Aditya Kale – A20355807

Cloud Computing

Programming Assignment – 1

- System Configuration:
  - I have performed all benchmarking experiment on amazon t2.micro instance.
  - RAM : 1 Gb
  - Storage: 8GB
  - Cores: 1
  - Operating system: Linux Ubuntu 14

Disk Benchmarking:

- 100 MB of data is been written and read on a file.
- These read write operations are performed sequentially and randomly by varying the data size as 1 Byte, 1 Kilo Byte and 1 Mega Byte.
- Following results are captured while performing the benchmark experiment. And based on these observations graphs are plotted.

For 1 Byte of Data:

		Latency in milli sec		Throughput In MB/S	
No of Threads	Access Type	Read	Write	Read	Write
1	Sequential	3.1E-05	2.9E-05	30.7778	32.9569
	Random	0.00015	0.00117	6.21197	0.81575
2	Sequential	0.00003	2.9E-05	31.2831	33.2819
	Random	0.00015	0.00117	6.1811	0.81326

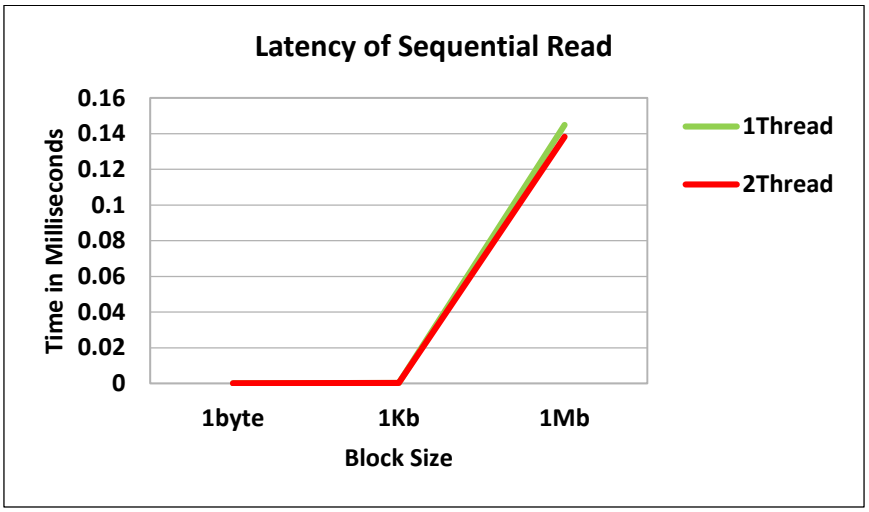
For 1 Kilo Byte Data:

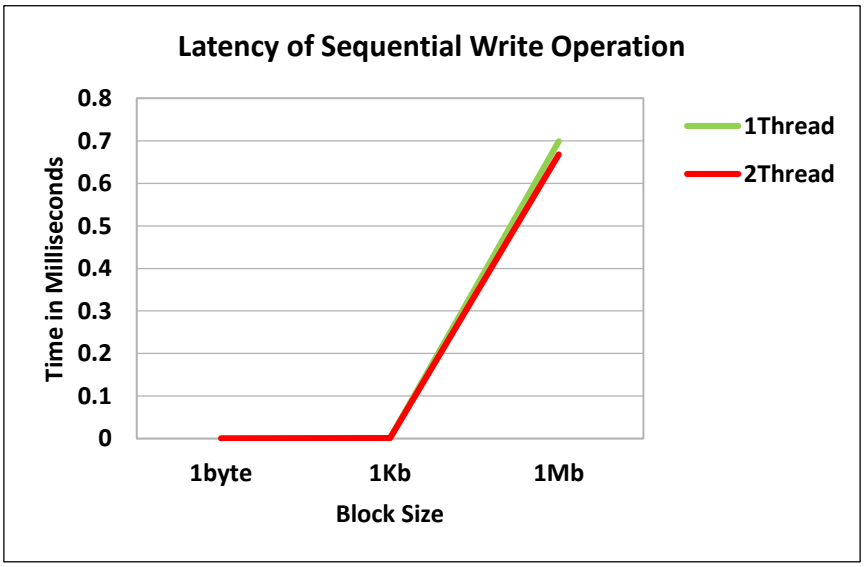
		Latency in milli sec		Throughput in MB/S	
No of Threads	Access Type	Read	Write	Read	Write
1	Sequential	0.00025	0.00089	3899.85	1097.35
	Random	0.00022	0.00183	4457.76	532.846
2	Sequential	0.00025	0.00098	3987.68	1294.61
	Random	0.00022	0.00199	4502.05	689.677

