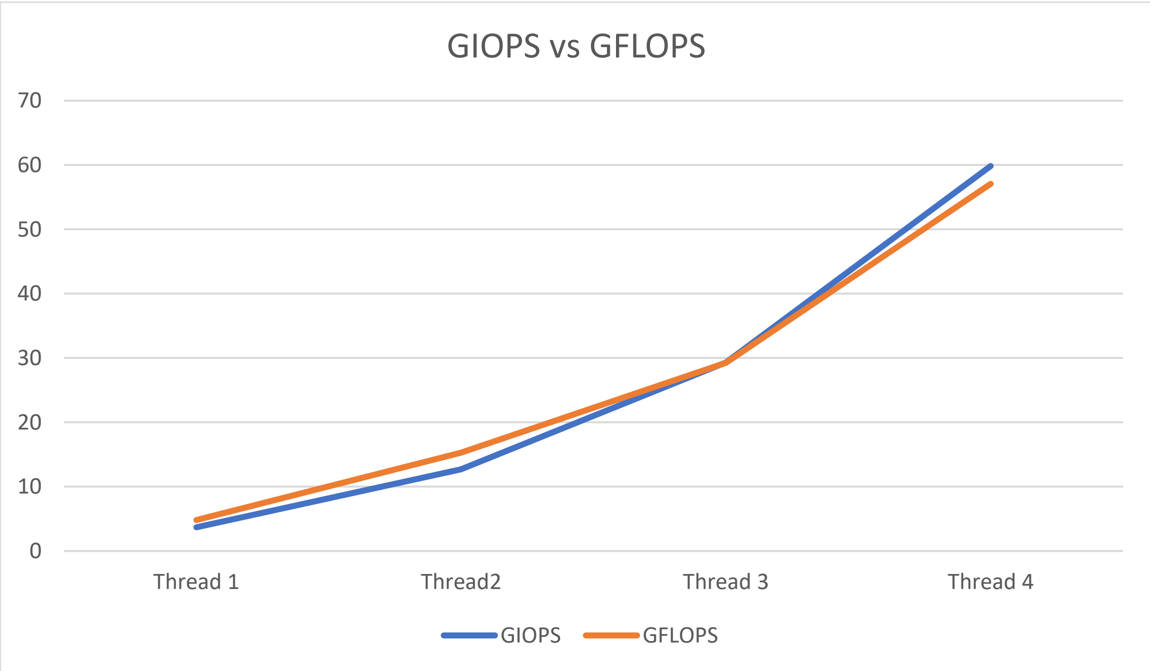
Evaluation Report

CPU section:

Here, the program has been designed to calculate the GIOPS and GFLOPS based on the number of threads provided to the program , here we have tried to achieve strong scaling by dividing the number of tasks between the threads.



x-axis is thread count and y-axis is operation.

Operation	Threads	Operation value(in GIOPS or GFLOPS)
GIOPS	1	3.692813
GIOPS	2	12.6971
GIOPS	4	29.33815
GIOPS	8	59.85392

GFLOPS	1	4.8198
GFLOPS	2	15.29461
GFLOPS	4	29.25313
GFLOPS	8	57.08323

Linpack Benchmark:

```
[cc@pa1-megh ~]$ ./xlinpack_xeon64
Input data or print help ? Type [data]/help :
1000
Number of equations to solve (problem size): 2000
Leading dimension of array: 15
Warning: incorrect parameter Leading dimension of array (2000),
must be not less than (2000),
set to default value (2000).
Number of trials to run: 20
Data alignment value (in Kbytes): 10000
Current date/time: Tue Oct 10 17:34:55 2017

CPU frequency:    2.895 GHz
Number of CPUs: 2
Number of cores: 2
Number of threads: 2

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                     : 20
Data alignment value (in Kbytes)            : 10000

Maximum memory requested that can be used = 42280000, at the size = 2000

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

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm)
2000   2000   10000    0.163      32.7485  4.298950e-12 3.739560e-02
2000   2000   10000    0.162      32.9755  4.298950e-12 3.739560e-02
2000   2000   10000    0.159      33.5981  4.298950e-12 3.739560e-02
2000   2000   10000    0.154      34.6323  4.298950e-12 3.739560e-02
2000   2000   10000    0.164      32.6598  4.298950e-12 3.739560e-02
2000   2000   10000    0.157      34.0875  4.298950e-12 3.739560e-02
2000   2000   10000    0.157      33.9869  4.298950e-12 3.739560e-02
2000   2000   10000    0.157      34.1062  4.298950e-12 3.739560e-02
2000   2000   10000    0.157      33.9820  4.298950e-12 3.739560e-02
2000   2000   10000    0.157      34.0467  4.298950e-12 3.739560e-02
2000   2000   10000    0.177      30.2357  4.298950e-12 3.739560e-02
2000   2000   10000    0.144      37.1347  4.298950e-12 3.739560e-02
2000   2000   10000    0.143      37.3431  4.298950e-12 3.739560e-02
2000   2000   10000    0.143      37.2725  4.298950e-12 3.739560e-02
2000   2000   10000    0.143      37.3402  4.298950e-12 3.739560e-02
2000   2000   10000    0.142      37.5019  4.298950e-12 3.739560e-02
2000   2000   10000    0.205      26.0651  4.298950e-12 3.739560e-02
2000   2000   10000    0.156      34.1940  4.298950e-12 3.739560e-02
2000   2000   10000    0.148      36.2119  4.298950e-12 3.739560e-02
2000   2000   10000    0.151      35.3217  4.298950e-12 3.739560e-02

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
2000   2000   10000     34.2722  37.5019

End of tests
```

Theoretical performance: of Chameleon:

2.29 Ghz * 2core *2CPU*8= 73.28
Practical value : 14.771185 GFLOPS
Efficiency: Practical/ Theoretical= (14.71185/73.28)*100 = 20.15%

