CS 553 Cloud Computing

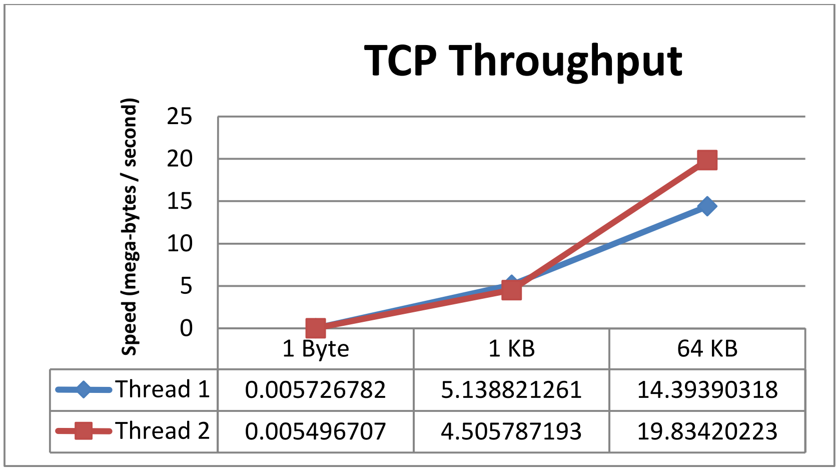
Programming Assignment 1

Sujay Gunjal (CWID: A20351746)

Performance

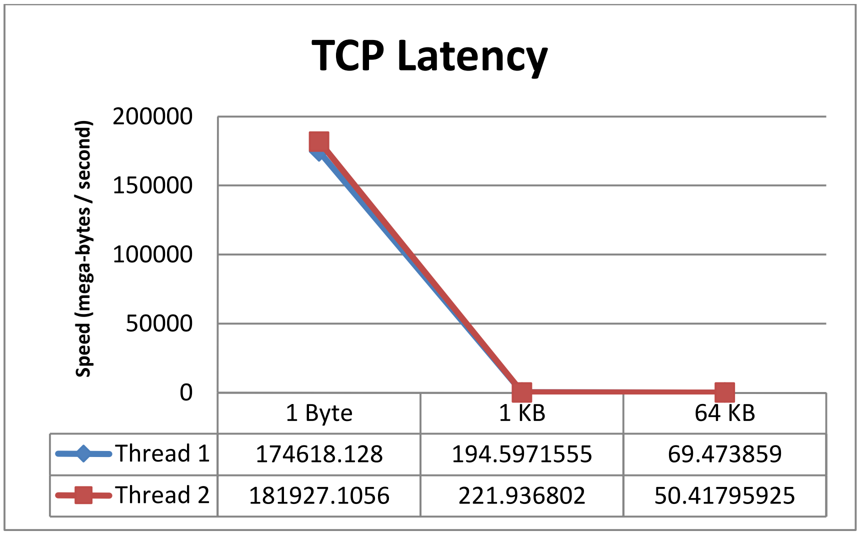
- **Network Benchmarking:**

1) TCP Throughput



	1 Byte	1 KB	64 KB
Thread 1	0.005726782	5.138821261	14.39390318
Thread 2	0.005496707	4.505787193	19.83420223

2) TCP Latency

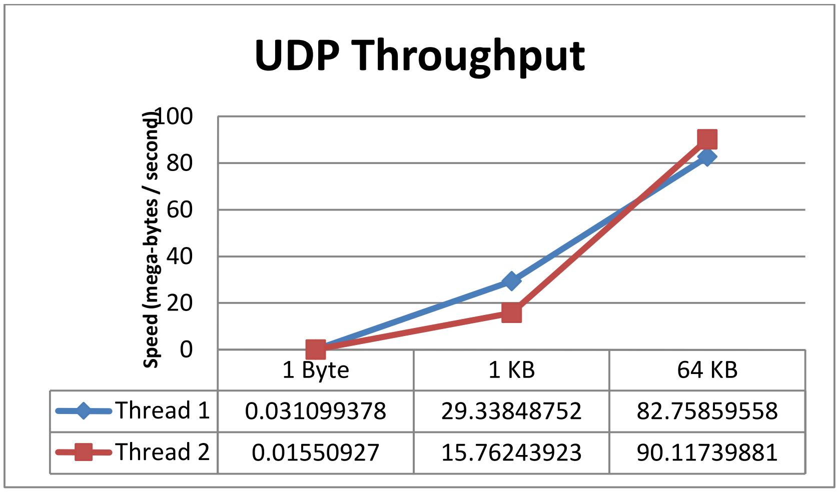


	1 Byte	1 KB	64 KB
Thread 1	174618.128	194.5971555	69.473859
Thread 2	181927.1056	221.936802	50.41795925

Observation:-

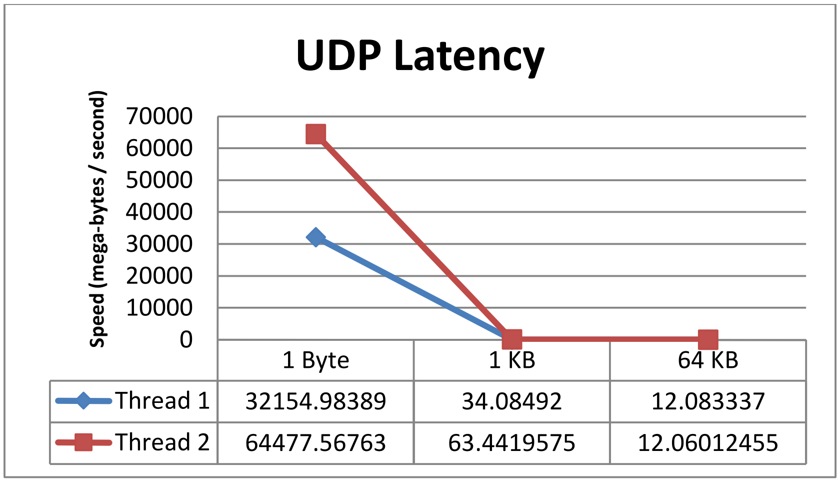
- There is drastic increasing in performance of TCP operations when we increase the packet size from 1 Byte to 64KB.
- Throughput of TCP operation goes on increasing and latency goes on decreasing as we increase the packet size.
- Throughput increases as we increase the concurrency.