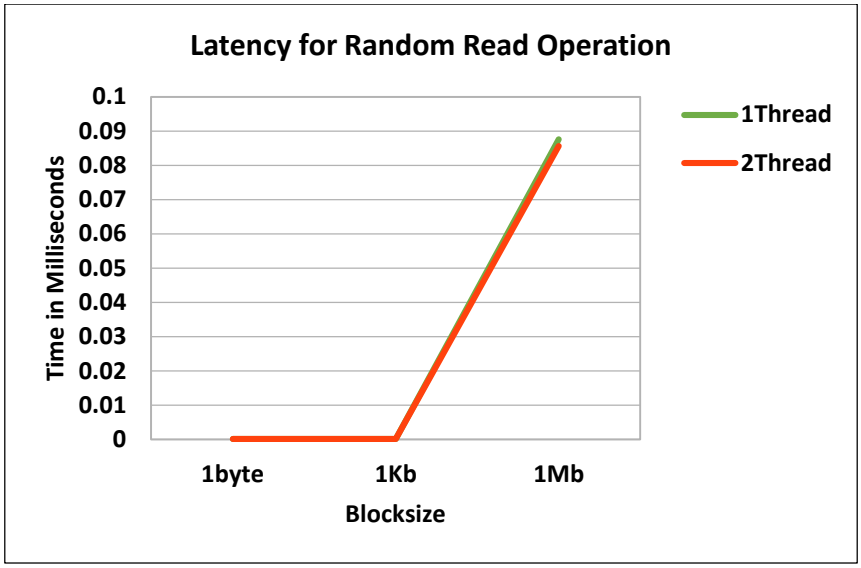
For 1 Mega Byte Data:

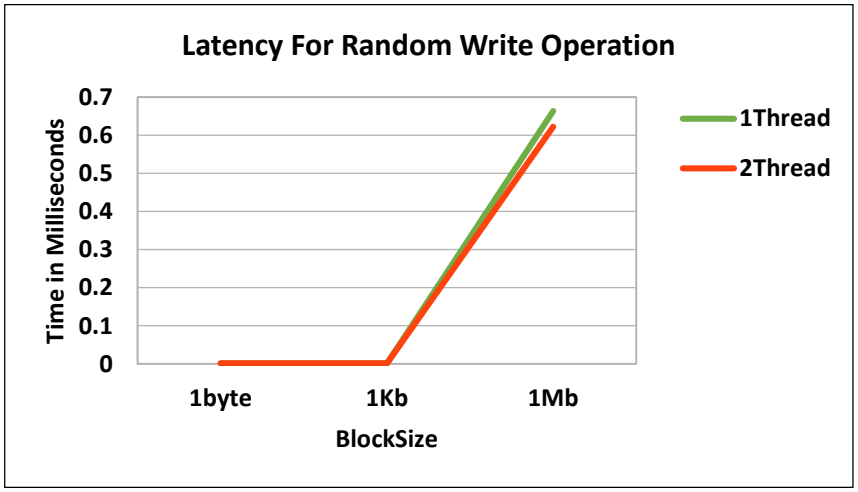
		Latency in MS		Throughput	
No of Threads	Access Type	Read	Write	Read	Write
1	Sequential	0.14493	0.6989	6899.88	1430.82
	Random	0.08756	0.66376	11420.7	1506.57
2	Sequential	0.13829	0.66787	7231.18	1606.3
	Random	0.08564	0.62262	11676.8	1497.13

Following are the graphs plotted for Latency using above data.