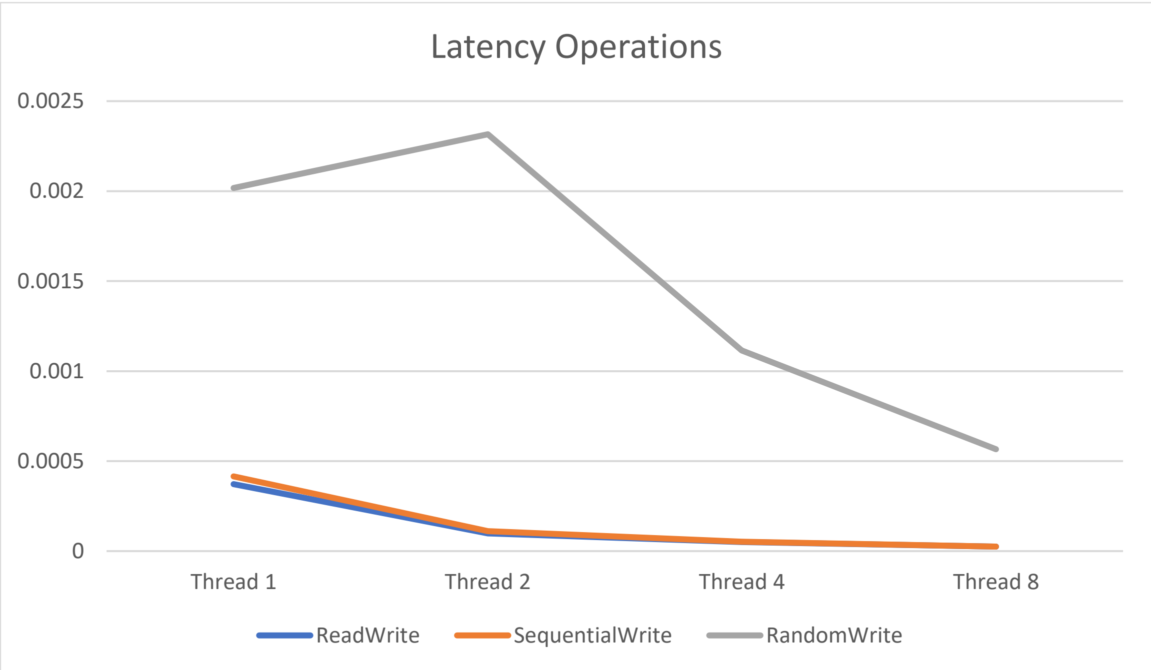
Conclusion:

1. We see the increase in GIOPS and GFLOPS values with increase in number of threads.
2. Increasing the scale of the number of operations to be performed increases the GIOPS and GFLOPS values.
3. Various experiments can be performed by scaling up the number of operations and trying to increase the efficiency further.

Memory section:

The program was designed to determine the performance of the memory.

Latency and Throughput measure the memory speed. Latency is measured in ms/bit and Throughput is measured in MB/sec. The current program is built to achieve strong scaling by the dividing the current fixed operation task among the number of threads generated.

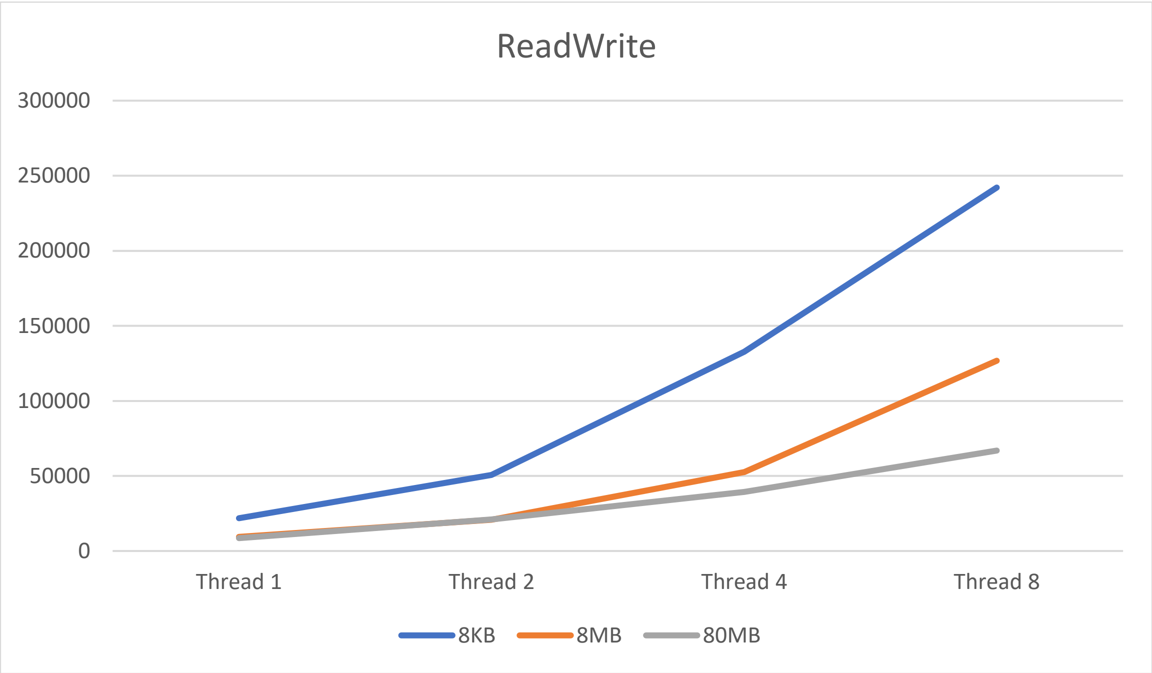


The thread count is taken in x-axis and Latency is taken in Y-axis for all 3 operations (1. ReadWrite, 2-SequentialWrite, 3- RandomWrite)

Operation	Memory Size	No. of Threads	Latency(in ms/bit)
ReadWrite	8B	1	0.000372
ReadWrite	8B	2	0.000098
ReadWrite	8B	4	0.00005

ReadWrite	8B	8	0.000025
SequentialWrite	8B	1	0.000415
SequentialWrite	8B	2	0.00011
SequentialWrite	8B	4	0.000053
SequentialWrite	8B	8	0.000025
RandomWrite	8B	1	0.002017
RandomWrite	8B	2	0.002316
RandomWrite	8B	4	0.001114
RandomWrite	8B	8	0.000566

Below, we have the chart illustrating the performance of the threads for the different blocksizes in ReadWrite task.

