

Performance Evaluation

1. CPU:

Floating point operations:

Observation 1:

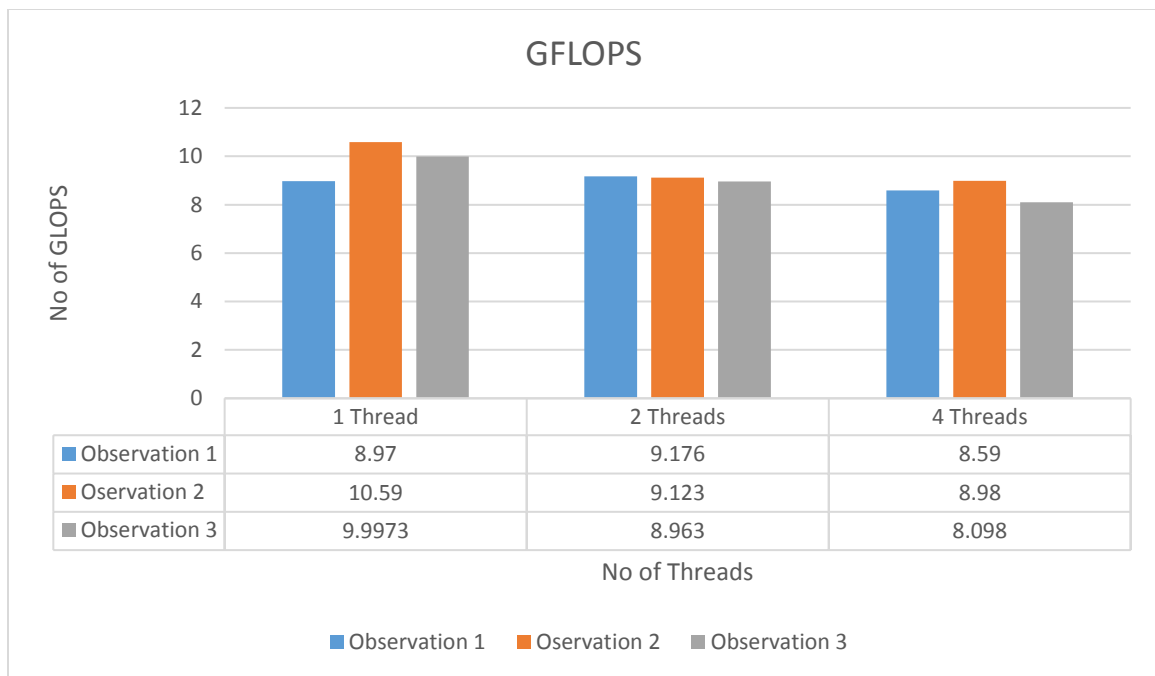
No of GFlops	No of threads
8.97	1
9.176	2
8.590	4

Observation 2:

No of GFlops	No of threads
10.59	1
9.123	2
8.98	4

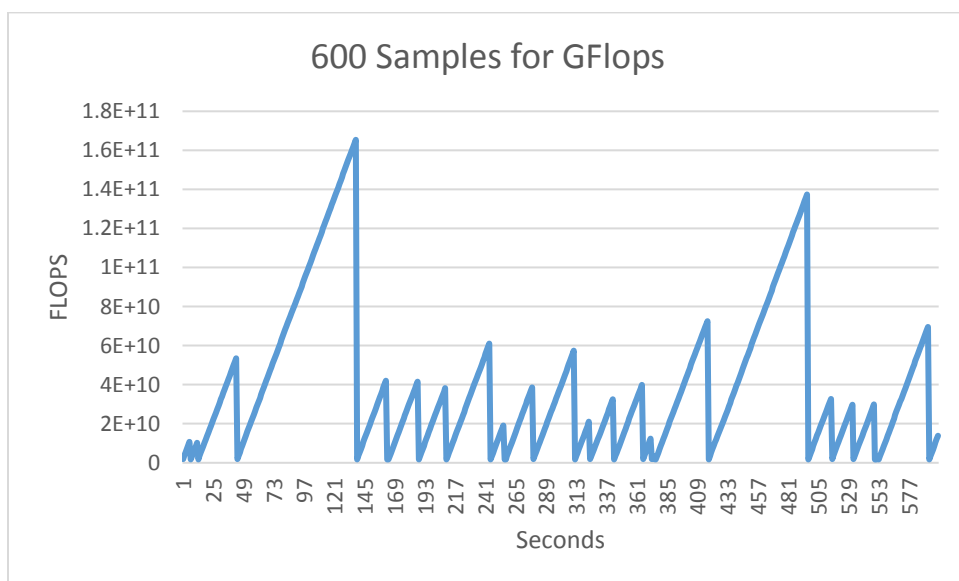
Observation 3:

No of GFlops	No of threads
9.997	1
8.96	2
8.098	4



Explanation:

Here graphs shows no of GFLOPs of CPU on y-axis and no of threads for calculating GFLOPs on X-axis. About 3 observations were carried out for measuring the no of GFLOPs of the CPU. Each observation by taken for each 1, 2 and 4 threads running concurrently. Here as it can be seen from graph the average GFLOPs remains around 9.12 with 1, 2 or 4 threads running concurrently. From the consistency of the system it can be said that the GFLOPs for Amazon EC2 t2.micro instance should be around 9-10.



Explanation:

Graph shows plot of Flops taken per second for a time period of 10 minutes.

Integer Operations:

Observation 1:

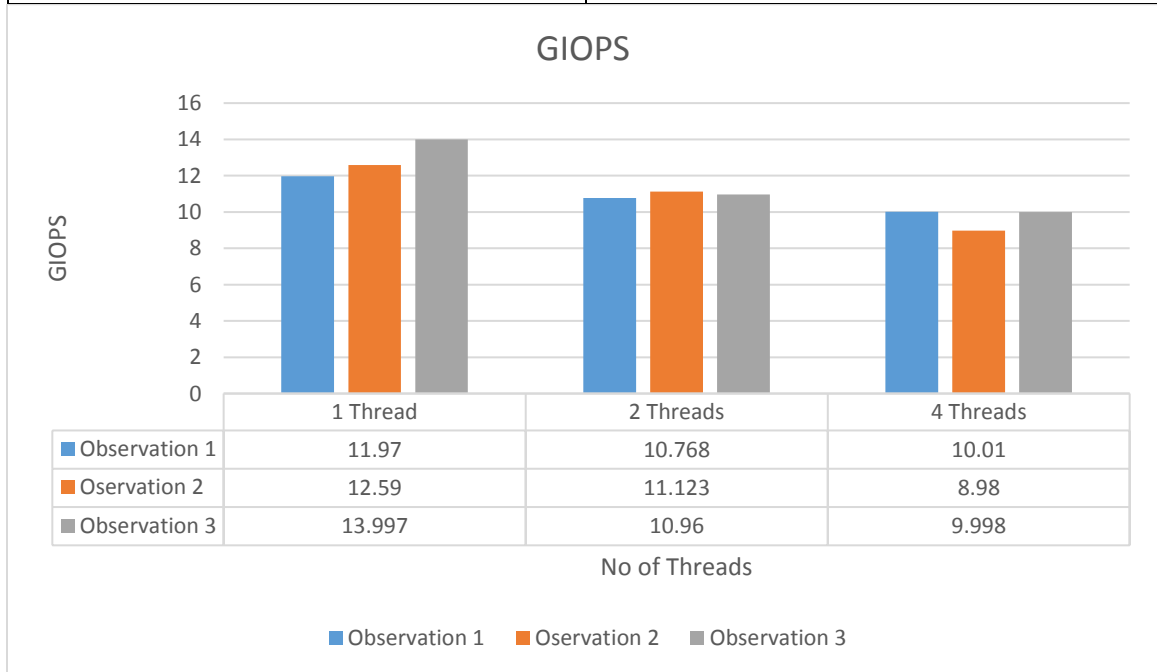
No of Glops	No of threads
11.97	1
10.768	2
10.010	4