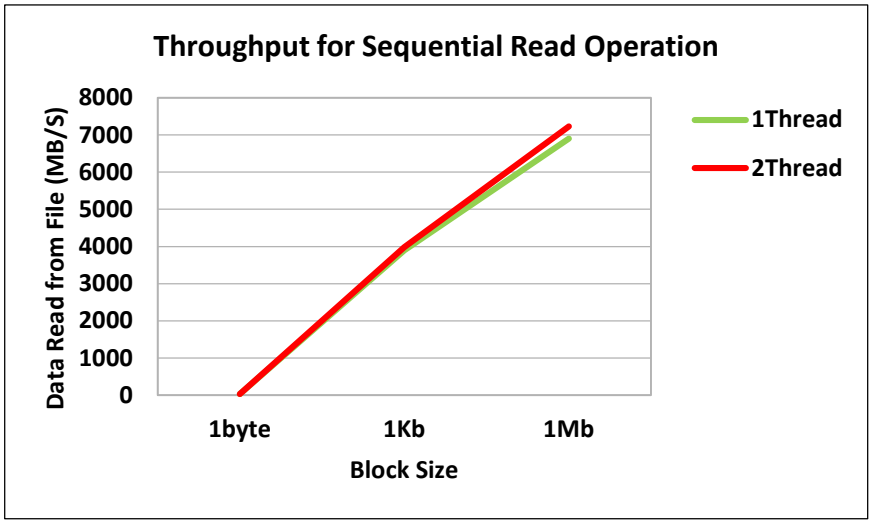


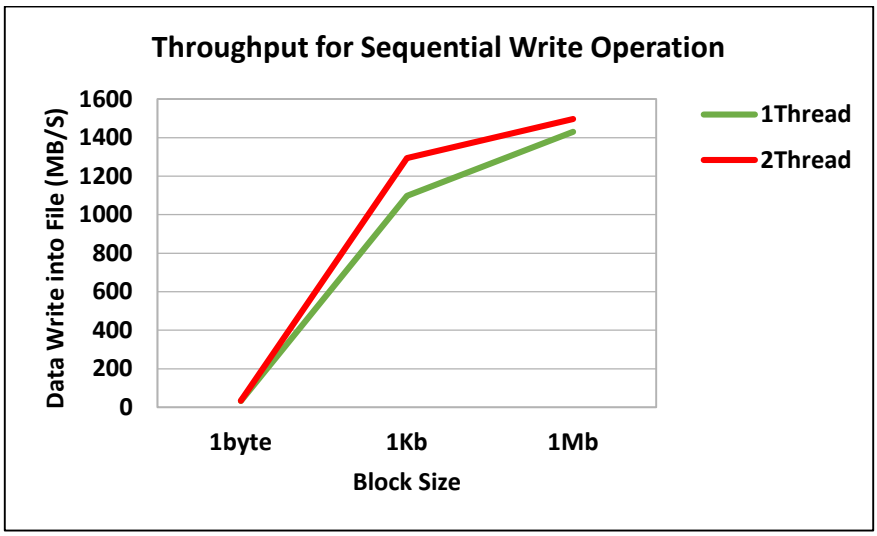


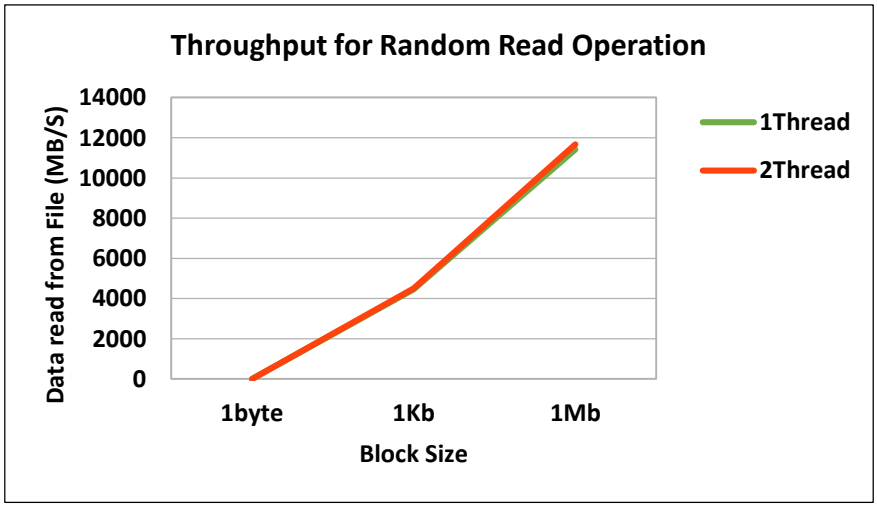


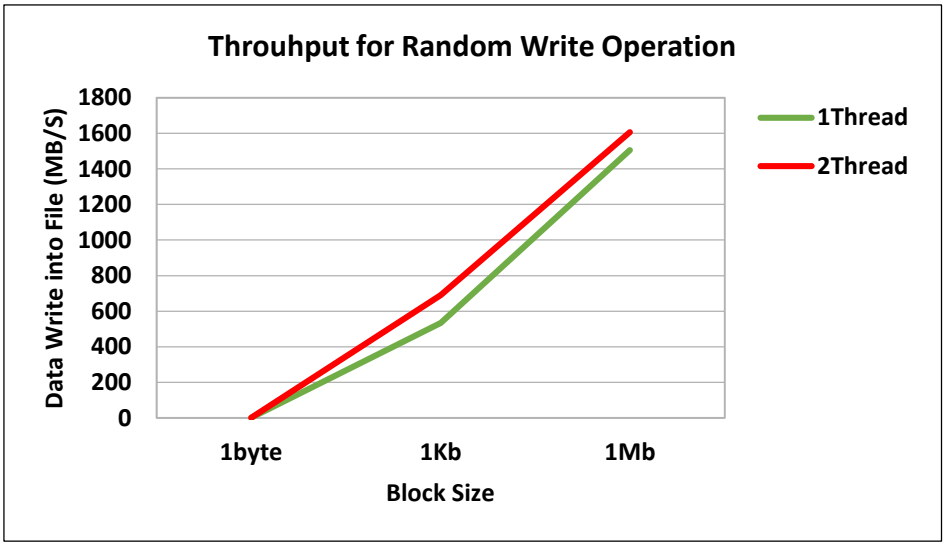
- **Analysis on Latency for Disk read write operations.**
  - Latency for sequential read write operations by 1 thread is always greater than the latency for two threads (when block size is 1 MB).
  - Same result is found for random read write operations.
- **Conclusion:**
  - Latency decreases as multiple threads increases.

Following are the graphs plotted for Throughput using above data.