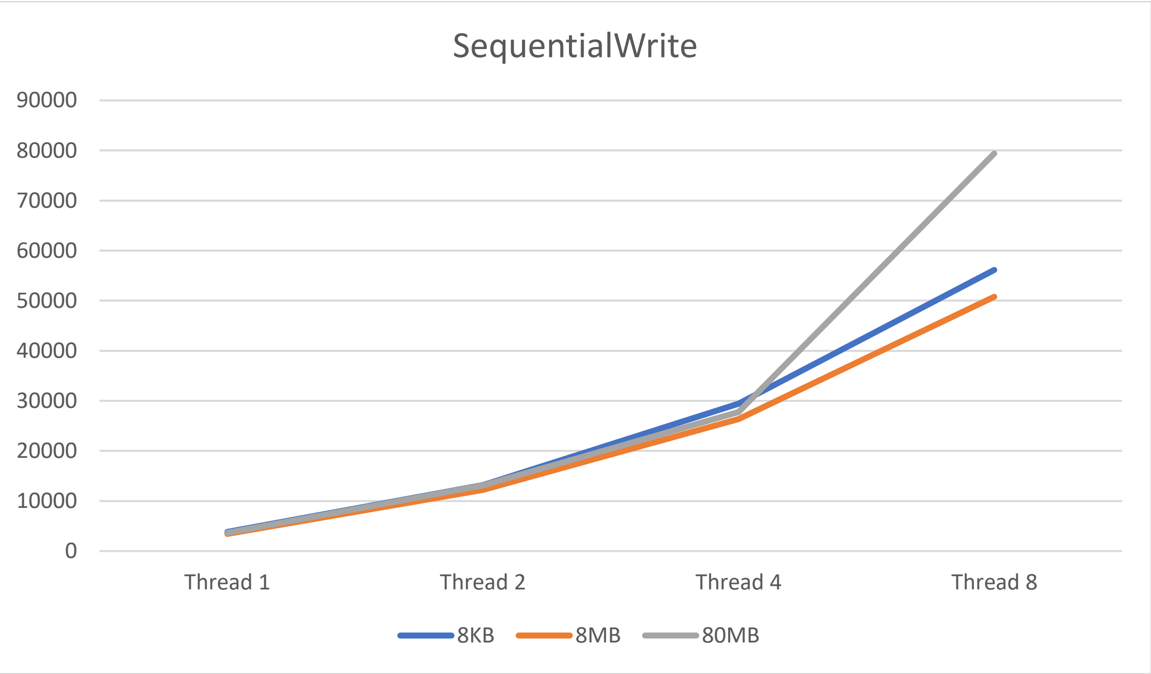


The threadcount is the x-axis and Throughtput in MB/sec is the y axis. Here only the readwrite scenario is considered.

Operation	Memory Size	Threads	Throughput(in MB/sec)

ReadWrite	8KB	1	21843.938
ReadWrite	8KB	2	50674.25
ReadWrite	8KB	4	132840.375
ReadWrite	8KB	8	242130.469
ReadWrite	8MB	1	9415.398
ReadWrite	8MB	2	20745.123
ReadWrite	8MB	4	52667.441
ReadWrite	8MB	8	126754.25
ReadWrite	80MB	1	8567.604
ReadWrite	80MB	2	21068.008
ReadWrite	80MB	4	39436.566
ReadWrite	80MB	8	66897.5

Below, we have the chart illustrating the performance of the threads for the different blocksizes in SequentialWrite task.

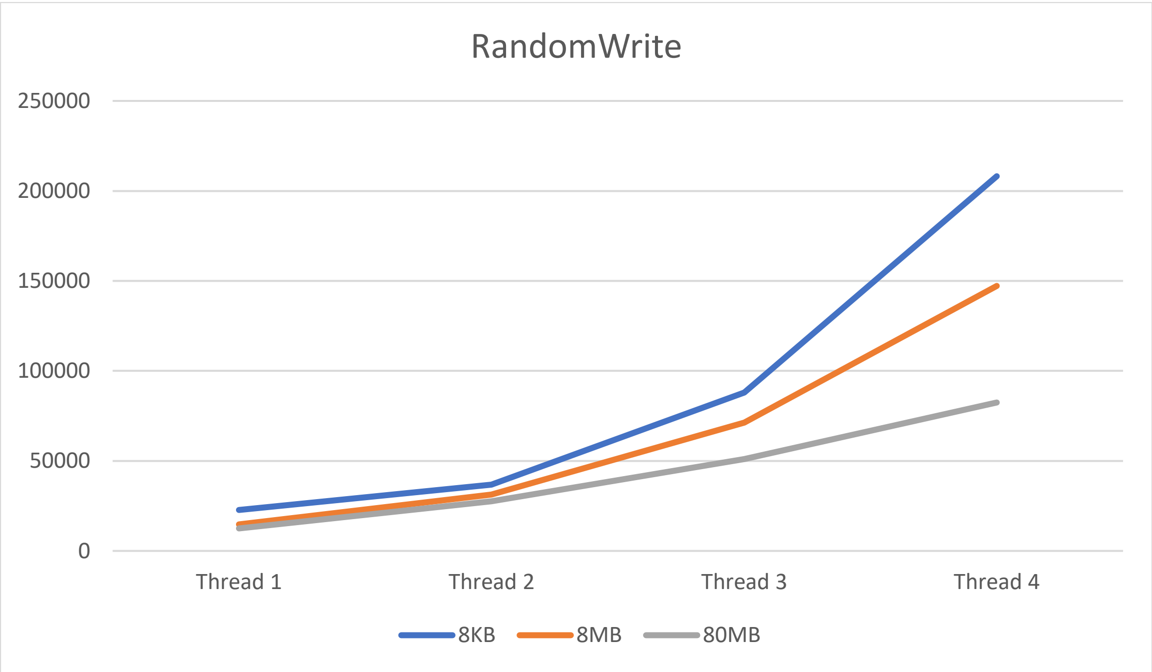


The threadcount is the x-axis and Throughtput in MB/sec is the y axis. Here only the sequentialwrite scenario is considered.