3) UDP Throughput



	1 Byte	1 KB	64 KB
◆ Thread 1	0.031099378	29.33848752	82.75859558
■ Thread 2	0.01550927	15.76243923	90.11739881

4) UDP Latency



	1 Byte	1 KB	64 KB
◆ Thread 1	32154.98389	34.08492	12.083337
■ Thread 2	64477.56763	63.4419575	12.06012455

Observation:-

- There is slight decreasing in performance of UDP operations when we increase the packet size from 1 Byte to 1KB while when we increase the packet size from 1 Byte to 64KB there is drastic increasing in performance.
- Throughput of UDP operation goes on increasing and latency goes on decreasing as we increase the packet size.
- Throughput increases as we increase the concurrency.

TCP and UDP throughput comparison:-

	TCP	UDP
(1 Byte) 1 Thread	0.0057268	0.0310994
2 Threads	0.0054967	0.0155093
(1KB) 1 Thread	5.1388213	29.338488
2 Threads	4.5057872	15.762439
(64KB) 1 Thread	14.393903	82.758596
2 Threads	19.834202	90.117399

Observations:-

- We can clearly see from above table that performance of UDP protocol is far better than TCP protocol.
- But TCP is more reliable than UDP.

Theoretical Performance of a Network:

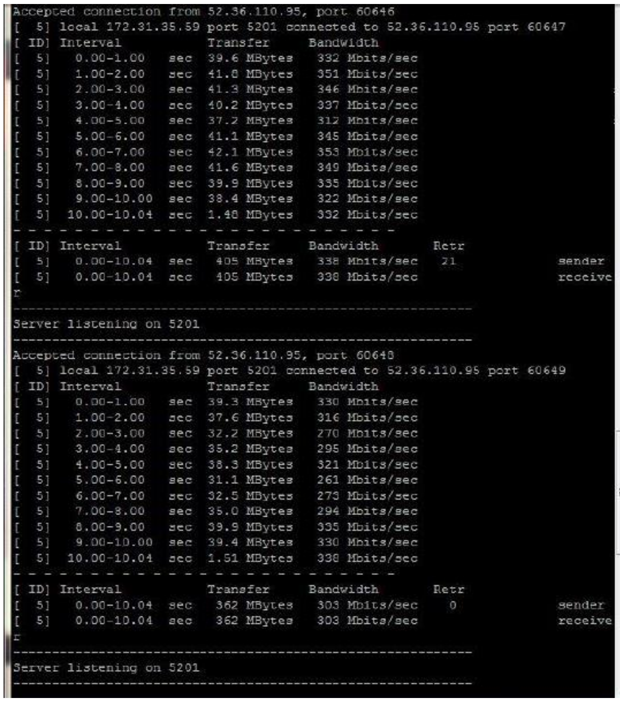
- Amazon has given t2.micro instance theoretical value for Network's Bandwidth is given as 2.7MB/S.

Extra Credit for Network Benchmarking:

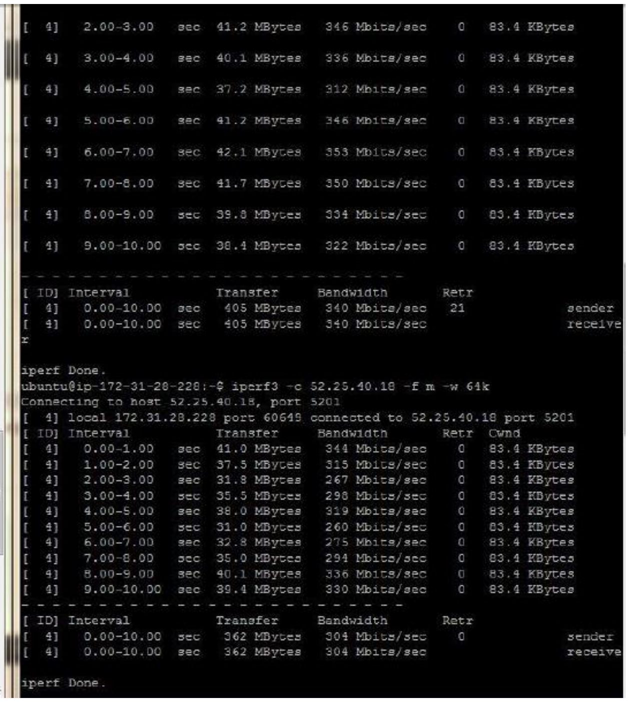
➢ Iperf Benchmarking system

- Compared evaluation of self-implemented benchmarking with Iperf benchmarking.
- Both systems (mine & Iperf) are evaluated on amazon's t2.micro instance.
- Imerf has been evaluated for packet size 64BK.

Iperf TCP :