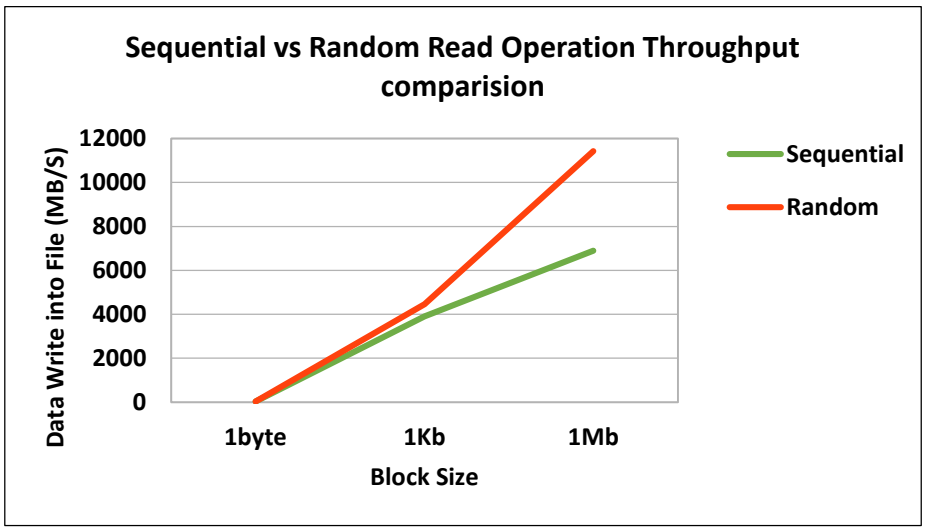




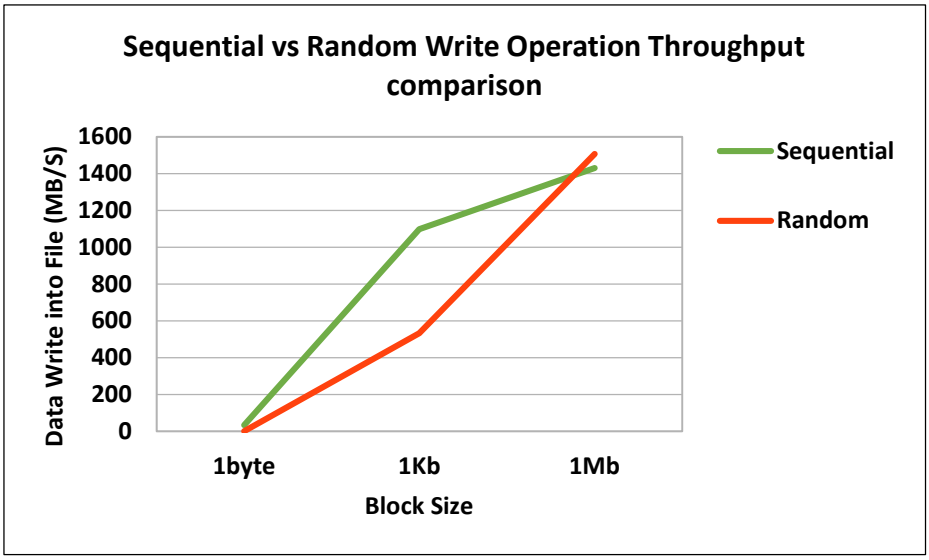
- **Analysis on Throughput for Disk read write operations.**
  - Throughput for sequential read write operations for 2 thread is always greater than the throughput for 1 thread.
  - Throughput for random read write operations for 2 thread is always greater than the throughput for 1 thread.
  - Throughput does not increase significantly when data block size is increased from 1byte to 1 kilo Byte.
