 MBytes   999 Mbits/sec
[  4]  1.0- 2.0 sec   119 MBytes  1.00 Gbits/sec
[  4]  2.0- 3.0 sec   121 MBytes  1.01 Gbits/sec
[  4]  3.0- 4.0 sec   109 MBytes   911 Mbits/sec
[  4]  4.0- 5.0 sec   117 MBytes   985 Mbits/sec
[  4]  5.0- 6.0 sec   112 MBytes   937 Mbits/sec
[  4]  6.0- 7.0 sec   120 MBytes  1.01 Gbits/sec
[  4]  7.0- 8.0 sec   120 MBytes  1.00 Gbits/sec
[  4]  8.0- 9.0 sec   119 MBytes  1.00 Gbits/sec
[  4]  9.0-10.0 sec   119 MBytes  1.00 Gbits/sec
[  4]  0.0-10.0 sec  1.15 GBytes   986 Mbits/sec
```

IPerf UDP Client:



```
Client connecting to 172.31.54.124, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 172.31.54.127 port 39880 connected with 172.31.54.124 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.257 ms    0/  893 (0%)
ubuntu@ip-172-31-54-127:~$ iperf -c 172.31.54.124 -u -i 8
------------------------------------------------------------
Client connecting to 172.31.54.124, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 172.31.54.127 port 46603 connected with 172.31.54.124 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 8.0 sec  1.00 MBytes  1.05 Mbits/sec
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.078 ms    0/  893 (0%)
ubuntu@ip-172-31-54-127:~$ 
```

IPerf UDP Server:



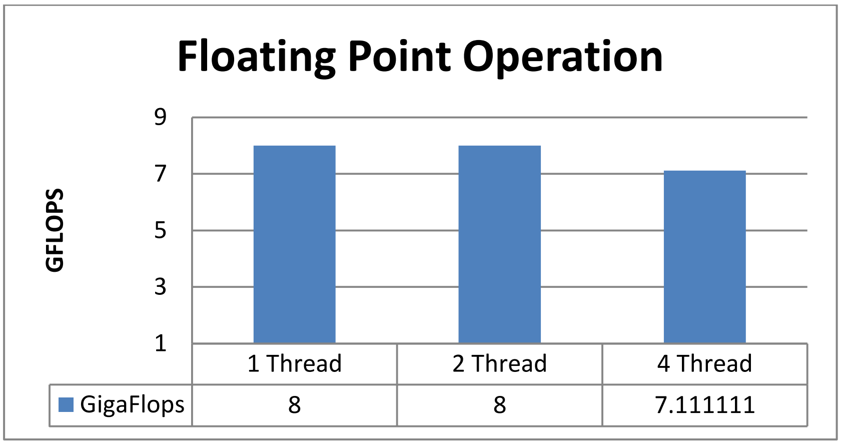
```
ubuntu@ip-172-31-54-124:~$ iperf -s -u 1
iperf: ignoring extra argument -- 1
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 172.31.54.124 port 5001 connected with 172.31.54.127 port 39880
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.257 ms    0/  893 (0%)
[  4] local 172.31.54.124 port 5001 connected with 172.31.54.127 port 46603
[  4]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.079 ms    0/  893 (0%)
```

CPU Benchmarking

a,b : Floating point operation:-

Processing the floating point operations by running the numbers of floating point instructions with 1 thread, 2 Thread and 4 threads, we will get following GFLOPS. As we can see that the as number of threads are increasing the processer's load increases as multiple threads are executing same number of instructions 1 Billion time, but because of the Core and Concurrent execution of threads in parallel, the amount of GFLOPS achieved is almost constant because of the Pipeline feature of present day's processor, which can able to execute multiple instructions at same time.