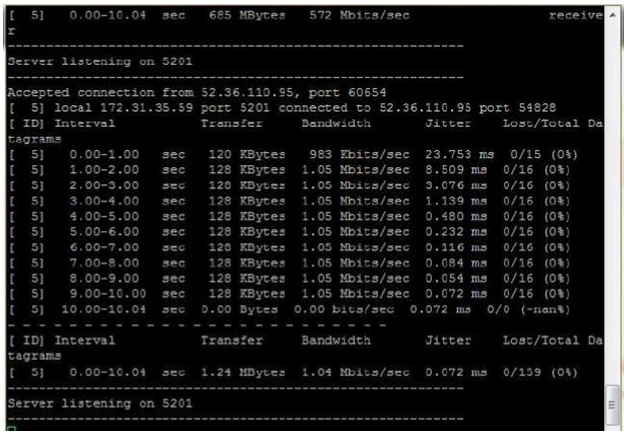


```
Accepted connection from 52.36.110.95, port 60646
[  5] local 172.31.35.59 port 5201 connected to 52.36.110.95 port 60647
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec  39.6 MBytes   332 Mbits/sec
[  5]   1.00-2.00   sec  41.8 MBytes   351 Mbits/sec
[  5]   2.00-3.00   sec  41.3 MBytes   346 Mbits/sec
[  5]   3.00-4.00   sec  40.2 MBytes   337 Mbits/sec
[  5]   4.00-5.00   sec  37.2 MBytes   312 Mbits/sec
[  5]   5.00-6.00   sec  41.1 MBytes   345 Mbits/sec
[  5]   6.00-7.00   sec  42.1 MBytes   353 Mbits/sec
[  5]   7.00-8.00   sec  41.6 MBytes   349 Mbits/sec
[  5]   8.00-9.00   sec  39.9 MBytes   335 Mbits/sec
[  5]   9.00-10.00  sec  38.4 MBytes   322 Mbits/sec
[  5]  10.00-10.04  sec  1.48 MBytes   332 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-10.04  sec   405 MBytes   338 Mbits/sec   21             sender
[  5]   0.00-10.04  sec   405 MBytes   338 Mbits/sec                  receive
r
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 52.36.110.95, port 60648
[  5] local 172.31.35.59 port 5201 connected to 52.36.110.95 port 60649
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec  39.3 MBytes   330 Mbits/sec
[  5]   1.00-2.00   sec  37.6 MBytes   316 Mbits/sec
[  5]   2.00-3.00   sec  32.2 MBytes   270 Mbits/sec
[  5]   3.00-4.00   sec  35.2 MBytes   295 Mbits/sec
[  5]   4.00-5.00   sec  38.3 MBytes   321 Mbits/sec
[  5]   5.00-6.00   sec  31.1 MBytes   261 Mbits/sec
[  5]   6.00-7.00   sec  32.5 MBytes   273 Mbits/sec
[  5]   7.00-8.00   sec  35.0 MBytes   294 Mbits/sec
[  5]   8.00-9.00   sec  39.9 MBytes   335 Mbits/sec
[  5]   9.00-10.00  sec  39.4 MBytes   330 Mbits/sec
[  5]  10.00-10.04  sec  1.51 MBytes   338 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  5]   0.00-10.04  sec   362 MBytes   303 Mbits/sec    0             sender
[  5]   0.00-10.04  sec   362 MBytes   303 Mbits/sec                  receive
r
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

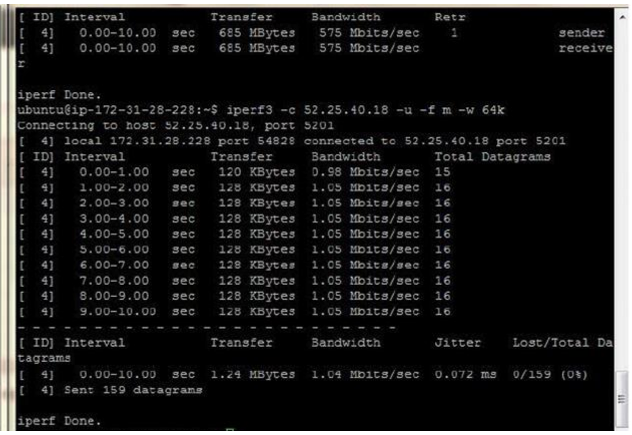


```
[  4]   2.00-3.00   sec  41.2 MBytes   346 Mbits/sec    0   83.4 KBytes

[  4]   3.00-4.00   sec  40.1 MBytes   336 Mbits/sec    0   83.4 KBytes

[  4]   4.00-5.00   sec  37.2 MBytes   312 Mbits/sec    0   83.4 KBytes

[  4]   5.00-6.00   sec  41.2 MBytes   346 Mbits/sec    0   83.4 KBytes

[  4]   6.00-7.00   sec  42.1 MBytes   353 Mbits/sec    0   83.4 KBytes

[  4]   7.00-8.00   sec  41.7 MBytes   350 Mbits/sec    0   83.4 KBytes

[  4]   8.00-9.00   sec  39.8 MBytes   334 Mbits/sec    0   83.4 KBytes

[  4]   9.00-10.00  sec  38.4 MBytes   322 Mbits/sec    0   83.4 KBytes

- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec   405 MBytes   340 Mbits/sec   21             sender
[  4]   0.00-10.00  sec   405 MBytes   340 Mbits/sec                  receive
r

iperf Done.
ubuntu@ip-172-31-28-228:~$ iperf3 -c 52.25.40.18 -f m -w 64k
Connecting to host 52.25.40.18, port 5201
[  4] local 172.31.28.228 port 60649 connected to 52.25.40.18 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  41.0 MBytes   344 Mbits/sec    0   83.4 KBytes
[  4]   1.00-2.00   sec  37.5 MBytes   315 Mbits/sec    0   83.4 KBytes
[  4]   2.00-3.00   sec  31.8 MBytes   267 Mbits/sec    0   83.4 KBytes
[  4]   3.00-4.00   sec  35.5 MBytes   298 Mbits/sec    0   83.4 KBytes
[  4]   4.00-5.00   sec  38.0 MBytes   319 Mbits/sec    0   83.4 KBytes
[  4]   5.00-6.00   sec  31.0 MBytes   260 Mbits/sec    0   83.4 KBytes
[  4]   6.00-7.00   sec  32.8 MBytes   275 Mbits/sec    0   83.4 KBytes
[  4]   7.00-8.00   sec  35.0 MBytes   294 Mbits/sec    0   83.4 KBytes
[  4]   8.00-9.00   sec  40.1 MBytes   336 Mbits/sec    0   83.4 KBytes
[  4]   9.00-10.00  sec  39.4 MBytes   330 Mbits/sec    0   83.4 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec   362 MBytes   304 Mbits/sec    0             sender
[  4]   0.00-10.00  sec   362 MBytes   304 Mbits/sec                  receive

iperf Done.
```

Iperf UDP:



```
[  5]   0.00-10.04  sec   685 MBytes   572 Mbits/sec                  receive
r
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 52.36.110.95, port 60654
[  5] local 172.31.35.59 port 5201 connected to 52.36.110.95 port 54828
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Da
tagrams
[  5]   0.00-1.00   sec   120 KBytes   983 Kbits/sec  23.753 ms  0/15 (0%)
[  5]   1.00-2.00   sec   128 KBytes  1.05 Mbits/sec  8.509 ms  0/16 (0%)
[  5]   2.00-3.00   sec   128 KBytes  1.05 Mbits/sec  3.076 ms  0/16 (0%)
[  5]   3.00-4.00   sec   128 KBytes  1.05 Mbits/sec  1.139 ms  0/16 (0%)
[  5]   4.00-5.00   sec   128 KBytes  1.05 Mbits/sec  0.480 ms  0/16 (0%)
[  5]   5.00-6.00   sec   128 KBytes  1.05 Mbits/sec  0.232 ms  0/16 (0%)
[  5]   6.00-7.00   sec   128 KBytes  1.05 Mbits/sec  0.116 ms  0/16 (0%)
[  5]   7.00-8.00   sec   128 KBytes  1.05 Mbits/sec  0.084 ms  0/16 (0%)
[  5]   8.00-9.00   sec   128 KBytes  1.05 Mbits/sec  0.054 ms  0/16 (0%)
[  5]   9.00-10.00  sec   128 KBytes  1.05 Mbits/sec  0.072 ms  0/16 (0%)
[  5]  10.00-10.04  sec  0.00 Bytes  0.00 bits/sec  0.072 ms  0/0 (-nan%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Da
tagrams
[  5]   0.00-10.04  sec  1.24 MBytes  1.04 Mbits/sec  0.072 ms  0/159 (0%)
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```



```
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec   685 MBytes   575 Mbits/sec    1             sender
[  4]   0.00-10.00  sec   685 MBytes   575 Mbits/sec                  receive
r

iperf Done.
ubuntu@ip-172-31-28-228:~$ iperf3 -c 52.25.40.18 -u -f m -w 64k
Connecting to host 52.25.40.18, port 5201
[  4] local 172.31.28.228 port 54828 connected to 52.25.40.18 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   120 KBytes  0.98 Mbits/sec  15
[  4]   1.00-2.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   2.00-3.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   3.00-4.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   4.00-5.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   5.00-6.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   6.00-7.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   7.00-8.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   8.00-9.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   9.00-10.00  sec   128 KBytes  1.05 Mbits/sec  16
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Da
tagrams
[  4]   0.00-10.00  sec  1.24 MBytes  1.04 Mbits/sec  0.072 ms  0/159 (0%)
[  4] Sent 159 datagrams

iperf Done.
```

Comparing this values with my system values, efficiency for my system is calculated and found to be high as compared to the Iperf performance.

- Instruction to TCP Iperf: 1. Install Iperf benchmark in local :

  Sudo apt-get install iperf3

2. Run TCP server Iperf3 –s
3. Run TCP client : Iperf3 –c @"public_ip"–f m –w 64k

- Instruction to UDP Iperf: 1. Install Iperf benchmark in local :

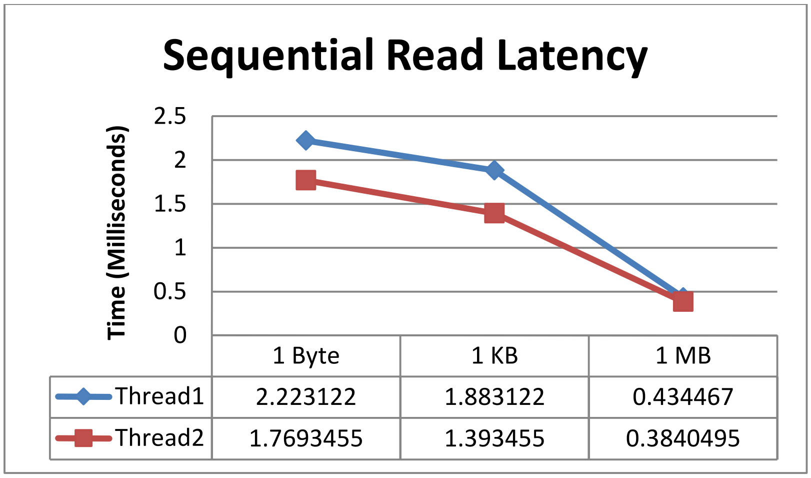
Sudo apt-get install iperf3
2. Run TCP server Iperf3 –s
3. Run TCP client : Iperf3 –c @"public_ip"–f m –w 64k

- For UDP:

1. For running a Server : Iperf3 –s
2. Run UDP client: Iperf3 –c @"public_ip" –u –f m –w 64k .

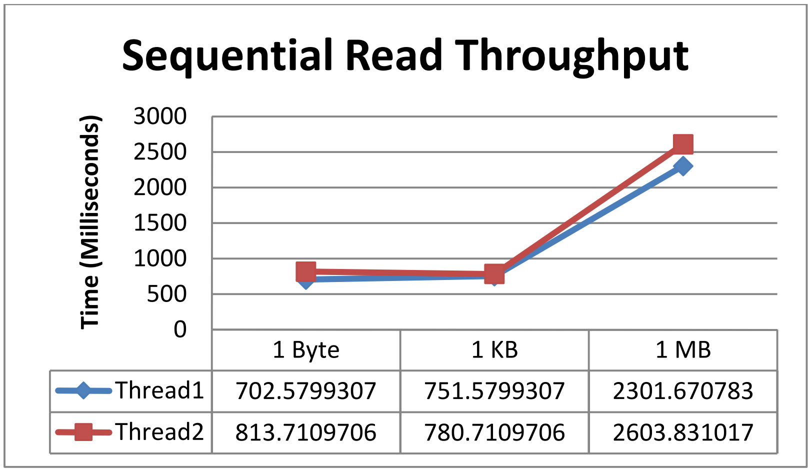
2) Disk Benchmarking:-

1) Sequential Read Latency:-



	1 Byte	1 KB	1 MB
Thread1	2.223122	1.883122	0.434467
Thread2	1.7693455	1.393455	0.3840495

2) Sequential Read Throughput:-

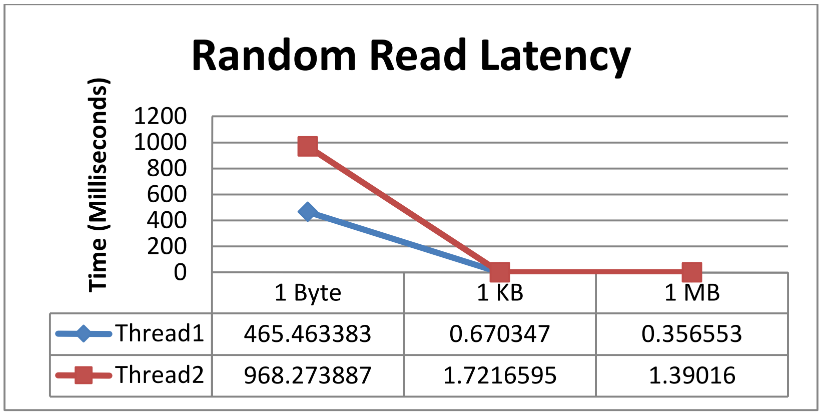


	1 Byte	1 KB	1 MB
Thread1	702.5799307	751.5799307	2301.670783
Thread2	813.7109706	780.7109706	2603.831017

Observation:-

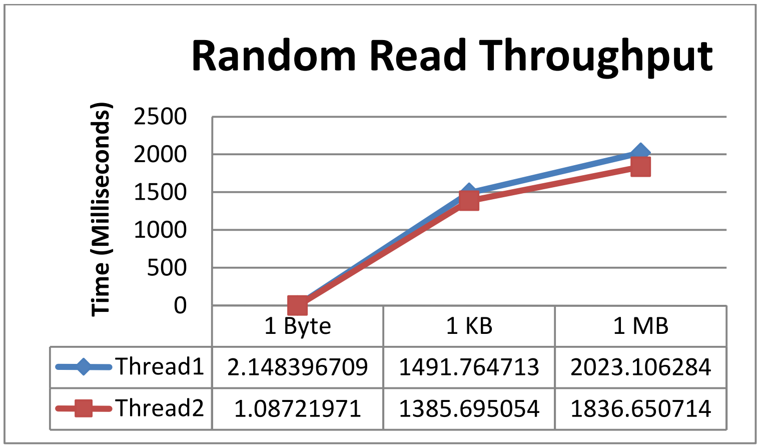
- There is increase in throughput as we increase the block-size from 1 Byte, 1 KB to 1MB.
- Throughput increases as we increase the concurrency and latency goes on decreasing.

3) Random Read Operation Latency:-



	1 Byte	1 KB	1 MB
◆ Thread1	465.463383	0.670347	0.356553
■ Thread2	968.273887	1.7216595	1.39016

4) Random Read Operation Throughput:-

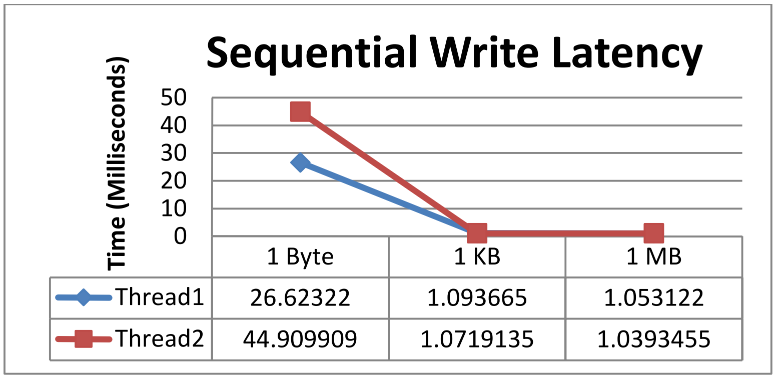


	1 Byte	1 KB	1 MB
◆ Thread1	2.148396709	1491.764713	2023.106284
■ Thread2	1.08721971	1385.695054	1836.650714

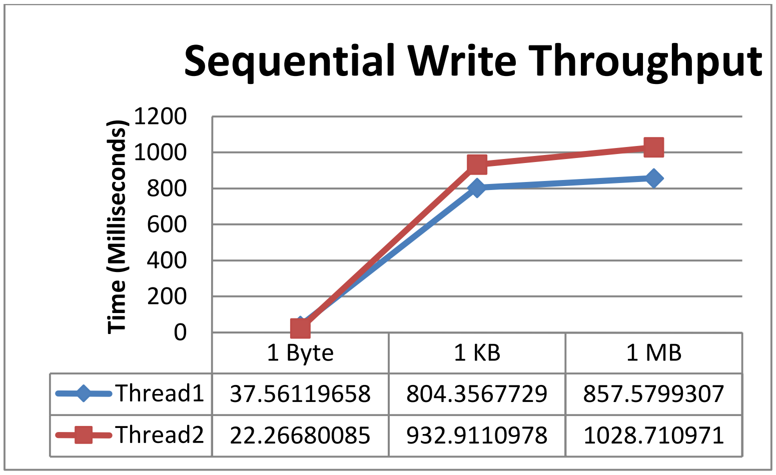
Observation:-

- Random read operation performance decreases as we increase the concurrency.
- Performance of system is better with 1 thread than 2 thread concurrent execution.

5) Sequential Write Operation Latency:-



6) Sequential Write Operation Throughput:-



Observation:-

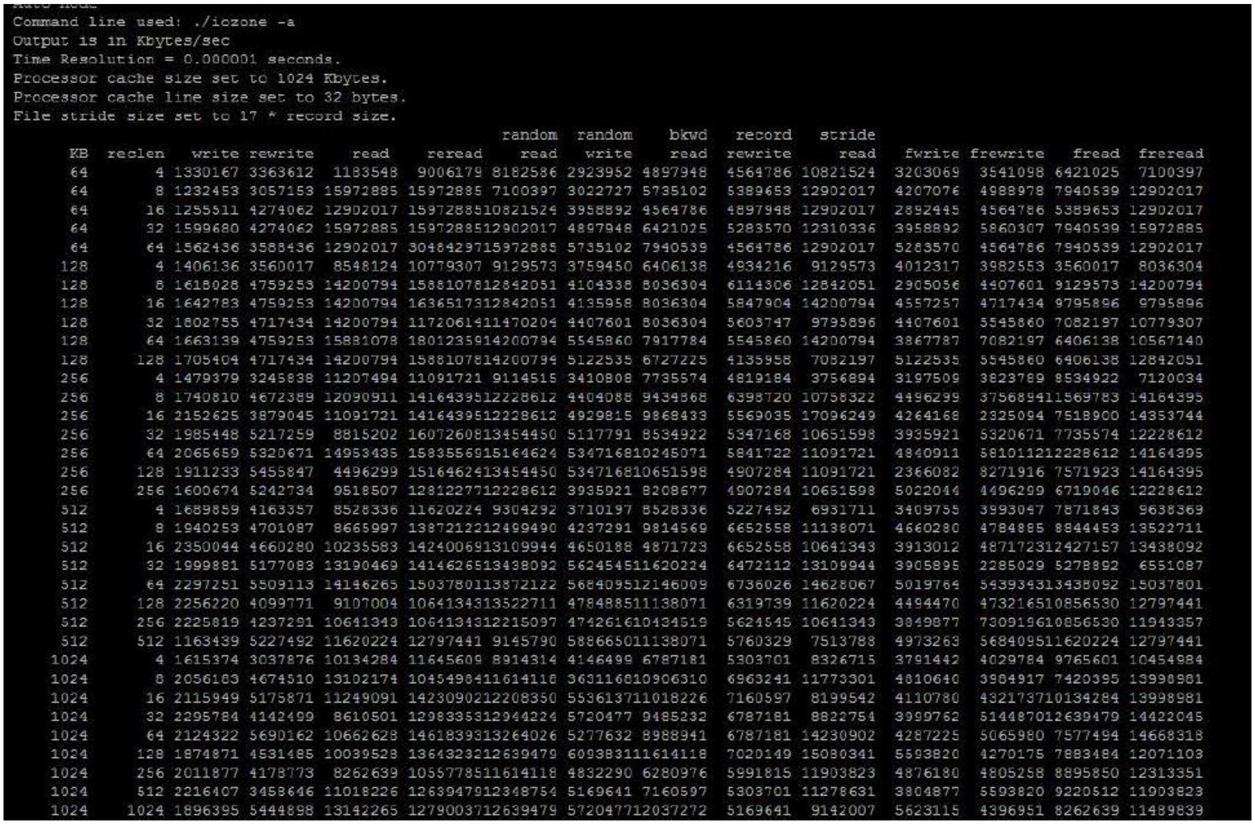
- There is increase in throughput as we increase the block-size from 1 Byte, 1 KB to 1MB.
- Throughput increases as we increase the concurrency and latency goes on decreasing.