Observation 2:

No of Glops	No of threads
12.59	1
11.123	2
8.98	4

Observation 3:

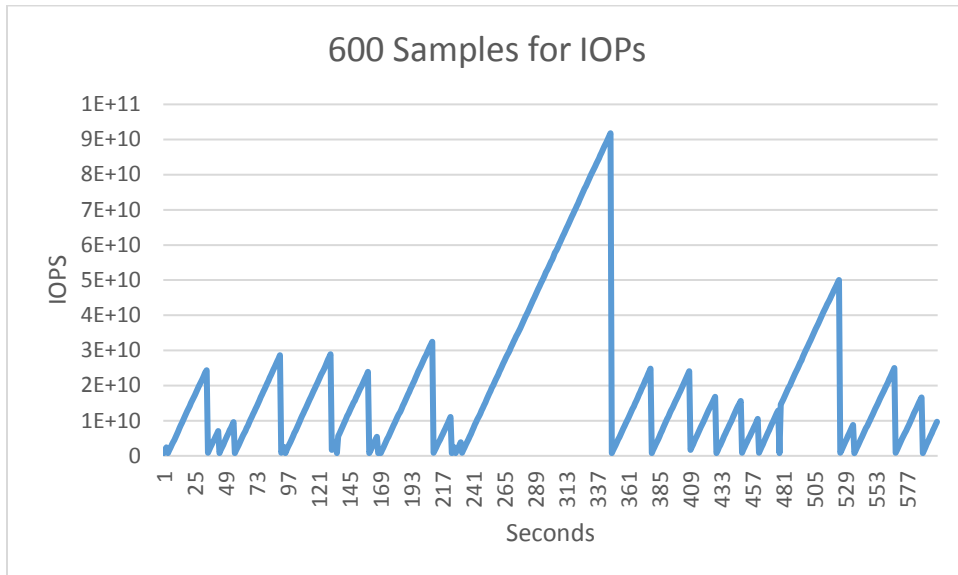
No of Glops	No of threads
13.997	1
10.96	2
9.998	4



Explanation:

Here graphs shows no of GIOPs of CPU on y-axis and no of threads for calculating GIOPs on X-axis. About 3 observations were carried out for measuring the no of GIOPs of the CPU. Each observation by taken for each 1, 2 and 4 threads running concurrently.

Here as it can be seen from graph the average GIOPs remains around 11 with 1, 2 or 4 threads running concurrently. From the consistency of the evaluation results it can be said that the GIOPs for the system should be around 11-12



Explanation:

Graph shows plot of lops taken per second for a time period of 10 minutes.

LINPACK:

```
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 18000
20000 22000 25000 26000 27000 30000 35000 40000 45000
Leading dimension of array : 1000 2000 5008 10000 15000 18000
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 =====

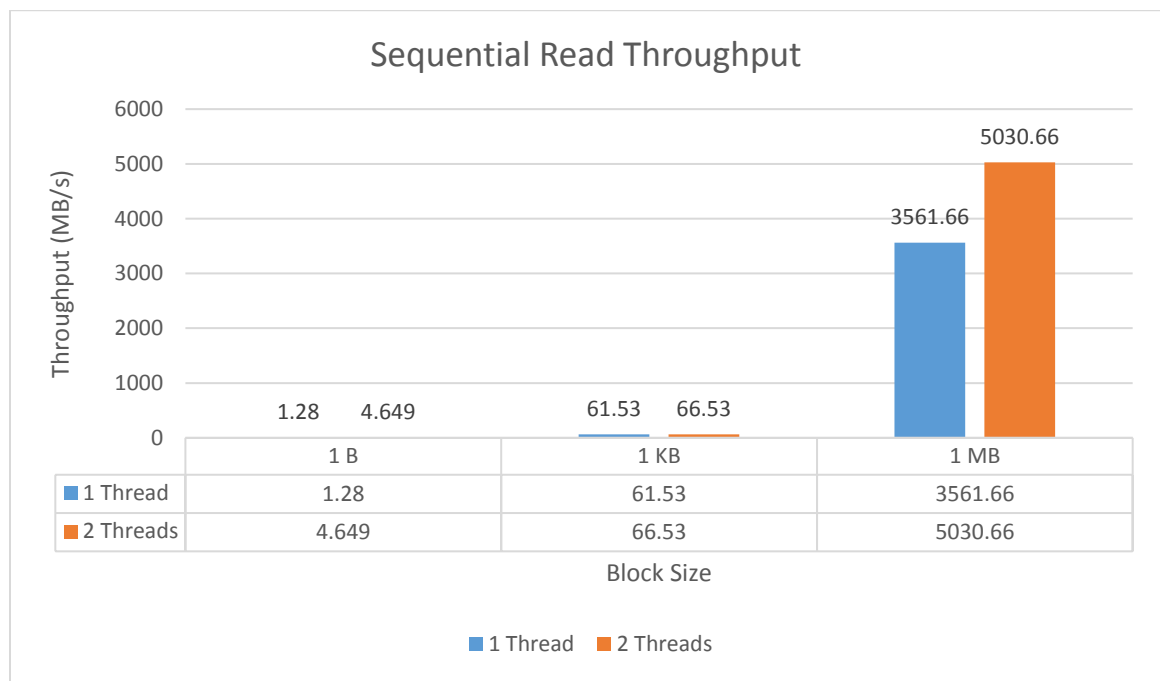
Size LDA Align. Time(s) GFlops Residual Residual(norm) Check
000 1000 4 0.025 26.2967 9.632295e-13 3.284860e-02 pass
000 1000 4 0.025 27.1659 9.632295e-13 3.284860e-02 pass
000 1000 4 0.025 27.2534 9.632295e-13 3.284860e-02 pass
000 1000 4 0.025 27.1335 9.632295e-13 3.284860e-02 pass
000 2000 4 0.187 28.5638 4.746648e-12 4.129002e-02 pass
000 2000 4 0.184 28.9604 4.746648e-12 4.129002e-02 pass
000 5008 4 2.458 33.9236 2.651185e-11 3.696863e-02 pass
000 5008 4 2.441 34.1558 2.651185e-11 3.696863e-02 pass
```