  - Throughput increases by a large amount when data block size is increased from 1Kb to 1Mb.

Throughput comparison Sequential vs Randomoperations for 1 Thread :

Sequential vs Random read operation throughput.

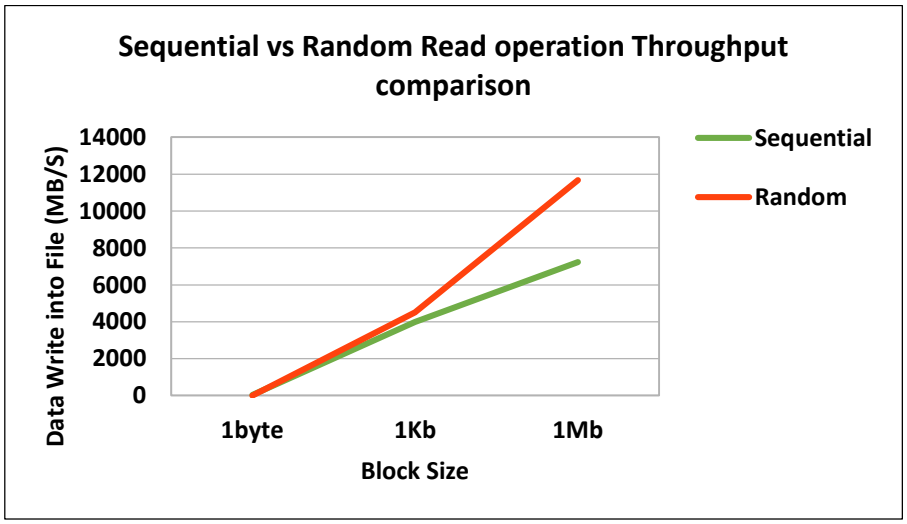


Sequential vs Random write operation throughput.

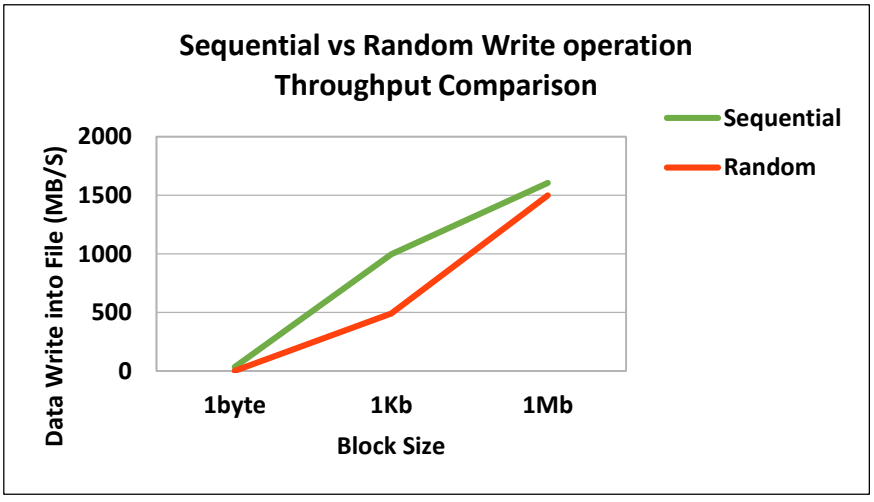


For 2 Thread:

Sequential vs Random read operation throughput.