Theoretical Performance of a disk:

Amazon has given t2.micro instance theoretical value for disk is given as 160Mb/S.

IOZONE Benchmarking system (Extra Credit)

- Compared evaluation of self-implemented benchmarking with IOZONE benchmarking.
- Both systems (mine & IOZONE) are evaluated on amazon's t2.micro instance.
- Imerf has been evaluated for packet size 100MBytes.



```
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    bkwd   record   stride
      KB  reclen    write rewrite     read    reread    read   write    read  rewrite     read   fwrite frewrite   fread  freread
      64       4 1330167 3363612  1183548   9006179 8182586 2923952 4897948  4564786 10821524  3203069  3541098 6421025  7100397
      64       8 1232453 3057153 15972885  15972885 7100397 3022727 5735102  5389653 12902017  4207076  4988978 7940539 12902017
      64      16 1255511 4274062 12902017  1597288510821524 3958892 4564786  4897948 12902017  2892445  4564786 5389653 12902017
      64      32 1599680 4274062 15972885  1597288512902017 4897948 6421025  5283570 12310336  3958892  5860307 7940539 15972885
      64      64 1562436 3588436 12902017  3048429715972885 5735102 7940539  4564786 12902017  5283570  4564786 7940539 12902017
     128       4 1406136 3560017  8548124  10779307 9129573 3759450 6406138  4934216  9129573  4012317  3982553 3560017  8036304
     128       8 1618028 4759253 14200794  1588107812842051 4104338 8036304  6114306 12842051  2905056  4407601 9129573 14200794
     128      16 1642783 4759253 14200794  1636517312842051 4135958 8036304  5847904 14200794  4557257  4717434 9795896  9795896
     128      32 1802755 4717434 14200794  1172061411470204 4407601 8036304  5603747  9795896  4407601  5545860 7082197 10779307
     128      64 1663139 4759253 15881078  1801235914200794 5545860 7917784  5545860 14200794  3867787  7082197 6406138 10567140
     128     128 1705404 4717434 14200794  1588107814200794 5122535 6727225  4135958  7082197  5122535  5545860 6406138 12842051
     256       4 1479379 3245838 11207494  11091721 9114515 3410808 7735574  4819184  3756894  3197509  3823789 8534922  7120034
     256       8 1740810 4672389 12090911  1416439512228612 4404088 9434868  6398720 10758322  4496299  375689411569783 14164395
     256      16 2152625 3879045 11091721  1416439512228612 4929815 9868433  5569035 17096249  4264168  2325094 7518900 14353744
     256      32 1985448 5217259  8815202  1607260813454450 5117791 8534922  5347168 10651598  3935921  5320671 7735574 12228612
     256      64 2065659 5320671 14953435  1583556915164624 534716810245071  5841722 11091721  4840911  581011212228612 14164395
     256     128 1911233 5455847  4496299  1516462413454450 534716810651598  4907284 11091721  2366082  8271916 7571923 14164395
     256     256 1600674 5242734  9518507  1281227712228612 3935921 8208677  4907284 10651598  5022044  4496299 6719046 12228612
     512       4 1689859 4163357  8528336  11620224 9304292 3710197 8528336  5227492  6931711  3409755  3993047 7871843  9638369
     512       8 1940253 4701087  8665997  1387212212499490 4237291 9814569  6652558 11138071  4660280  4784885 8844453 13522711
     512      16 2350044 4660280 10235583  1424006913109944 4650188 4871723  6652558 10641343  3913012  487172312427157 13438092
     512      32 1999881 5177083 13190469  1414626513438092 562454511620224  6472112 13109944  3905895  2285029 5278892  6551087
     512      64 2297251 5509113 14146265  1503780113872122 568409512146009  6736026 14628067  5019764  543934313438092 15037801
     512     128 2256220 4099771  9107004  1064134313522711 478488511138071  6319739 11620224  4494470  473216510856530 12797441
     512     256 2225819 4237291 10641343  1064134312215097 474261610434519  5624545 10641343  3849877  730919610856530 11943357
     512     512 1163439 5227492 11620224  12797441 9145790 588665011138071  5760329  7513788  4973263  568409511620224 12797441
    1024       4 1615374 3037876 10134284  11645609 8914314 4146499 6787181  5303701  8326715  3791442  4029784 9765601 10454984
    1024       8 2056183 4674510 13102174  1045498411614118 363116810906310  6963241 11773301  4810640  3984917 7420395 13998981
    1024      16 2115949 5175871 11249091  1423090212208350 553613711018226  7160597  8199542  4110780  432173710134284 13998981
    1024      32 2295784 4142499  8610501  1298335312944224 5720477 9485232  6787181  8822754  3999762  514487012639479 14422045
    1024      64 2124322 5690162 10662628  1461839313264026 5277632 8988941  6787181 14230902  4287225  5065980 7577494 14668318
    1024     128 1874871 4531485 10039528  1364323212639479 609383111614118  7020149 15080341  5593820  4270175 7883484 12071103
    1024     256 2011877 4178773  8262639  1055778511614118 4832290 6280976  5991815 11903823  4876180  4805258 8895850 12313351
    1024     512 2216407 3458646 11018226  1263947912348754 5169641 7160597  5303701 11278631  3804877  5593820 9220512 11903823
    1024    1024 1896395 5444898 13142265  1279003712639479 572047712037272  5169641  9142007  5623115  4396951 8262639 11489839
```