Operation	Memory Size	Threads	Throughput (in MB/sec)
SequentialWrite	8KB	1	3824.22

SequentialWrite	8KB	2	13206.854
SequentialWrite	8KB	4	29461.268
SequentialWrite	8KB	8	56145.355
SequentialWrite	8MB	1	3431.176
SequentialWrite	8MB	2	12215.052
SequentialWrite	8MB	4	26368.135
SequentialWrite	8MB	8	50786.094
SequentialWrite	80MB	1	3646.243
SequentialWrite	80MB	2	13147.168
SequentialWrite	80MB	4	27767.611
SequentialWrite	80MB	8	79397.539

Below, we have the chart illustrating the performance of the threads for the different blocksizes in RandomWrite task.



The threadcount is the x-axis and Throughtput in MB/sec is the y axis. Here only the randomwrite scenario is considered.

Operation	Memory Size	ThreadCount	Throughput(in MB/sec)
RandomWrite	8KB	1	22743.932
RandomWrite	8KB	2	36863.039

RandomWrite	8KB	4	87985.734
RandomWrite	8KB	8	208156.531
RandomWrite	8MB	1	14735.51
RandomWrite	8MB	2	31367.742
RandomWrite	8MB	4	71223.633
RandomWrite	8MB	8	147218.984
RandomWrite	80MB	1	12503.511
RandomWrite	80MB	2	27559.107
RandomWrite	80MB	4	51020.156
RandomWrite	80MB	8	82442.688

Stream Benchmark:

Run on local VM :

```
-------------------------------------------------------------
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 13299 microseconds.
   (= 13299 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:           10944.8     0.014917     0.014619     0.015679
Scale:          10553.4     0.015496     0.015161     0.017150
Add:            11837.3     0.020899     0.020275     0.024645
Triad:          11528.0     0.021119     0.020819     0.021996
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
```

Run on chameleon:

```
[cc@pa1msss ~]$ gcc -O stream.c -o stream
[cc@pa1msss ~]$ ./stream
-------------------------------------------------------------
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 13391 microseconds.
   (= 13391 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:           10608.6     0.015462     0.015082     0.016293
Scale:          11147.5     0.014624     0.014353     0.015104
Add:            12126.1     0.020389     0.019792     0.021427
Triad:          11951.0     0.021021     0.020082     0.022235
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
```

Clock rate for Memory : 2200 (MHz) (assumption)
Bus Size = 64 bits, or 64/8 = 8 Bytes
DDR3 = 2 (assumption)- Multiplier
Theoretical : 2200Mhz*8*2 = 35200

Stream benchmark performance : 10608.6
Efficiency: (Steam benchmark performance/ Theoretical) *100 = (10608.6/35200)*100 = 30.13%

Conclusion :

1. We have achieved an increase in throughput all for all operations and decrease in latency.
2. The experiment can be further tried by varied blocksize and observing further latency and throughput.

Disk Benchmark:

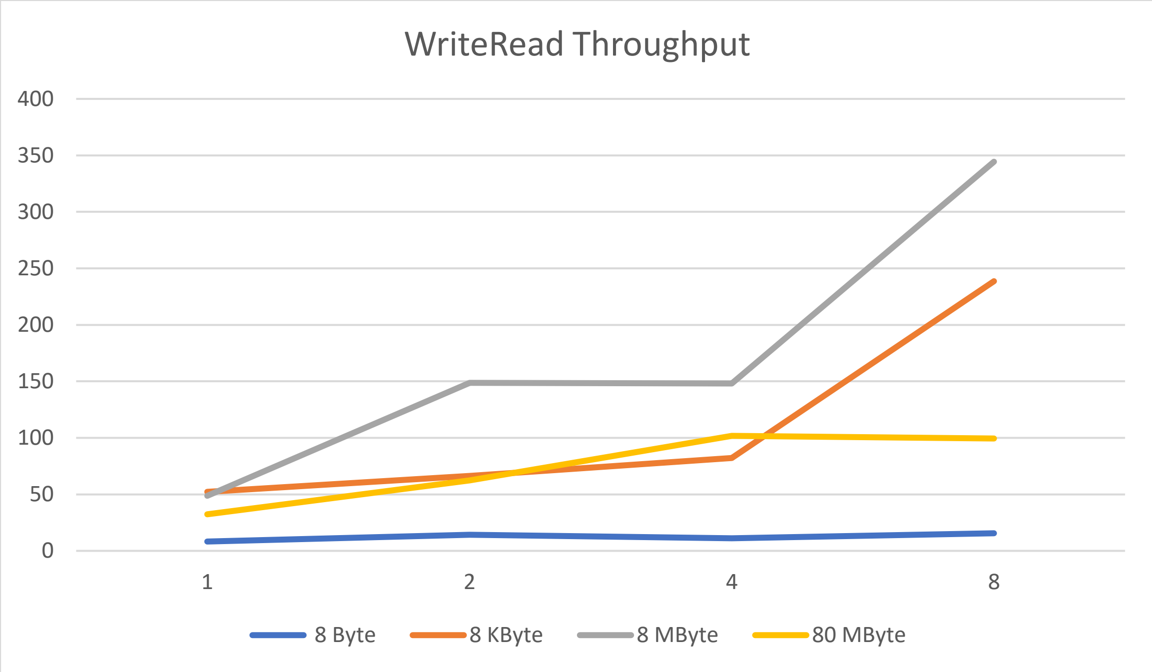
The disk benchmark program does a test on a file of a bigger size, here(8 GB)

- 8GB file size is created for 8B, 8KB, 8MB, 80MB block size and varying concurrency (1, 2, 4, 8) threads, sequential read, random read and read+write operations is calculated and throughput and latency is calculated.
- Here we are calculating average latency accessing a block of 8 Byte from the disk. We have also calculated throughput in MByte per Second.