Explanations

2. DISK:

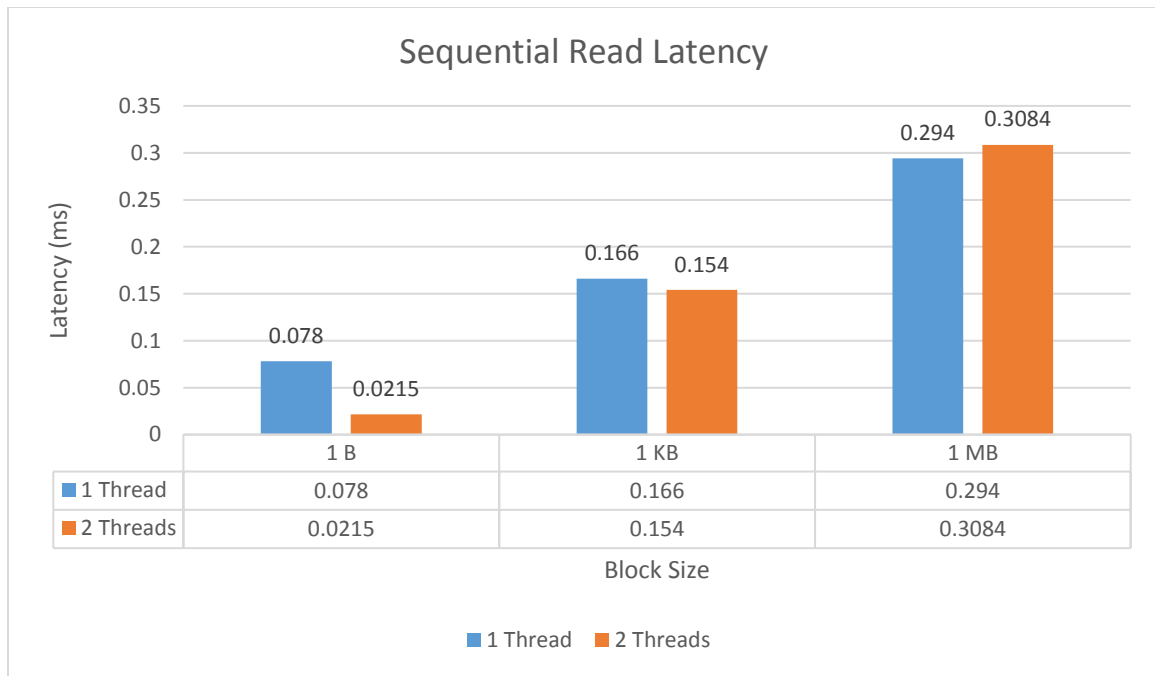
Sequential Read:

Data Size	Throughput (MB/s)	Latency (ms)	Threads
1 B	1.280	0.078	1
1 KB	61.53	0.166	1
1 MB	3561.66	0.294	1
1 B	4.649	0.0215	2
1 KB	66.53	0.154	2
1 MB	5030.66	0.3084	2



Explanation:

Here y-axis represents the throughput in MB/s and x-axis represents the Data size read sequentially. Throughput increases as size of data read increases. This is because throughput is calculated by data read/ time taken. So as data size increases throughput increases. Also when running 2 threads concurrently, throughput obtained is similar. Apart from this on comparing results with IOZONE read similar results were obtained.

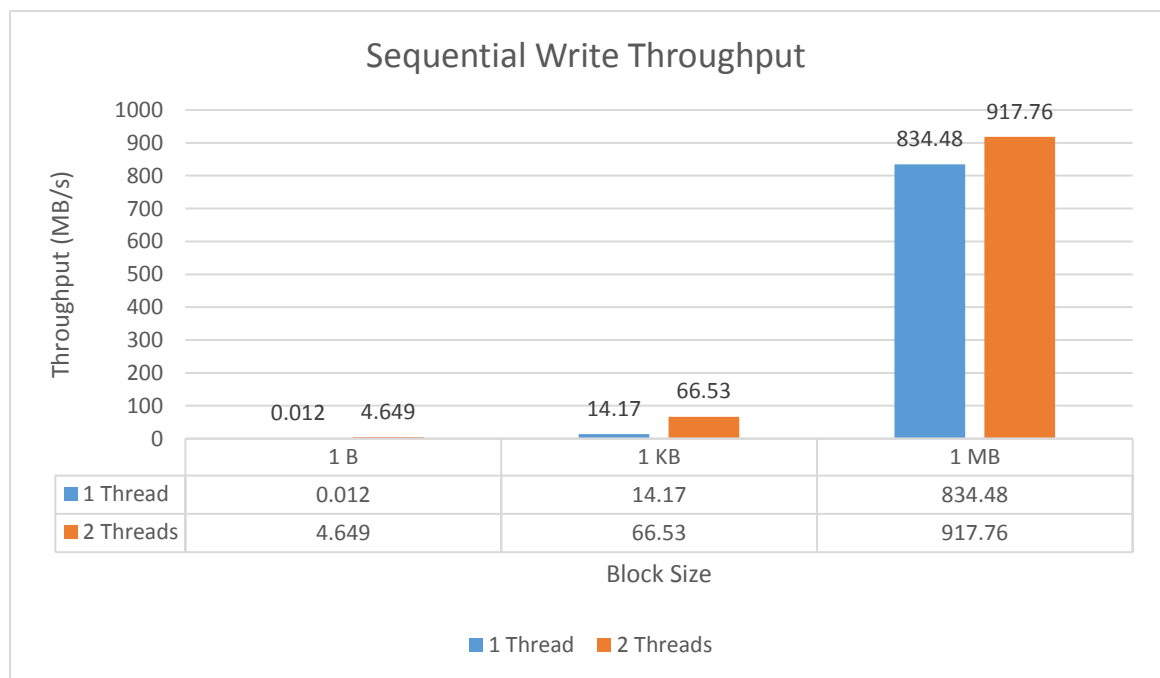


Explanation:

Here Y-axis represent Latency (ms) and X-axis represent data size that is being read sequentially. Latency is the time taken for the disk to read and reply back. So as data increases the latency increases. Hence the graph makes sense. Also on comparing the results with that of IOZONE, implied the correctness of the results.

Sequential Write:

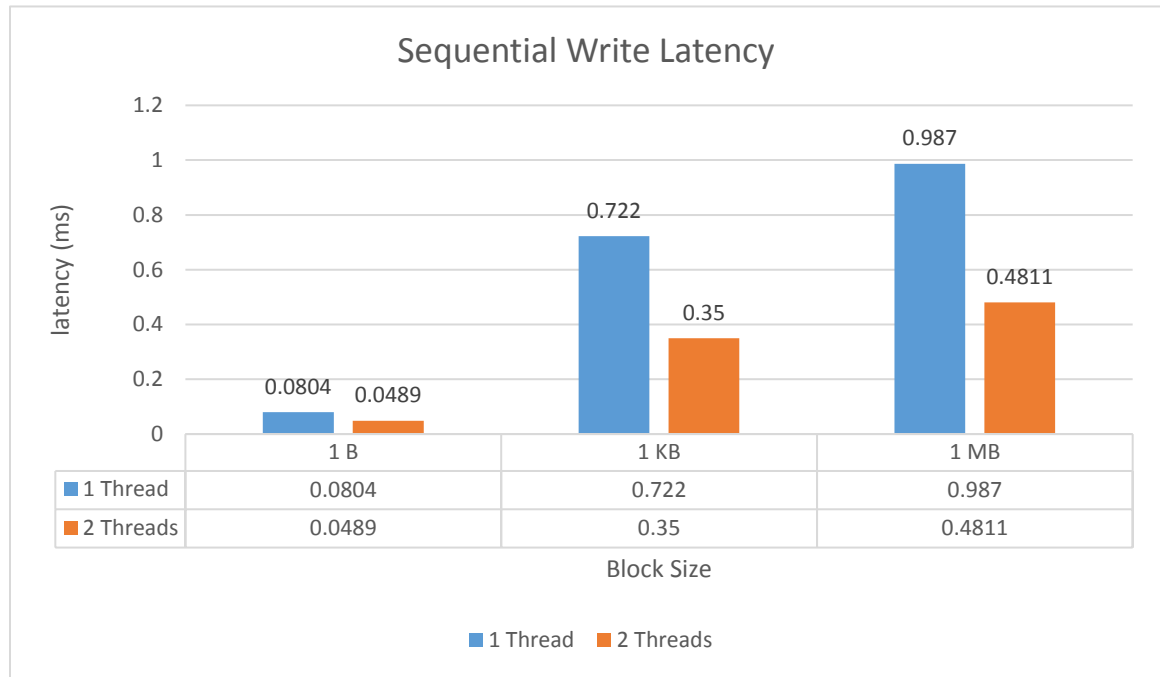
Data Size	Throughput (MB/s)	Latency (ms)	Threads
1 B	0.012	0.0804	1
1 KB	14.17	0.722	1
1 MB	834.48	0.987	1
1 B	4.649	0.0489	2
1 KB	66.53	0.350	2
1 MB	917.76	0.4811	2