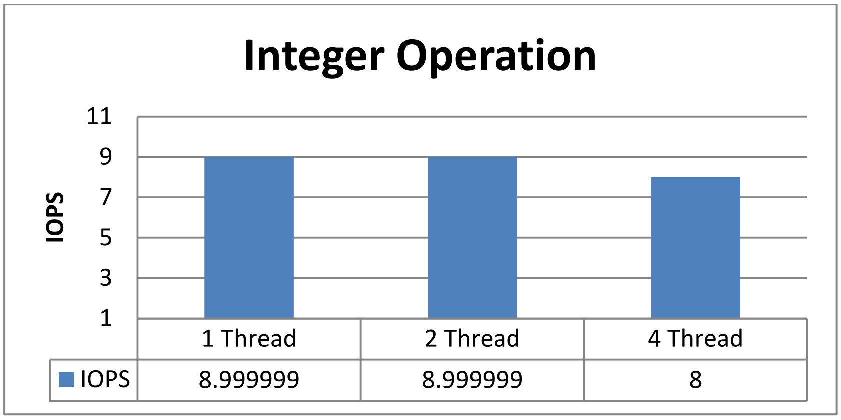
Sr. No	Operation	GFLOPS	Number of Threads
1	Floating Point	8.000000	1
2	Floating Point	8.000000	2
3	Floating Point	7.111111	4



Integer Operation:

Processing of Integer operations by running the numbers of integer instructions with 1 thread, 2 Thread and 4 threads, we will get following GIOPS. As we can see that the as number of threads are increasing the processer's load increases as multiple threads are executing same number of instructions 1 Billion time, but because of the Core and Concurrent execution of threads in parallel, the amount of GIOPS achieved is almost constant because of the Pipeline feature of present day's processor, which can able to execute multiple instructions at same time.

Sr. No	Operation	GIOPS	Number of Threads
1	Integer	8.999999	1
2	Integer	8.999999	2
3	Integer	8.000000	4



C. Theoretical Peak Performance = GHz * Number of Core * Number of Instruction/cycle

Now for Amazon EC2 T2 Micro uses the Xeon E5-2670 v2 processor.

So,

Number of Core for EC2 T2 Micro= 1

Processor Base Frequency (GHz) = 2.5 GHz

Instruction per cycle of Xeon E5-2670 v2 processor = 8

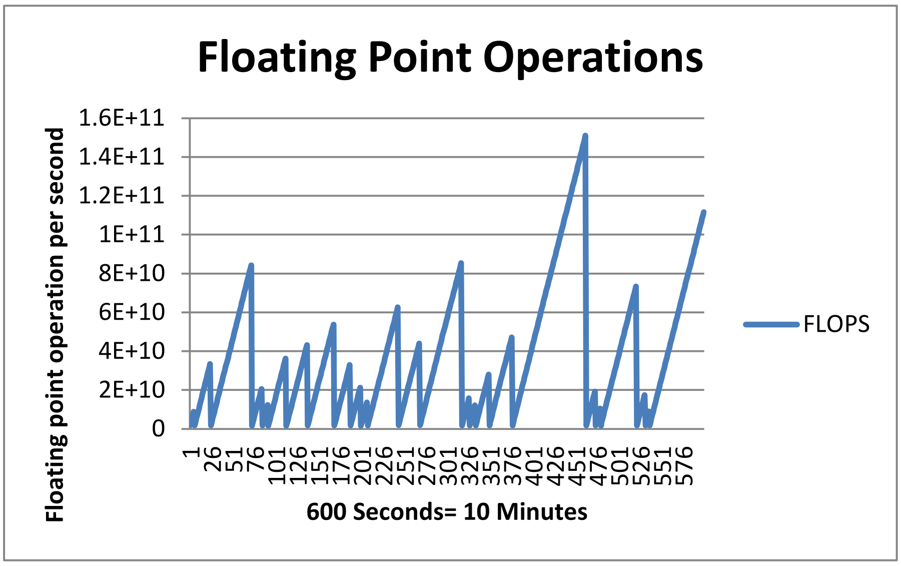
Theoretical Peak Performance (flops/sec) = 2.5 * 1 * 8

= 20 flops/sec

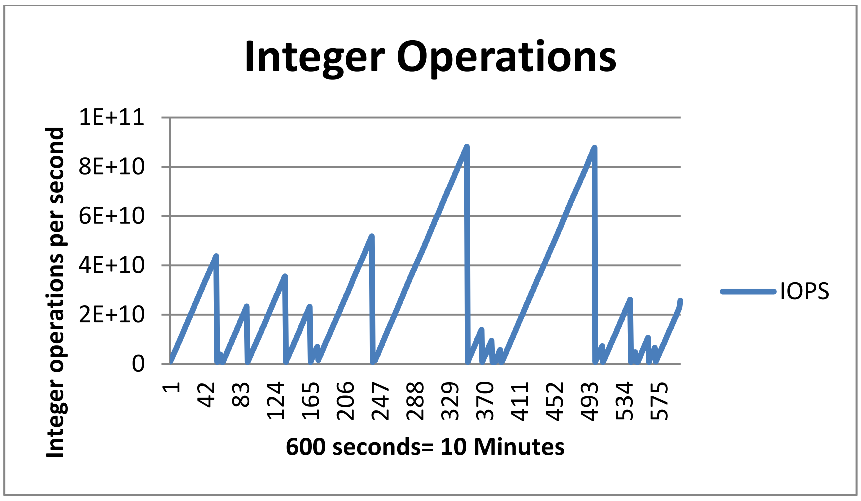
D. Compared to Theoretical performance, the efficiency achieved by running the benchmark on T2 Micro of AWS was near to around 40% of the Theoretical performance specified by the Intel Family for Xeon E5-2670 v2.