WriteRead :

Latency (Avg): 0.001471126 ms

Number of Threads	8 Byte	8 KByte	8 MByte	80 MByte
1	8.239	52.233	48.712	32.359
2	14.242	66.26	148.7	62.306
4	11.258	82.201	148.21	101.76
8	15.554	238.64	344.37	99.404

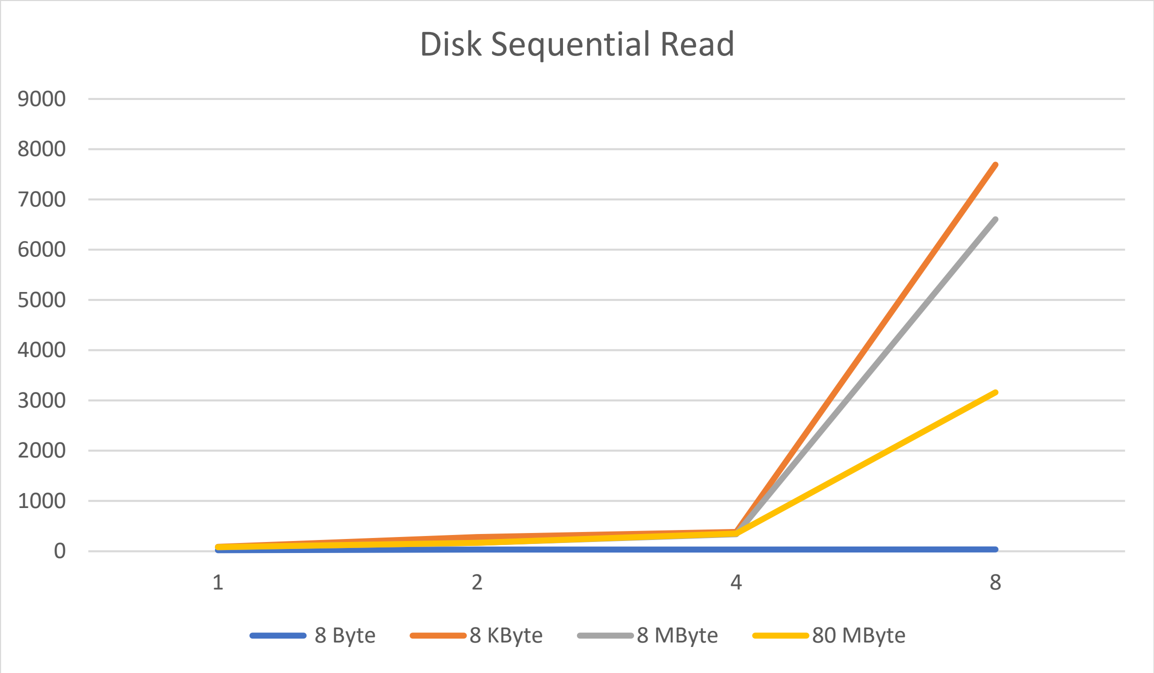


X axis is number of threads and Y axis is throughput

Sequential Read:

Latency (Avg): 0.000425626 ms

Number of Threads	8 Byte	8 KByte	8 MByte	80 MByte
1	21.18	88.122	79.833	82.299
2	33.169	279.46	169.95	165.57
4	32.371	385.3	339.9	349.68
8	36.517	7692.3	6606.1	3161.8

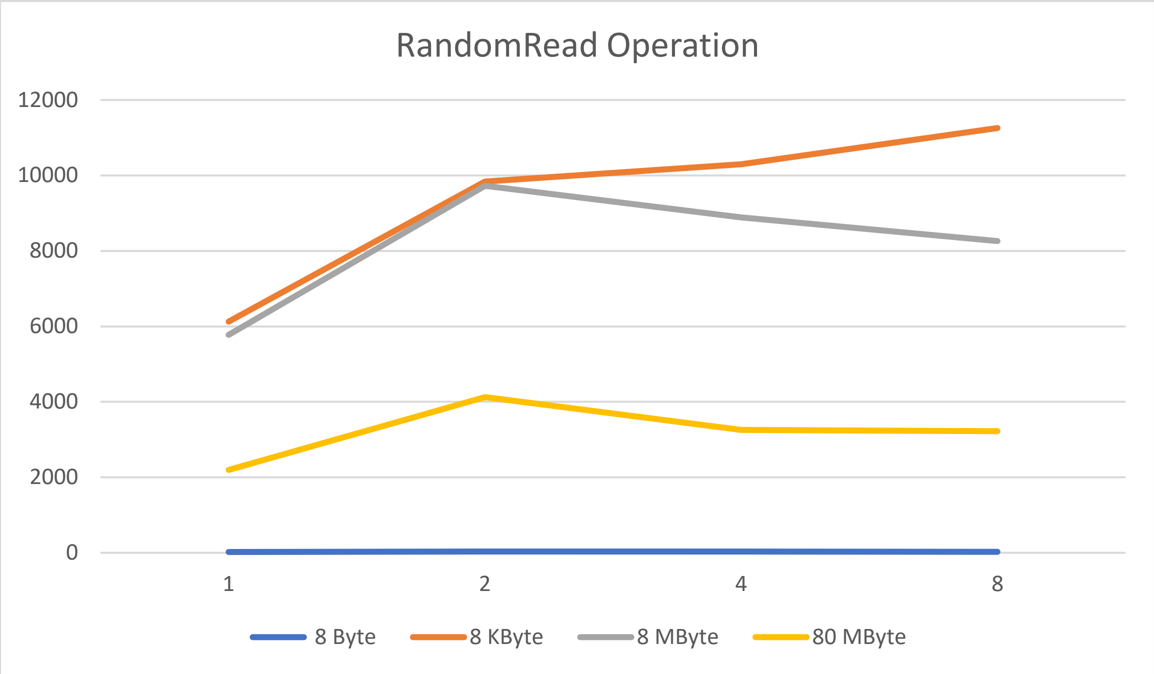


X axis is number of threads and Y axis is throughput

Random READ:

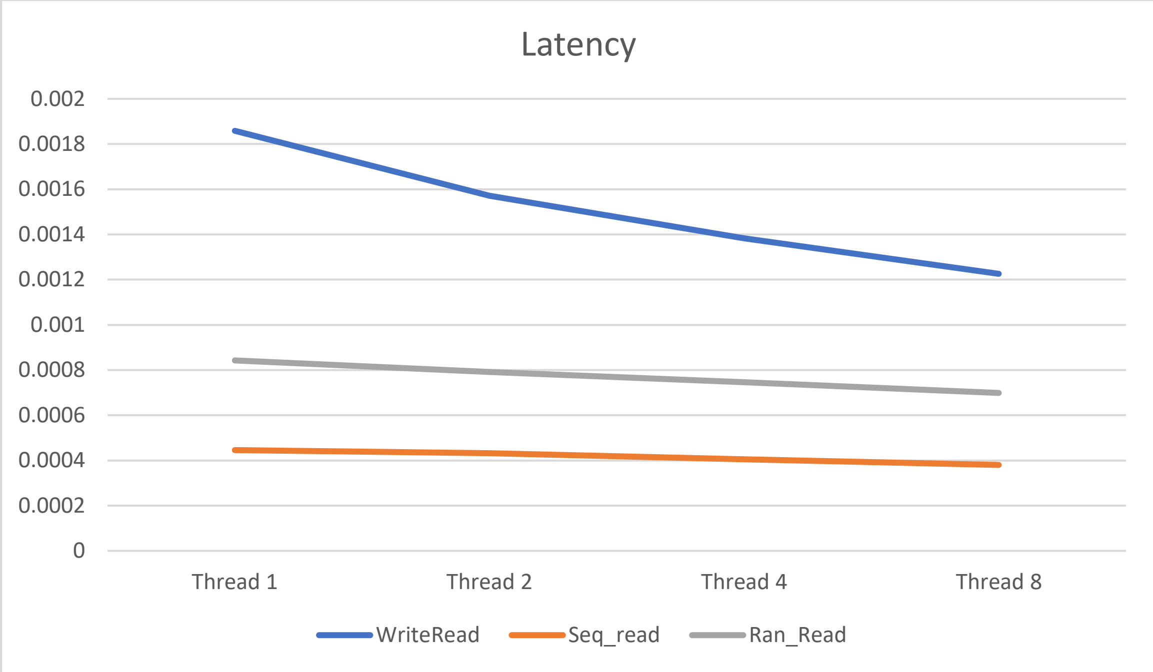
Latency (Avg): 0.0007425856 ms

Number of Threads	8 Byte	8 KByte	8 MByte	80 MByte
1	21.181	6125.7	5774.7	2195.68
2	30.646	9835.6	9725	4125.34
4	32.894	10299	8890.6	3257.43
8	25.507	11256	8258.7	3225.93



X axis is number of threads and Y axis is throughput.

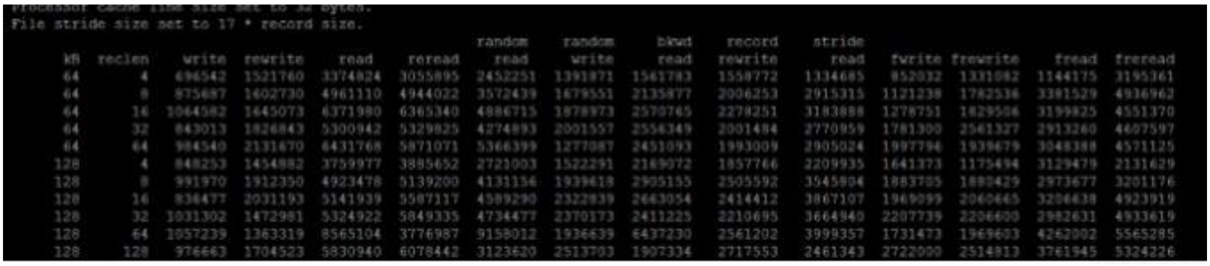
	WriteRead	Seq_read	Ran_Read
Thread 1	0.001859	0.000446	0.000843
Thread 2	0.001571	0.000433	0.000791
Thread 4	0.001383	0.000406	0.000746
Thread 8	0.001226	0.00038	0.000699



X axis is number of threads and Y axis is latency.

The disk we are observing can be a HDD as the latency is less than 5 ms.

IOZone:


Processor cache line size set to 32 bytes.
File stride size set to 17 * record size.

kB	reclen	write	rewrite	read	reread	random read	random write	bkwd read	record rewrite	stride read	fwrite	frewrite	fread	freread
64	4	696542	1521760	3374824	3055895	2452251	1391871	1561783	1558772	1334685	852032	1331082	1144175	3195361
64	8	875687	1602730	4961110	4944022	3572439	1679551	2135877	2006253	2915315	1121238	1782536	3381529	4936962
64	16	1064582	1645073	6371980	6365340	4886715	1878973	2570765	2278251	3183888	1278751	1829506	3199825	4551370
64	32	843013	1826843	5300942	5329825	4274893	2001557	2556349	2001484	2770959	1781300	2561327	2913260	4607597
64	64	984540	2131670	6431768	5871071	5366399	1277087	2451093	1993009	2905024	1997796	1939679	3048388	4571125
128	4	848253	1454882	3759977	3885652	2721003	1522291	2169072	1857766	2209935	1641373	1175494	3129479	2131629
128	8	991970	1912350	4923478	5139200	4131156	1939618	2905155	2505592	3545904	1883705	1880429	2973677	3201176
128	16	836477	2031193	5141939	5587117	4589290	2322839	2663054	2414412	3867107	1969099	2060665	3206638	4923919
128	32	1031302	1472981	5324922	5849335	4734477	2370173	2411225	2210695	3664940	2207739	2206600	2982631	4933619
128	64	1057239	1363319	8565104	3776987	9158012	1936639	6437230	2561202	3999357	1731473	1969603	4262002	5565285
128	128	976663	1704523	5830940	6078442	3123620	2513703	1907334	2717553	2461343	2722000	2514813	3761945	5324226

Conclusion:

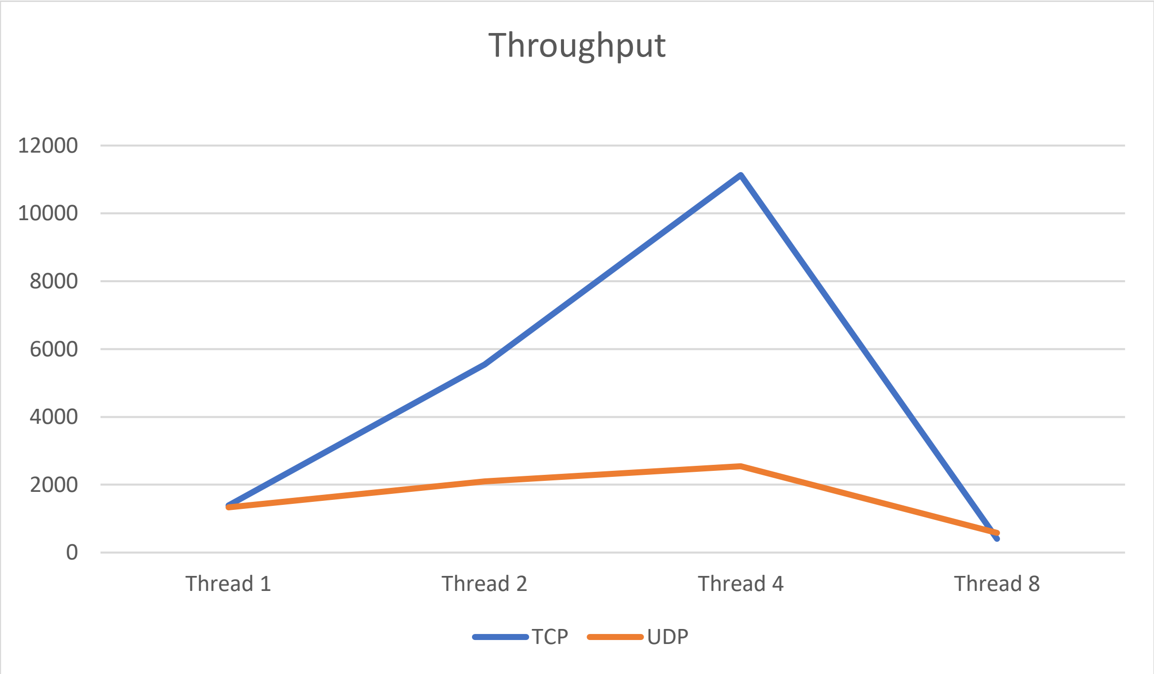
From Disk Benchmark performance we conclude that write read operation takes more run time than any other operation(Ran_read or Seq_read). The throughput is achived maximum in the case with of 80mb block size with least latency for 8 thread. Throughput increases with increase in number of threads.

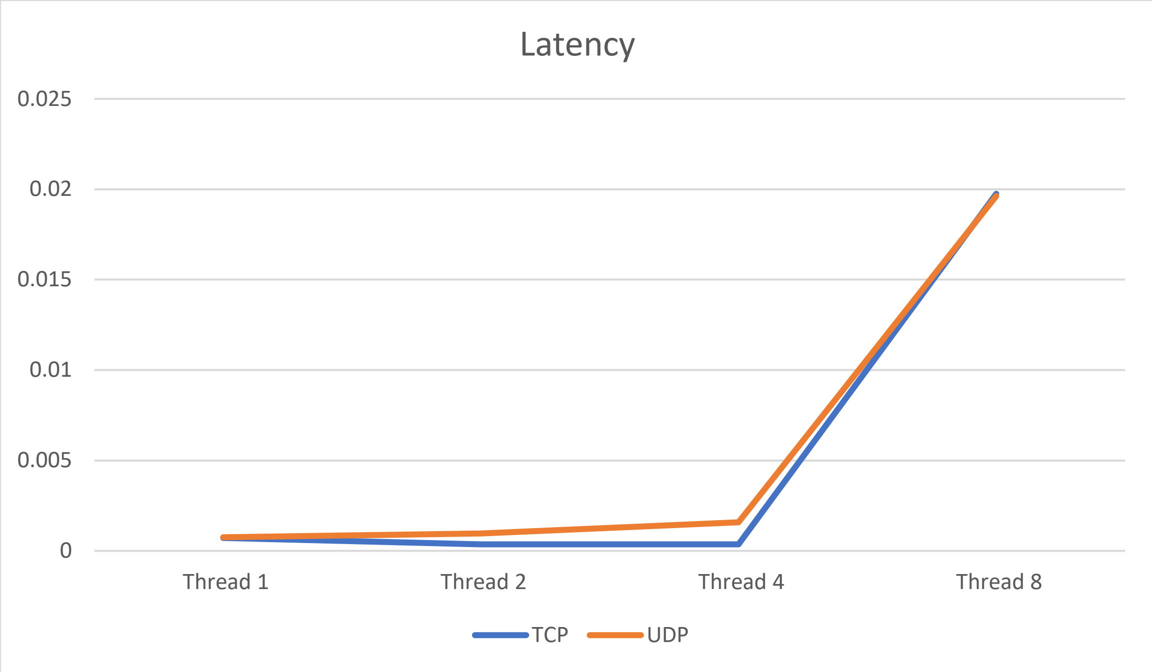
Network Benchmark:

Network performance is measured in terms of throughput and latency for sending a packet. Throughput is measured in mb/sec and latency is measured in ms. The packet send here is of 64kb as fixed packet size to calculate throughputs while 8kb packet size to measure latency.

The observations are as shown below:

No. of thread	Connection type	Throughput	Latency
1	tcp	1387.534 Mbits/sec	0.000721 ms/bit
2	tcp	5542.625 Mbits/sec	0.000361 ms/bit
4	tcp	11130.435 Mbits/sec	0.000359 ms/bit
8	tcp	405.056 Mbits/sec	0.019750 ms/bit
1	udp	1331.599 Mbits/sec	0.000751 ms/bit
2	udp	2098.361 Mbits/sec	0.000953 ms/bit
4	udp	2545.680 Mbits/sec	0.001571 ms/bit
8	udp	580.522 Mbits/sec	0.019625 ms/bit