Explanation:

Here y-axis represents the throughput in MB/s and x-axis represents the Data size written sequentially. Throughput increases as size of data read increases. This is because throughput is calculated by data read/ time taken. So as data size increases throughput increases. Also when running 2 threads concurrently, throughput obtained is similar.

Sequential write takes some time more than sequential read it is quite obvious and implies the correctness of the results. Apart from this on comparing results with IOZONE write similar results were obtained.

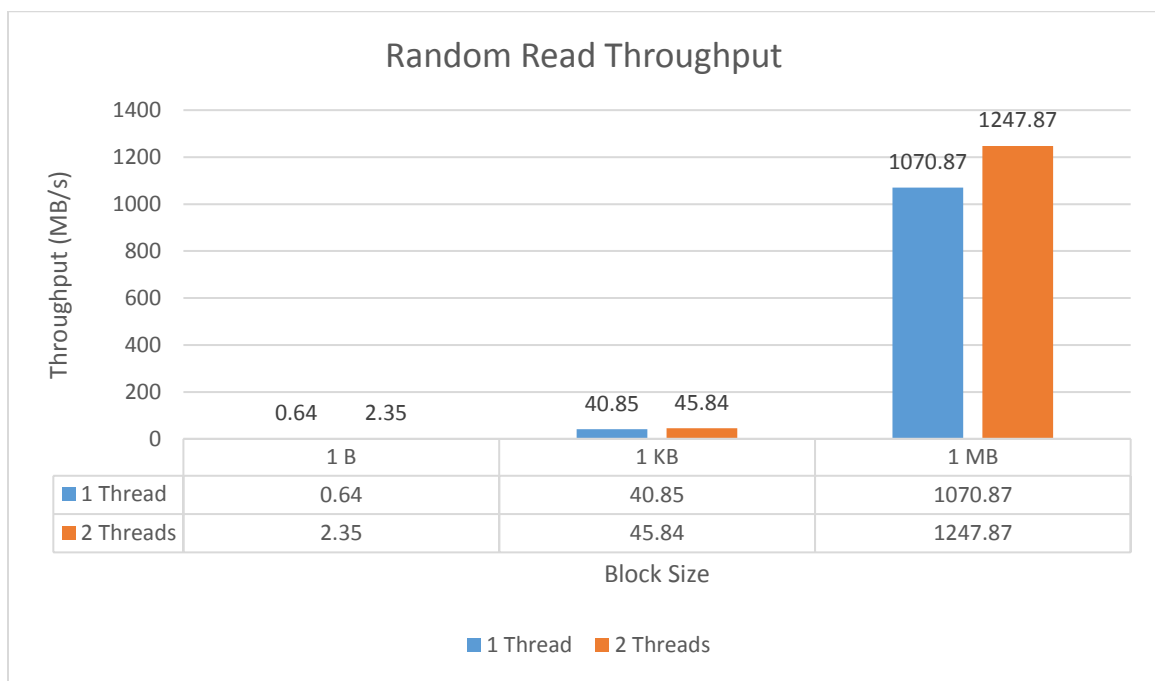


Explanation:

Here Y-axis represent Latency (ms) and X-axis represent data size that is being written sequentially. Latency is the time taken for the disk to read and reply back. So as data increases the latency increases. Hence the graph makes sense. Also on comparing the results with that of IOZONE, implied the correctness of the results.

Random Read:

Data Size	Throughput (MB/s)	Latency (ms)	Threads
1 B	0.64	0.098	1
1 KB	40.85	1.22	1
1 MB	1070.87	171.9	1
1 B	2.35	122.5	2
1 KB	45.84	21.4	2
1 MB	1247.87	193	2



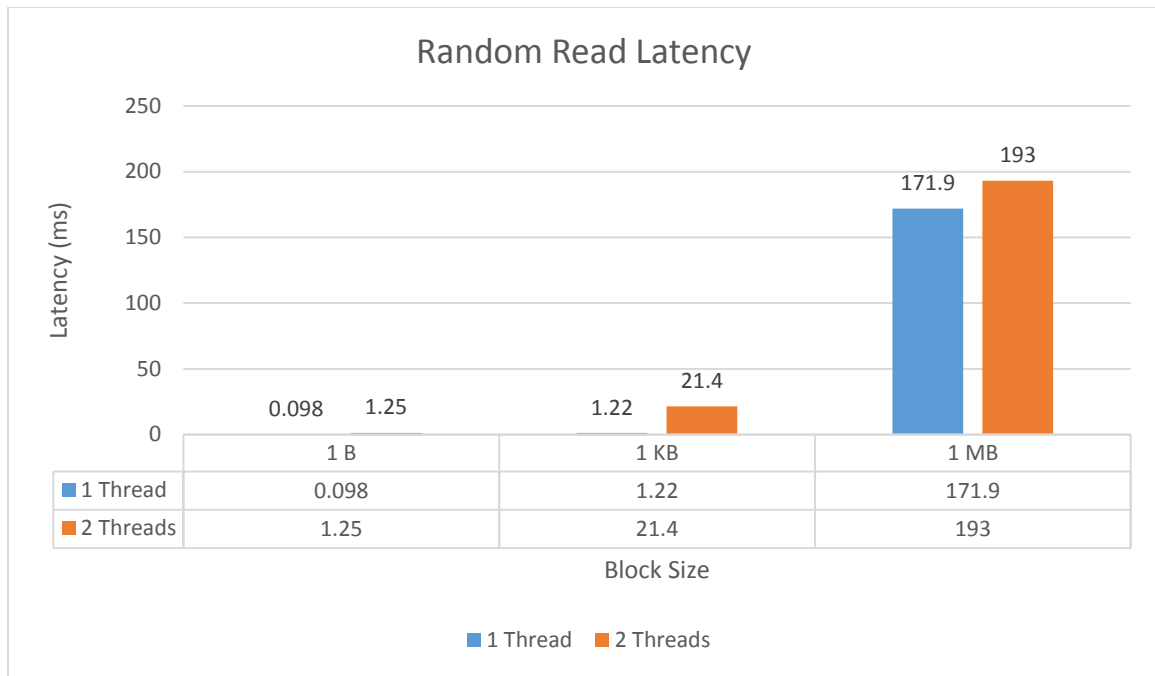
Explanation:

Here y-axis represents the throughput in MB/s and x-axis represents the Data size read randomly. Throughput increases as size of data read increases. This is because throughput is calculated by data read/ time taken. So as data size increases throughput increases.