Sequential vs Random write operation throughput.



Analysis:

- Sequential write is faster than random write as for random write pointer has to move frequently to different disk location. Thus increasing the seek time and thus decreasing throughput.
- Similarly throughput for reading/writing 1 Byte data is small as compared throughput for reading / writing 1MB of data.

Theoretical Performance of a disk:

- For t2.micro instance theoretical value for disk is given as 160Mb/S.

Extra Credit: IOZONE Benchmarking system

- I have evaluated IZONE benchmarking system on amazon t2.micro instance.
- Also I have compared these benchmarking value with my implementation.
- Following are the results observed after running the IOZONE benchmark.

```
ubuntu@ip-172-31-45-63:~/iozone3_394/src/current$ ./iozone -r#m -a -b op.xls
        Iozone: Performance Test of File I/O
                Version $Revision: 3.394 $
                Compiled for 64 bit mode.
                Build: linux

        Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
                     Al Slater, Scott Rhine, Mike Wisner, Ken Goss
                     Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
                     Randy Dunlap, Mark Montague, Dan Million, Gavin Brebner,
                     Jean-Marc Zucconi, Jeff Blomberg, Benny Halevy, Dave Boone,
                     Erik Habbinga, Kris Strecker, Walter Wong, Joshua Root,
                     Fabrice Bacchella, Zhenghua Xue, Qin Li, Darren Sawyer.
                     Ben England.

        Run began: Fri Feb 12 23:21:58 2016

        Record Size 4 KB
        Auto Mode
        Command line used: ./iozone -r#m -a -b op.xls
        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 1232453 2067979 12902017 12902017 7100397 3791156 4897948  4643754 12902017  3958892  3738358 6421025  7940539
             128       4 1487977 3657016 11720614 11720614 8548124 3867787 4104338  5325799  7582312  3785961  3867787 8548124  8548124
             256       4 1686137 4350555  9192546 12228612 9518507 3410808 7965107  5320671  5971678  3368013  4070199 9434868  4929815
             512       4 1585086 4228947 11374040 12215097 7871843 3905895 7871843  5684095  8665997  3963567  4091959 4784885 10641343
            1024       4 1597351 3849207 10906310 11614118 9142007 3908759 9300377  5690162  7319232  2403712  2107643 9402175 11018226
            2048       4 1977058 4204968 11320334  5644567 9441504 4154130 7065219  5611381  9101380  3977179  408693010240672 10959265
            4096       4 1979748 3349029  9188050 11193514 9327731 3823862 6714018  5299475  8902382  4072211  4128976 9846293  9897345
            8192       4 1978244 2805379  6390362 11329356 9102385 3942233 5728718  5692651  8858329  4025365  2811347 6325656 10422130
           16384       4 1885835 2597716  6453256  9086505 8655057 3746603 5750677  5793828  6272998  3348444  2566380 6303494  8520909
           32768       4 1758030 2624593  6468078  6587750 5746543 2452698 5765103  5822252  5799648  2601886  2578940 6304016  6586487
           65536       4 1729501 2685907  6624552  6725038 5587993 2419278 5983957  5823955  5859711  2626896  2594881 6451048  6516504
          131072       4 1648019 2600943  6726421  6737880 5357105 2324098 5869771  5785575  5745006  2493949  2530349 6510623  6510007
          262144       4  164432  159431  6747796  6743947 5184106  141934 5824893  5763068  5550599   308948   154959 6457239  6504030
```

For 1 MB data following results are found.