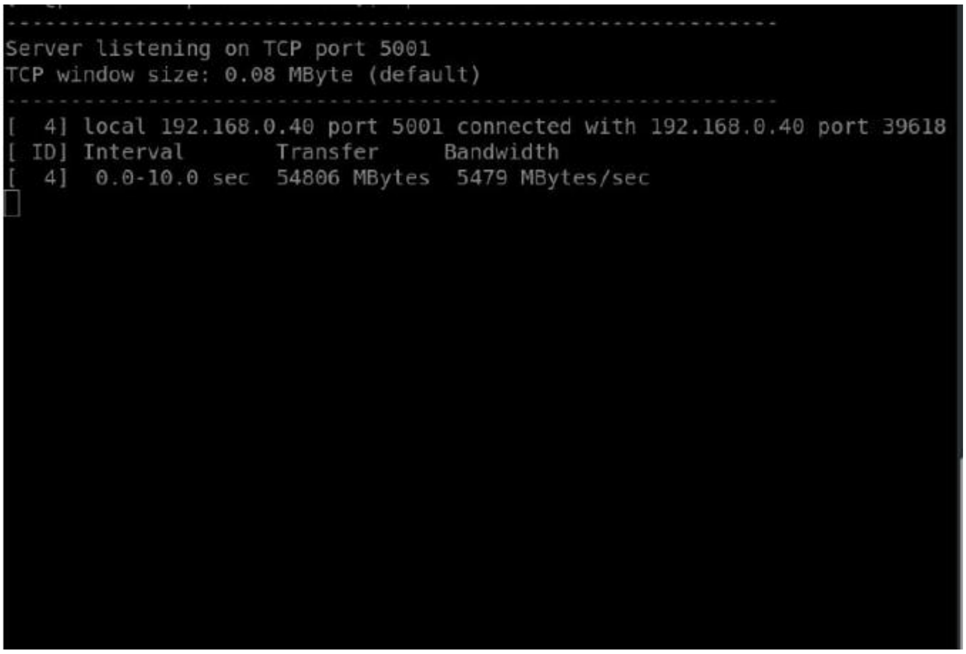
X axis is number of threads and y is latency

IPERF Benchmark:

Steps:
1. Downloaded iperf3-3.1.3-1.fc24.x86_64.rpm
2. Moved the file onto the Chameleon testbed.
3. sudo yum install iperf3-3.1.3-1.fc24.x86_64.rpm
4. On server side, We've used the command: iperf3 -s
5. On Client side, We've used the command: iperf3 -c (public ip address of instance) port (port no)
“ iperf3 -c 192.168.0.121 port 22 “

Outputs:

Server TCP



```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 0.08 MByte (default)
------------------------------------------------------------
[  4] local 192.168.0.40 port 5001 connected with 192.168.0.40 port 39618
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-10.0 sec  54806 MBytes  5479 MBytes/sec
```

```
-bash: iperf3-s: command not found
[cc@pa1-prat ~]$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.0.199, port 56226
[  5] local 192.168.0.199 port 5201 connected to 192.168.0.199 port 56228
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec  4.40 GBytes  37.8 Gbits/sec
[  5]   1.00-2.00   sec  5.14 GBytes  44.2 Gbits/sec
[  5]   2.00-3.00   sec  5.47 GBytes  47.0 Gbits/sec
[  5]   3.00-4.00   sec  5.29 GBytes  45.4 Gbits/sec
[  5]   4.00-5.00   sec  5.09 GBytes  43.8 Gbits/sec
[  5]   5.00-6.00   sec  5.24 GBytes  45.0 Gbits/sec
[  5]   6.00-7.00   sec  4.15 GBytes  35.7 Gbits/sec
[  5]   7.00-8.00   sec  4.22 GBytes  36.2 Gbits/sec
[  5]   8.00-9.00   sec  5.01 GBytes  43.1 Gbits/sec
[  5]   9.00-10.00  sec  5.29 GBytes  45.4 Gbits/sec
[  5]  10.00-10.00  sec  0.00 Bytes  0.00 bits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-10.00  sec  0.00 Bytes  0.00 bits/sec                  sender
[  5]   0.00-10.00  sec  49.3 GBytes  42.3 Gbits/sec                  receiver
-----------------------------------------------------------
```

Client TCP

```
------------------------------------------------------------
Client connecting to 192.168.0.40, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  3] local 192.168.0.40 port 39618 connected with 192.168.0.40 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  54806 MBytes  5480 MBytes/sec
```

```
[cc@pa1-prat ~]$ iperf -c 192.168.0.199 port 22
-bash: iperf: command not found
[cc@pa1-prat ~]$ iperf3 -c 192.168.0.199 port 22
Connecting to host 192.168.0.199, port 5201
[  4] local 192.168.0.199 port 56228 connected to 192.168.0.199 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  4.40 GBytes  37.8 Gbits/sec    0   3.25 MBytes
[  4]   1.00-2.00   sec  5.15 GBytes  44.3 Gbits/sec    0   3.25 MBytes
[  4]   2.00-3.00   sec  5.46 GBytes  46.9 Gbits/sec    0   3.25 MBytes
[  4]   3.00-4.00   sec  5.29 GBytes  45.5 Gbits/sec    0   3.25 MBytes
[  4]   4.00-5.00   sec  5.09 GBytes  43.7 Gbits/sec    0   3.25 MBytes
[  4]   5.00-6.00   sec  5.24 GBytes  45.0 Gbits/sec    0   3.25 MBytes
[  4]   6.00-7.00   sec  4.15 GBytes  35.7 Gbits/sec    0   3.25 MBytes
[  4]   7.00-8.00   sec  4.22 GBytes  36.2 Gbits/sec    0   3.25 MBytes
[  4]   8.00-9.00   sec  5.02 GBytes  43.1 Gbits/sec    0   3.25 MBytes
[  4]   9.00-10.00  sec  5.28 GBytes  45.4 Gbits/sec    0   3.25 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  49.3 GBytes  42.4 Gbits/sec    0             sender
[  4]   0.00-10.00  sec  49.3 GBytes  42.4 Gbits/sec                  receiver

iperf Done.
[cc@pa1-prat ~]$
[cc@pa1-prat ~]$
```

Client udp

```
------------------------------------------------------------
Client connecting to 192.168.0.40, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size: 0.20 MByte (default)
------------------------------------------------------------
[  3] local 192.168.0.40 port 53556 connected with 192.168.0.40 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  0.12 MBytes/sec
[  3] Sent 893 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec  1.25 MBytes  0.13 MBytes/sec   0.000 ms    0/  893 (0
%)
```

Server udp

```
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size: 0.20 MByte (default)
------------------------------------------------------------
[  3] local 192.168.0.40 port 5001 connected with 192.168.0.40 port 53556
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Da
tagrams
[  3]  0.0-10.0 sec  1.25 MBytes  0.13 MBytes/sec   0.001 ms    0/  893 (
0%)
```

Conclusion:

We can view from the plot that the throughputs of TCP is higher than throughputs of UDP, while latency goes almost parallel being latency of UDP more than latency of TCP with considering different threads with different block size in all operations. Also we conclude that running iperf benchmark code gives slightly better result than the coded throughput results.