Also when running 2 threads concurrently, throughput obtained is similar.

Apart from this on comparing results with IOZONE read similar results were obtained.

Random read takes some time more than sequential read. This is because in random read a Math.random() function is defined which defines the position from which to read the data. Hence, random read takes more time compared to sequential read and hence lower throughput.

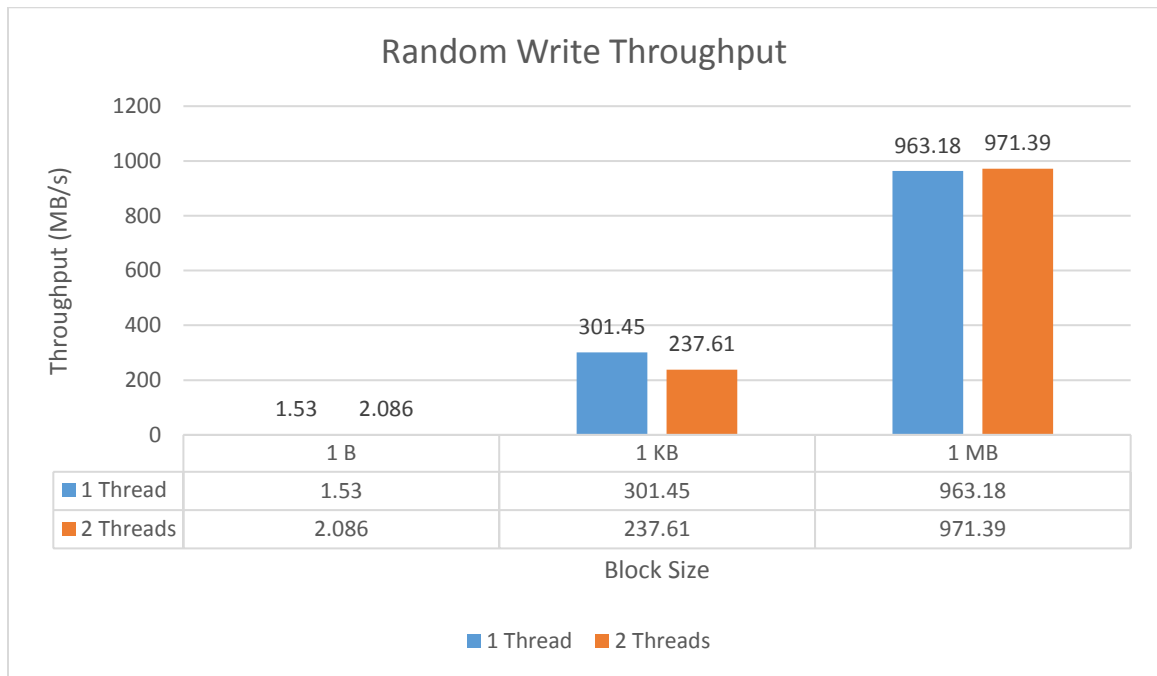


Explanation:

Here Y-axis represent Latency (ms) and X-axis represent data size that is being read randomly. Latency is the time taken for the disk to read and reply back. So as data increases the latency increases. Hence the graph makes sense. Also on comparing the results with that of IOZONE, implied the correctness of the results.

Random Write:

Data Size	Throughput (MB/s)	Latency (ms)	Threads
1 B	1.53	0.0565	1
1 KB	301.45	0.198	1
1 MB	963.18	162.9	1
1 B	2.086	180.7	2
1 KB	237.61	10.8	2
1 MB	971.39	190.2	2

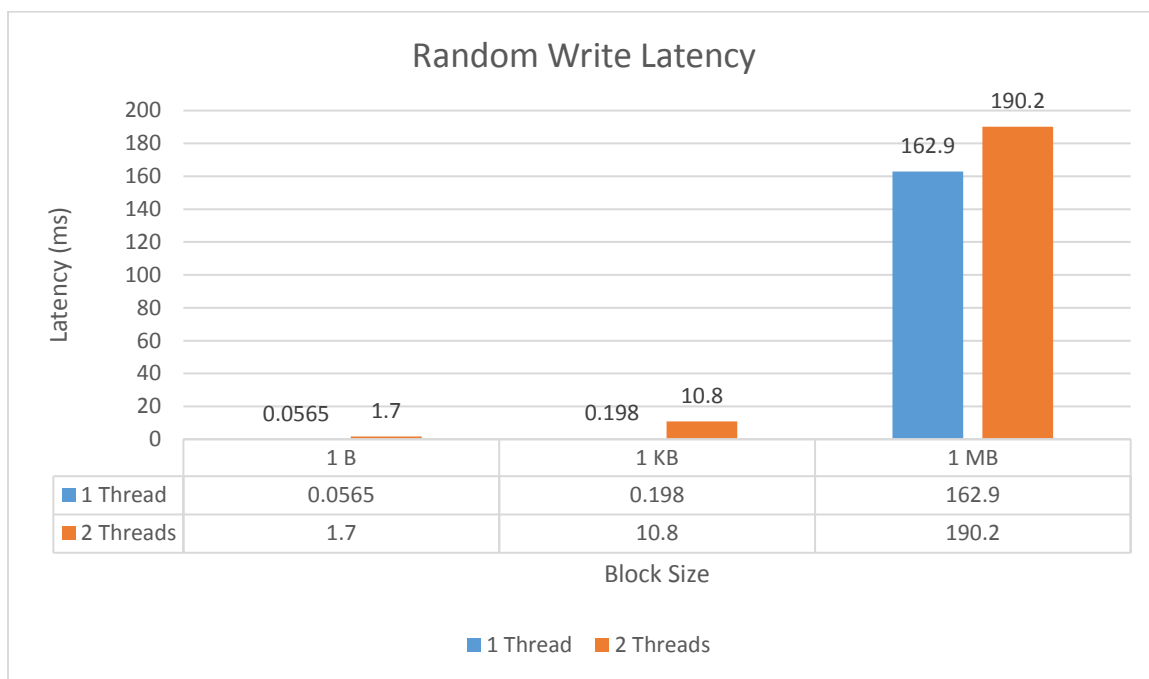


Explanation:

Here y-axis represents the throughput in MB/s and x-axis represents the Data size written randomly. Throughput increases as size of data read increases. This is because throughput is calculated by data read/ time taken. So as data size increases throughput increases.

Also when running 2 threads concurrently, throughput obtained is similar.

Apart from this on comparing results with IOZONE random write similar results were obtained. Random write takes some time more than sequential write. This is because in random read a Math.random() function is defined which defines the position from which to read the data. Hence, random write takes more time compared to sequential write and hence lower throughput.



Explanation:

Here Y-axis represent Latency (ms) and X-axis represent data size that is being written randomly. Latency is the time taken for the disk to read and reply back. So as data increases the latency increases. Hence the graph makes sense. Also on comparing the results with that of IOZONE, implied the correctness of the results.