Operations	Throughput in MB/S
Sequential Read	9402.175
Sequential Write	2403.712
Random Read	9142.007
Random Write	3908.759

- **Comparing this values with my system values.**

Operations	My System Throughput	IOZONE Throughput
Sequential Read	7231.180852	9402.175
Sequential Write	1497.297378	2403.712
Random Read	4457.764641	9142.007
Random Write	1606.128988	3908.759

- **Efficiency for my system is calculated and found to be:**

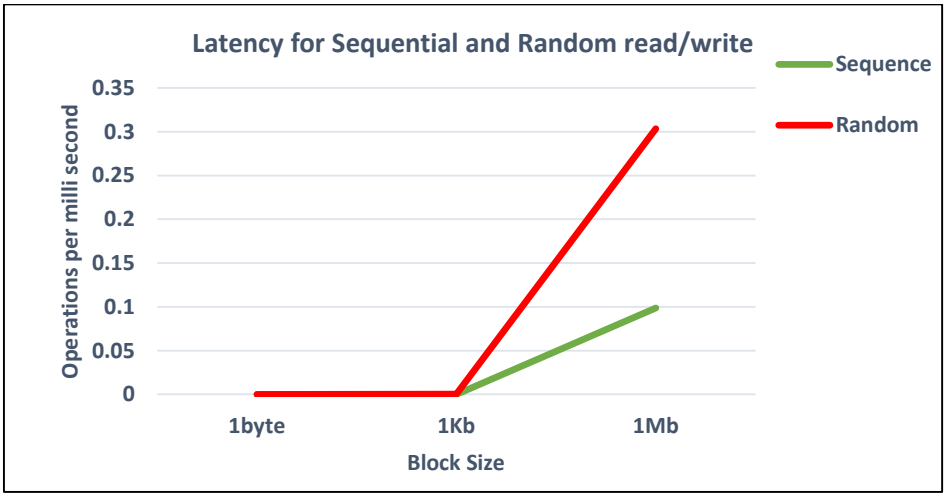
Operations	Efficiency of My system
Sequential Read	76.90966029 %
Sequential Write	62.29104726 %
Random Read	48.7613348 %
Random Write	41.09050949 %

Memory Benchmarking:

- 100 MB of data is been written and read into a memory.
- These read write operations are performed sequentially and randomly by varying the data size as 1 Byte, 1 Kilo Byte and 1 Mega Byte.
- Following results are observed and graphs are plotted using these observations.

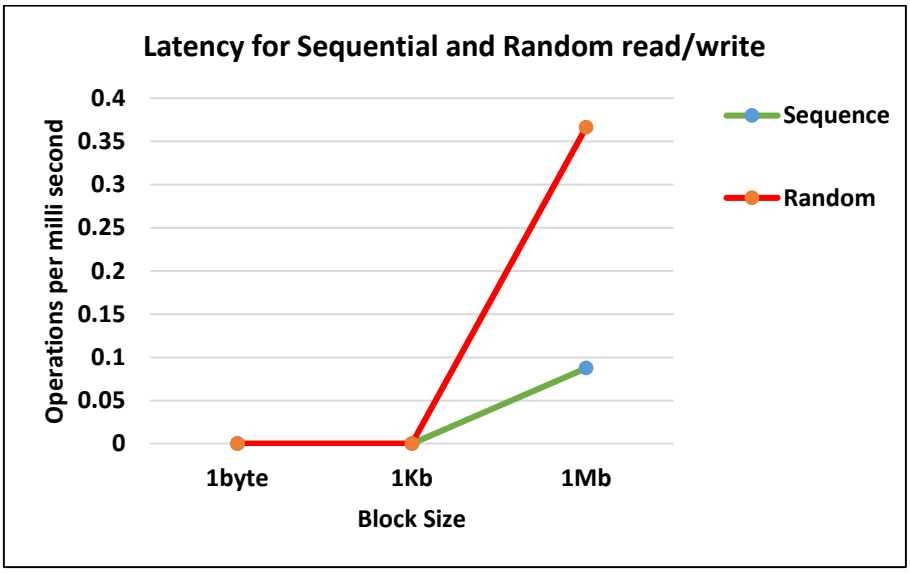
Latency for Sequential and Random read/write for 1 thread

	Latency Thread1	
Bytesize	**Sequence**	**Random**
1byte	5E-06	0.000001
1Kb	3.7E-05	0.000439
1Mb	0.0987	0.3032



Latency for Sequential and Random read/write for 2 concurrent threads

	Latency Thread2	
Bytesize	**Sequence**	**Random**
1byte	5E-06	0.000001
1Kb	3.6E-05	0.000063
1Mb	0.0875	0.3665