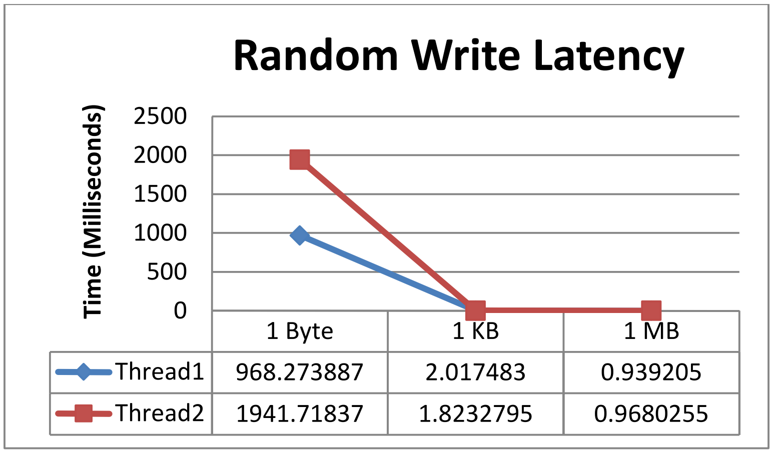
Following are the results for 1024KB :

Operations	Throughput in MB/S
Sequential Read	780.710970570172
Sequential Write	932.911097770482
Random Read	1385.6950542465
Random Write	932.911097770482

Comparing this values with my system values.

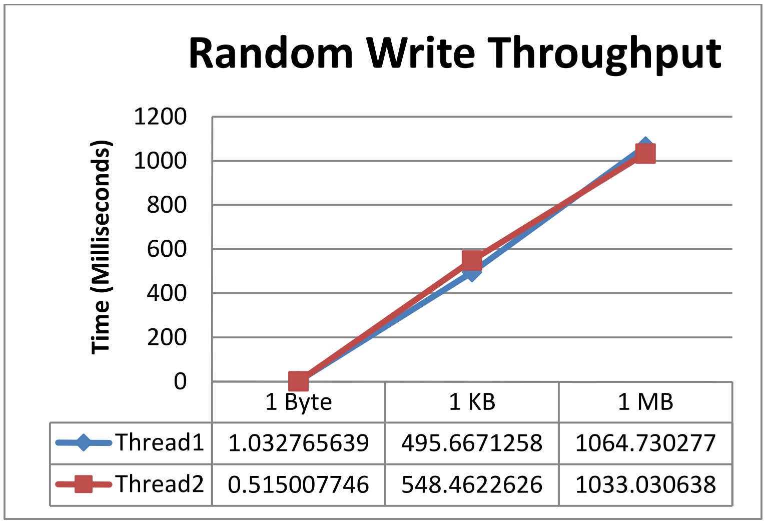
Operations	My System Throughput	IOZONE Throughput
Sequential Read	41234.38	14563456
Sequential Write	1350.3435	1347123
Random Read	1138.63	8765876
Random Write	78.77	4356754

7) **Random Write Latency:-**



	1 Byte	1 KB	1 MB
Thread1	968.273887	2.017483	0.939205
Thread2	1941.71837	1.8232795	0.9680255

8) **Random Write Throughput**

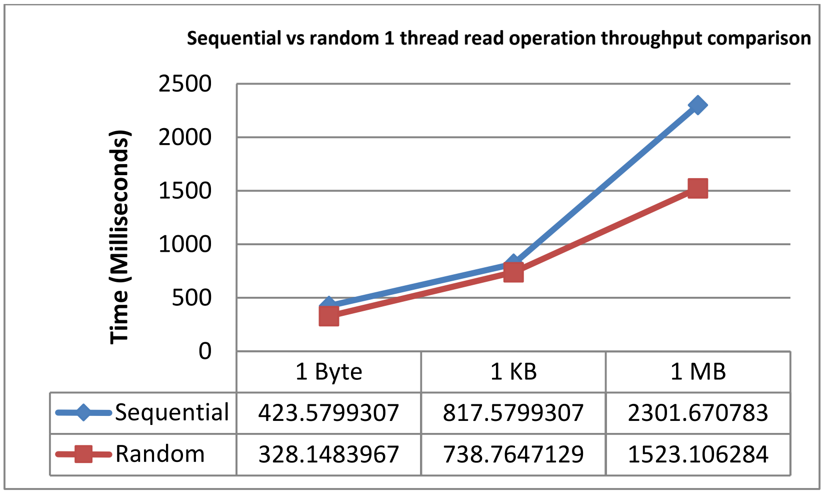


	1 Byte	1 KB	1 MB
Thread1	1.032765639	495.6671258	1064.730277
Thread2	0.515007746	548.4622626	1033.030638

Observation:-

- There is liner increase in throughput when there is increase in block-size.
- As we increase the concurrency performance of random write operation starts decreasing.

9) Sequential and random 1 thread read operation throughput comparison

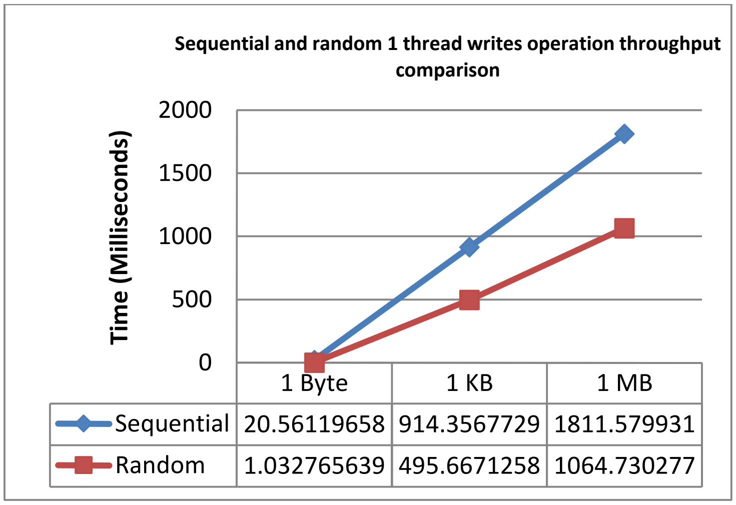


	1 Byte	1 KB	1 MB
◆ Sequential	423.5799307	817.5799307	2301.670783
■ Random	328.1483967	738.7647129	1523.106284

Observation:-

- Performance of sequential read operation is better than random read operation as we increase the block size.