IOZONE:

```
Ben England.
ubuntu@ip-172-31-28-125: ~/iozone3_394/src/current

Run began: Wed Feb 10 23:52:59 2016

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.

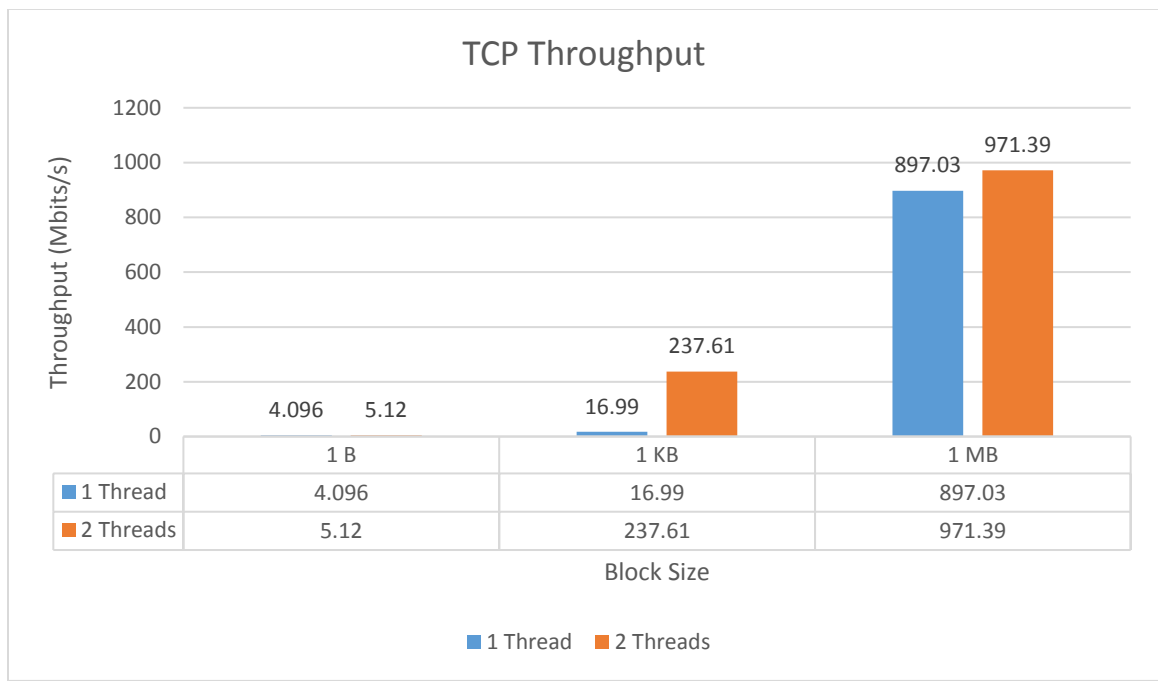
random random bk
wd record stride
ad rewrite KB reclen write rewrite read reread read write re
      read fwrite frewrite fread freread
97 4274062 10821524 3791156 3738358 6421025 10402178 12902017 7940539 3541098 71003
64 8 1828508 1947927 16983463 1597288512902017 4564786 71003
97 5283570 15972885 4897948 4564786 7940539 9006179
64 16 1828508 4564786 15972885 15972885 5283570 2133730 71003
97 5283570 6271021 2222043 4897948 9318832 9318832
64 32 1828508 2278628 15972885 7100397 6421025 2298136 79405
39 5389653 4274062 4274062 6421025 9006179 4018152
64 64 2067979 3791156 20962191 2096219112310336 4897948108215
24 5860307 9318832 4564786 4988978 7940539 20962191
128 4 1622919 3867787 11720614 11720614 8548124 3982553 70821
97 2286447 8548124 2058509 4135958 8548124 10779307
128 8 2027414 4934216 14200794 1588107811720614 4889281 85481
24 6406138 12842051 4135958 4444086 9977956 10567140
128 16 1755594 4889281 9129573 1588107814200794 5122535 79177
84 5603747 11720614 4012317 532579910567140 14200794
128 32 1939522 4759253 14200794 1588107812842051 5603747 91295
73 6045455 14200794 5122535 554586010567140 11720614
128 64 1997245 4934216 15881078 1588107812842051 5325799 85481
24 6114306 14200794 3867787 7582312 9795896 10779307
128 128 2238774 5122535 15881078 1863766414200794 5545860 91295
73 5545860 7582312 5379161 532579910779307 9795896
256 4 1752173 3879045 11569783 11695808 9192546 3823789 77917
08 5347168 8208677 4070199 4009406 9518507 7120034
256 8 2165650 4054829 14953435 1607260812812277 5022044106515
98 7073132 11207494 4819184 475515811569783 15164624
256 16 2613746 5217259 15164624 1607260811091721 5687020110917
21 3950402 11695808 4572895 511779112228612 13454450
256 32 2726577 5455847 13625180 1495343514164395 5810112107583
```

Figure shows the results obtained on running IOZONE on Amazon EC2 t2.micro instance.

3. Network:

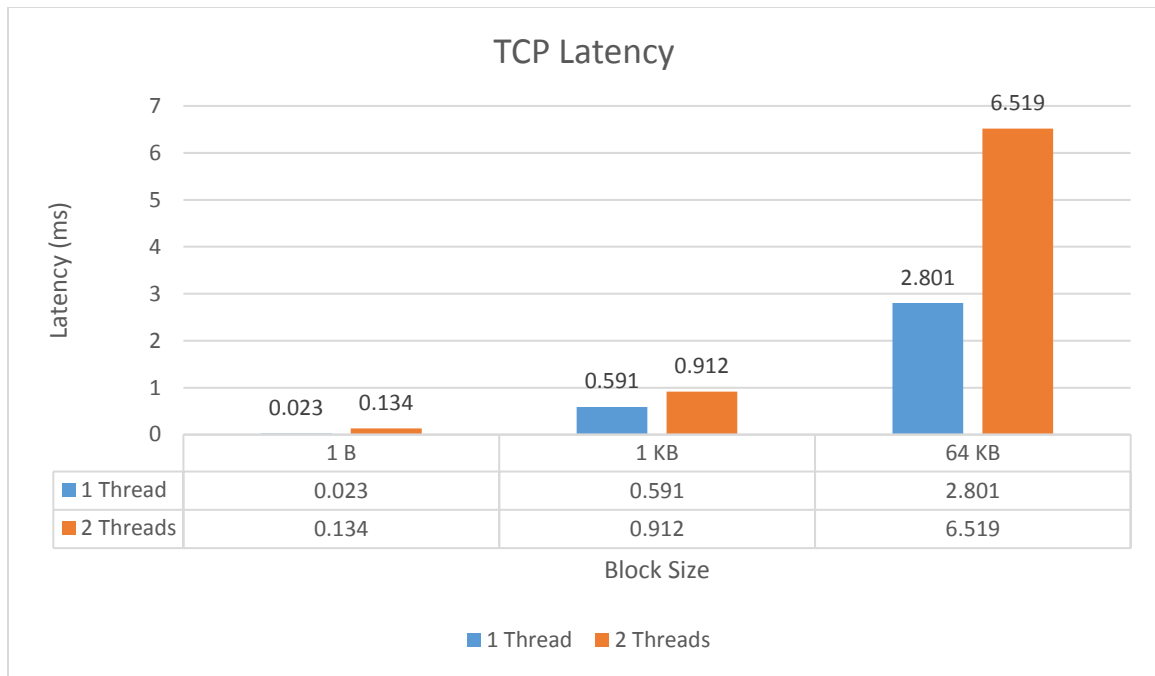
TCP:

Data Size	Throughput (Mbits/s)	Latency (ms)	Threads
1 B	4.096	0.023	1
1 KB	16.99	0.591	1
64 KB	897.03	2.801	1
1 B	5.12	0.134	2
1 KB	237.61	0.912	2
64 KB	971.39	6.519	2



Explanation:

Here y-axis represents the throughput in Mbits/s and x-axis represents the Data size sent and received. Throughput increases as size of data increases. This is because throughput is calculated by data / time taken. So as data size increases throughput increases. Also when running 2 threads concurrently, throughput obtained is similar. Apart from this on comparing results with Iperf for tcp similar results were obtained. the data.