e. Benchmark on floating point and integer instructions and 4 threads for a 10-minute period for each one, with samples each second.

By running the benchmark for 10min and taking samples at each second about how many instructions are executed per second. For Execution of Floating Point operation for about 10minutes with 4 threads, processor with 1 core (EC2 T2 Micro), can able to process on an average around 36573441772 floating operations per seconds. Now the present day's processor can able to process multiple instructions at the same time because of threads and instructions pipelining feature. After analysing the graph plot of flops for 10minutes, we can deduce that the processor is able to execute almost same amount of floating point instructions each seconds but some variation is expected in the benchmark created by me as practical condition processor has many processes to executes, so the graph is not as constants. But ideal scenario is the one where we can expect the same amount of floating point instructions per seconds.

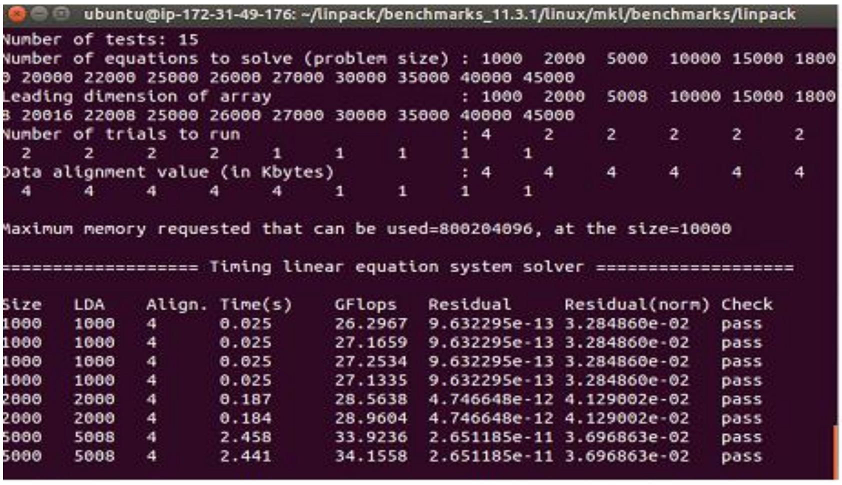


For Integer Operation, as processor can able to execute multiple instructions at one time with threading and pipelining feature, the performance of processor for execution of Integer operation is on an average 25670206268 integer operations per second. Also after plot of total integer operations on graph against each second, we can analyse that the processor's capability to execute integer instructions is almost constant at each second, in ideal scenario we can expect the same number of integer operations execution per second.



Linpack benchmarking:

It’s a CPU benchmarking tool, which gives the GFLOPS, and IOPS based on different size of instructions set. Its calculated based on the varying number of threads.



ubuntu@ip-172-31-49-176: ~/linpack/benchmarks_11.3.1/linux/mkl/benchmarks/linpack

```
Number of tests: 15
Number of equations to solve (problem size) : 1000  2000  5000  10000 15000 1800
0 20000 22000 25000 26000 27000 30000 35000 40000 45000
Leading dimension of array                  : 1000  2000  5008  10000 15000 1800
8 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      0.025      26.2967  9.632295e-13 3.284860e-02   pass
1000   1000   4      0.025      27.1659  9.632295e-13 3.284860e-02   pass
1000   1000   4      0.025      27.2534  9.632295e-13 3.284860e-02   pass
1000   1000   4      0.025      27.1335  9.632295e-13 3.284860e-02   pass
2000   2000   4      0.187      28.5638  4.746648e-12 4.129002e-02   pass
2000   2000   4      0.184      28.9604  4.746648e-12 4.129002e-02   pass
5000   5008   4      2.458      33.9236  2.651185e-11 3.696863e-02   pass
5000   5008   4      2.441      34.1558  2.651185e-11 3.696863e-02   pass
```