10) Sequential and random 1 thread writes operation throughput comparison

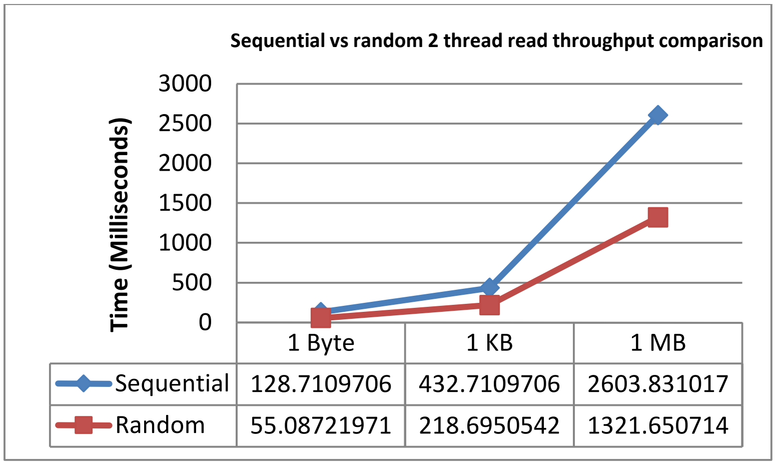


	1 Byte	1 KB	1 MB
◆ Sequential	20.56119658	914.3567729	1811.579931
■ Random	1.032765639	495.6671258	1064.730277

Observation:-

- Performance of sequential write operation is better than random write operation as we increase the block size.

11) Sequential vs random 2 thread read comparison:-

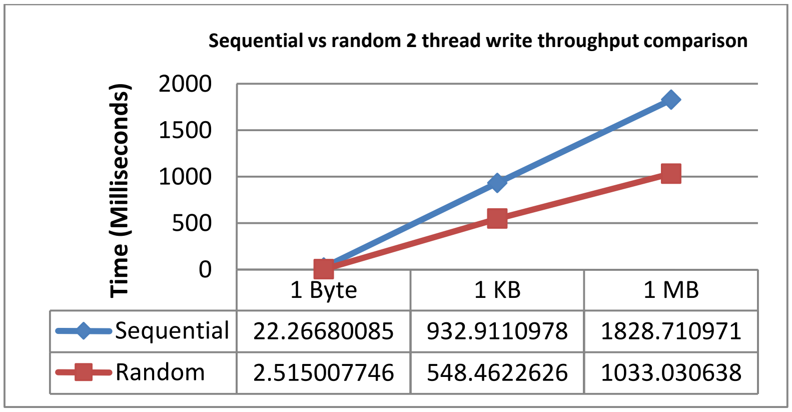


	1 Byte	1 KB	1 MB
Sequential	128.7109706	432.7109706	2603.831017
Random	55.08721971	218.6950542	1321.650714

Observation:-

- There is drastic increase in sequential read operation as we increase the concurrency on other hand throughput of random read operation is not as good as sequential read while we increase the concurrency.

12) Sequential vs random 2 thread write comparison



	1 Byte	1 KB	1 MB
Sequential	22.26680085	932.9110978	1828.710971
Random	2.515007746	548.4622626	1033.030638

Observation:-

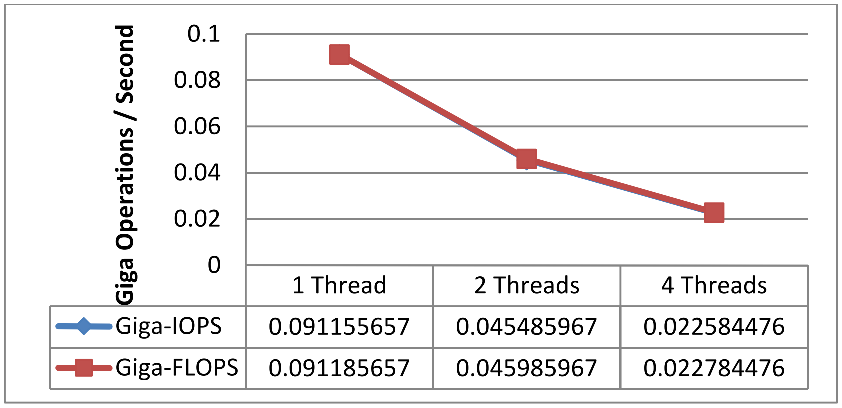
- Throughput of sequential write operation with 2 concurrent threads is better than random write operation.

5) CPU Benchmarking:-

1) CPU operations:-

Number of Threads	FLOPS	IOPS
1	10.966637	10.884979
2	21.745764	21.654305
4	43.889532	43.719291

2) CPU Benchmarking graph:-



	1 Thread	2 Threads	4 Threads
Giga-IOPS	0.091155657	0.045485967	0.022584476
Giga-FLOPS	0.091185657	0.045985967	0.022784476

X-axis : GFLOPS and GIOPS

Y-axis : Number of threads

- Above Table shows CPU speed in terms of GIGA FLOPS and GIGA IOPS i.e. Giga Floating Point Operation per Second and Giga Integer Operations per second respectively.

- This benchmark is performed in three different threads (1, 2 & 4).

- As more threads put more overhead of thread maintenance, concurrency and switching we can conclude that, the optimal number of concurrency for best performance is achieved using 4 threads.

- Theoretical Peak Performance = number of cores* clock cycle * FLOPs/cycle
  = 2*1.7*4
  = 13.6 GFLOPS

- Efficiency = (FLOPS for 1 thread /Theoretical Peak Performance)*100

  = (0.091185657/13.6)* 100

  = 0.6 %