Explanation:

Here Y-axis represent Latency (ms) and X-axis represent data size that is being sent and received. Latency is the time taken for the network to read and reply back. So as data increases the latency increases. Hence the graph makes sense. Also on comparing the results with that of Iperf tcp, implied the correctness of the results

Iperf:

```
ubuntu@ip-172-31-34-51: ~  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
ubuntu@ip-172-31-34-51:~$ sudo apt-get install iperf  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following NEW packages will be installed:  
  iperf  
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.  
Need to get 56.3 kB of archives.  
After this operation, 174 kB of additional disk space will be used.  
Get:1 http://us-west-2.ec2.archive.ubuntu.com/ubuntu/ trusty/universe iperf amd64  
  2.0.5-3 [56.3 kB]  
Fetched 56.3 kB in 0s (3,189 kB/s)  
Selecting previously unselected package iperf.  
(Reading database ... 51150 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-172-31-34-51:~$ iperf -s -i 1  
-----  
Server listening on TCP port 5001  
TCP window size: 85.3 KByte (default)  
-----  
[  4] local 172.31.34.51 port 5001 connected with 172.31.28.125 port 38882  
[ ID] Interval      Transfer    Bandwidth  
[  4] 0.0- 1.0 sec   119 MBytes  1.00 Gbits/sec  
[  4] 1.0- 2.0 sec   121 MBytes  1.01 Gbits/sec  
[  4] 2.0- 3.0 sec   120 MBytes  1.01 Gbits/sec  
[  4] 3.0- 4.0 sec   121 MBytes  1.01 Gbits/sec  
[  4] 4.0- 5.0 sec   120 MBytes  1.01 Gbits/sec  
[  4] 5.0- 6.0 sec   120 MBytes  1.01 Gbits/sec  
[  4] 6.0- 7.0 sec   120 MBytes  1.01 Gbits/sec  
[  4] 7.0- 8.0 sec   119 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  1000 Mbits/sec  
[  4] 0.0-10.0 sec   1.17 GBytes  1.01 Gbits/sec
```

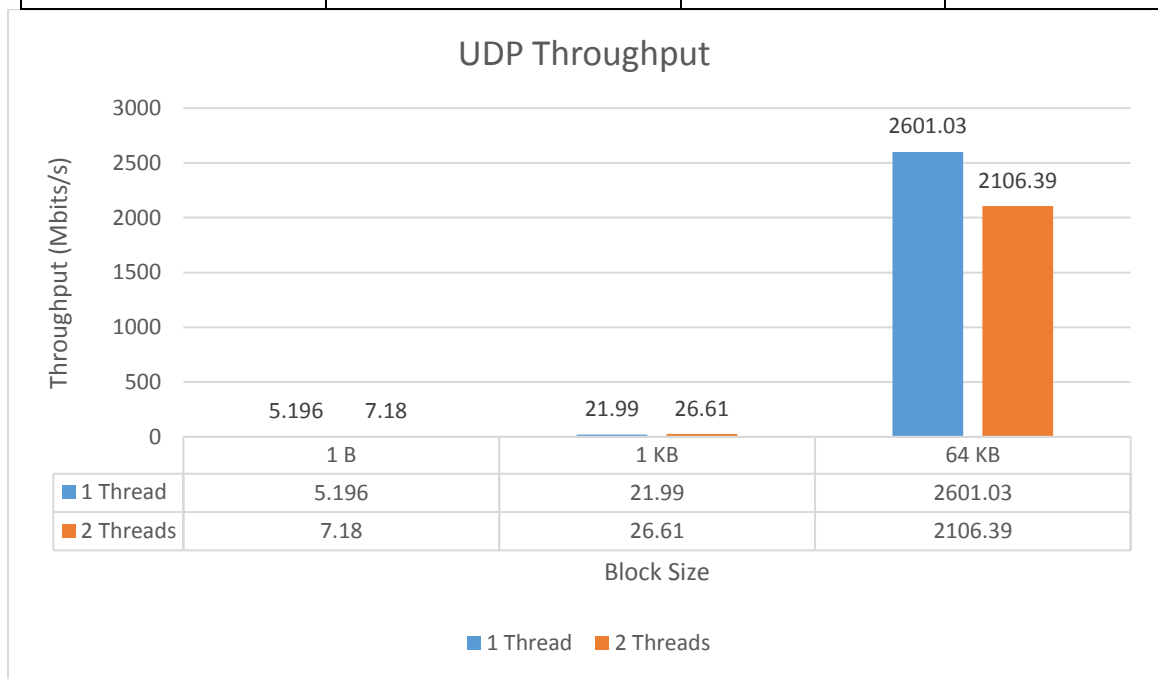
Iperf TCP Server

```
ubuntu@ip-172-31-28-125: ~  
ubuntu@ip-172-31-28-125:~/iozone3_394/src/current$ cd ..  
ubuntu@ip-172-31-28-125:~/iozone3_394/src$ cd ..  
ubuntu@ip-172-31-28-125:~/iozone3_394$ cd ..  
ubuntu@ip-172-31-28-125:~$ sudo apt-get install iperf  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
iperf is already the newest version.  
0 upgraded, 0 newly installed, 0 to remove and 37 not upgraded.  
ubuntu@ip-172-31-28-125:~$ iperf -o server ip -i 1  
error: Name or service not known  
ubuntu@ip-172-31-28-125:~$ iperf -c 172.31.28.125 -i 1  
connect failed: Connection refused  
ubuntu@ip-172-31-28-125:~$ iperf -c 172.31.36.162 -i 1  
connect failed: Connection timed out  
ubuntu@ip-172-31-28-125:~$  
ubuntu@ip-172-31-28-125:~$  
ubuntu@ip-172-31-28-125:~$  
ubuntu@ip-172-31-28-125:~$  
ubuntu@ip-172-31-28-125:~$ iperf -c 172.31.34.51 -i 1  
-----  
Client connecting to 172.31.34.51, TCP port 5001  
TCP window size: 325 KByte (default)  
-----  
[  3] local 172.31.28.125 port 38882 connected with 172.31.34.51 port 5001  
[ ID] Interval      Transfer    Bandwidth  
[  3] 0.0- 1.0 sec   122 MBytes  1.02 Gbits/sec  
[  3] 1.0- 2.0 sec   121 MBytes  1.02 Gbits/sec  
[  3] 2.0- 3.0 sec   120 MBytes  1.00 Gbits/sec  
[  3] 3.0- 4.0 sec   121 MBytes  1.02 Gbits/sec  
[  3] 4.0- 5.0 sec   120 MBytes  1.00 Gbits/sec  
[  3] 5.0- 6.0 sec   121 MBytes  1.01 Gbits/sec  
[  3] 6.0- 7.0 sec   120 MBytes  1.01 Gbits/sec  
[  3] 7.0- 8.0 sec   119 MBytes  1.00 Gbits/sec  
[  3] 8.0- 9.0 sec   119 MBytes  995 Mbits/sec  
[  3] 9.0-10.0 sec   120 MBytes  1.00 Gbits/sec  
[  3] 0.0-10.0 sec   1.17 GBytes  1.01 Gbits/sec  
ubuntu@ip-172-31-28-125:~$
```

Iperf TCP Client

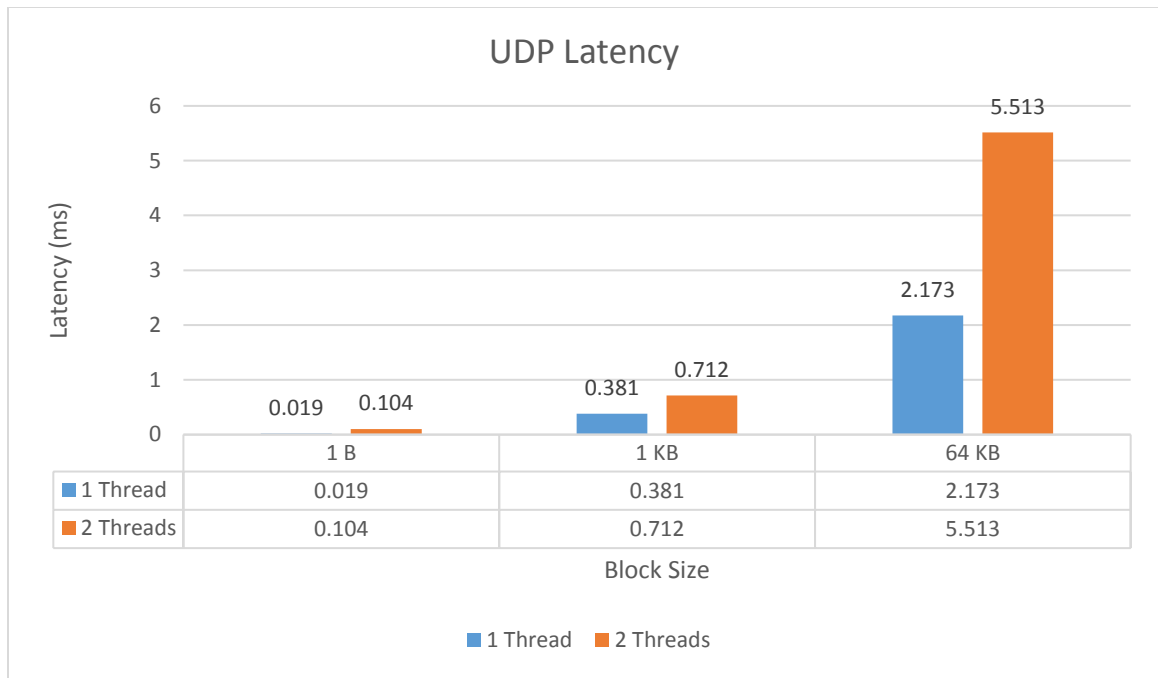
UDP:

Data Size	Throughput (Mbits/s)	Latency (ms)	Threads
1 B	5.196	0.019	1
1 KB	21.99	0.381	1
64 KB	2601.03	2.173	1
1 B	7.18	0.104	2
1 KB	26.61	0.712	2
64 KB	2106.39	5.513	2



Explanation:

Here y-axis represents the throughput in Mbits/s and x-axis represents the Data size sent and received. Throughput increases as size of data increases. This is because throughput is calculated by data / time taken. So as data size increases throughput increases. Also when running 2 threads concurrently, throughput obtained is similar. Apart from this on comparing results with Iperf for udp similar results were obtained. the data.



Explanation:

Here Y-axis represent Latency (ms) and X-axis represent data size that is being sent and received. Latency is the time taken for the network to read and reply back. So as data increases the latency increases. Hence the graph makes sense. Also on comparing the results with that of Iperf udp, implied the correctness of the results

IPerf UDP Server:

```
ubuntu@ip-172-31-28-125: ~  
Building dependency tree  
Reading state information... Done  
iperf is already the newest version.  
0 upgraded, 0 newly installed, 0 to remove and 37 not upgraded.  
ubuntu@ip-172-31-28-125:~$ iperf -s -i 1 -u  
-----  
Server listening on UDP port 5001  
Receiving 1470 byte datagrams  
UDP buffer size: 208 KByte (default)  
-----  
[ 3] local 172.31.28.125 port 5001 connected with 172.31.34.51 port 57083  
[ ID] Interval      Transfer    Bandwidth   Jitter     Lost/Total Datagrams  
[ 3] 0.0- 1.0 sec   128 KBytes  1.05 Mbits/sec  0.048 ms   0/ 89 (0%)  
[ 3] 1.0- 2.0 sec   128 KBytes  1.05 Mbits/sec  0.175 ms   0/ 89 (0%)  
[ 3] 2.0- 3.0 sec   128 KBytes  1.05 Mbits/sec  0.101 ms   0/ 89 (0%)  
[ 3] 3.0- 4.0 sec   128 KBytes  1.05 Mbits/sec  0.370 ms   0/ 89 (0%)  
[ 3] 4.0- 5.0 sec   128 KBytes  1.05 Mbits/sec  1.448 ms   0/ 89 (0%)  
[ 3] 5.0- 6.0 sec   129 KBytes  1.06 Mbits/sec  1.687 ms   0/ 90 (0%)  
[ 3] 6.0- 7.0 sec   128 KBytes  1.05 Mbits/sec  0.096 ms   0/ 89 (0%)  
[ 3] 7.0- 8.0 sec   128 KBytes  1.05 Mbits/sec  0.177 ms   0/ 89 (0%)  
[ 3] 8.0- 9.0 sec   128 KBytes  1.05 Mbits/sec  0.053 ms   0/ 89 (0%)  
[ 3] 9.0-10.0 sec   128 KBytes  1.05 Mbits/sec  0.441 ms   0/ 89 (0%)  
[ 3] 0.0-10.0 sec   1.25 MBytes  1.05 Mbits/sec  0.696 ms   0/ 893 (0%)
```

IPerf udp client

```
ubuntu@ip-172-31-34-51: ~/SourceCode
0 upgraded, 0 newly installed, 0 to remove and 61 not upgraded.
ubuntu@ip-172-31-34-51:~/SourceCode$ iperf -c 172.31.28.125 -i 1 -u
-----
Client connecting to 172.31.28.125, UDP port 5001
Sending 1470 byte datagrams
UDP buffer size: 208 KByte (default)
-----
[  3] local 172.31.34.51 port 57083 connected with 172.31.28.125 port 5001
[ ID] Interval           Transfer     Bandwidth
[  3] 0.0- 1.0 sec      129 KBytes  1.06 Mbits/sec
[  3] 1.0- 2.0 sec      128 KBytes  1.05 Mbits/sec
[  3] 2.0- 3.0 sec      128 KBytes  1.05 Mbits/sec
[  3] 3.0- 4.0 sec      128 KBytes  1.05 Mbits/sec
[  3] 4.0- 5.0 sec      128 KBytes  1.05 Mbits/sec
[  3] 5.0- 6.0 sec      128 KBytes  1.05 Mbits/sec
[  3] 6.0- 7.0 sec      129 KBytes  1.06 Mbits/sec
[  3] 7.0- 8.0 sec      128 KBytes  1.05 Mbits/sec
[  3] 8.0- 9.0 sec      128 KBytes  1.05 Mbits/sec
[  3] 9.0-10.0 sec      128 KBytes  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    0.696 ms    0/ 893 (0%)
ubuntu@ip-172-31-34-51:~/SourceCode$
```