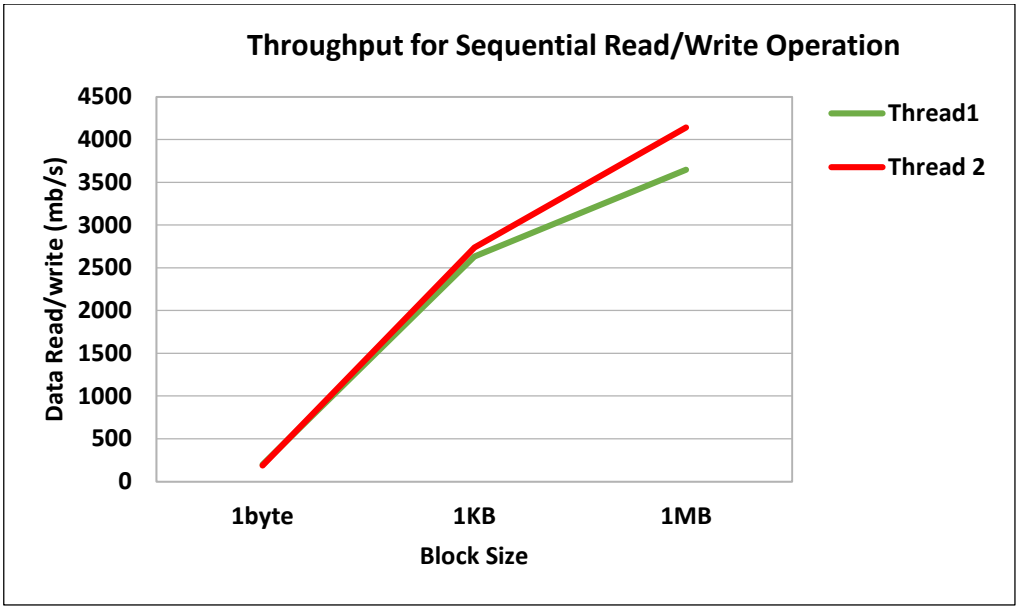


Observations:

- Latency of a system increases by a large amount for 1MB packets.
- Latency for Random read/write is greater than Sequential read write operations.

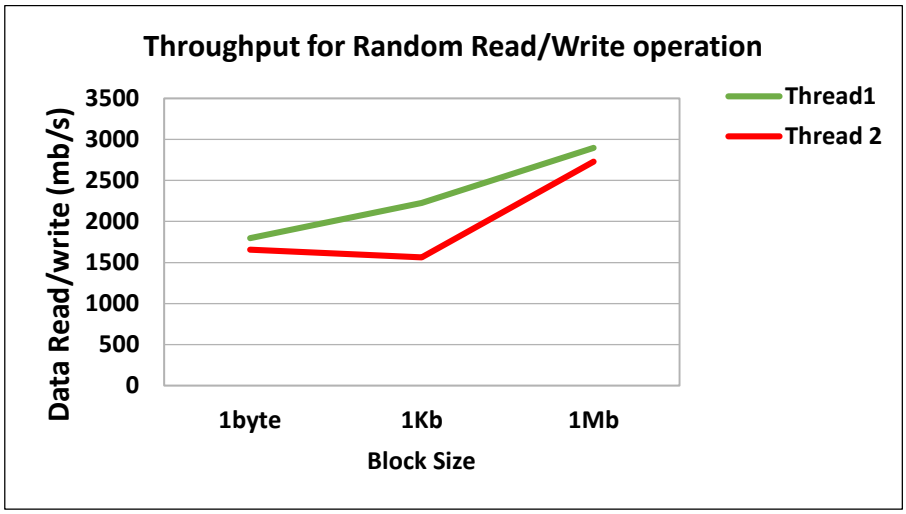
Throughput for Sequential read/write.

Byte Size	Thread1	Thread 2
1byte	198.183	186.35733
1KB	2632.44	2735.6919
1MB	3647.71	4142.5714



Throughput for Random read/write.

Byte Size	Thread1	Thread 2
1byte	1795.66	1656.6602
1Kb	2226.04	1562.51
1Mb	2898.15	2728.513

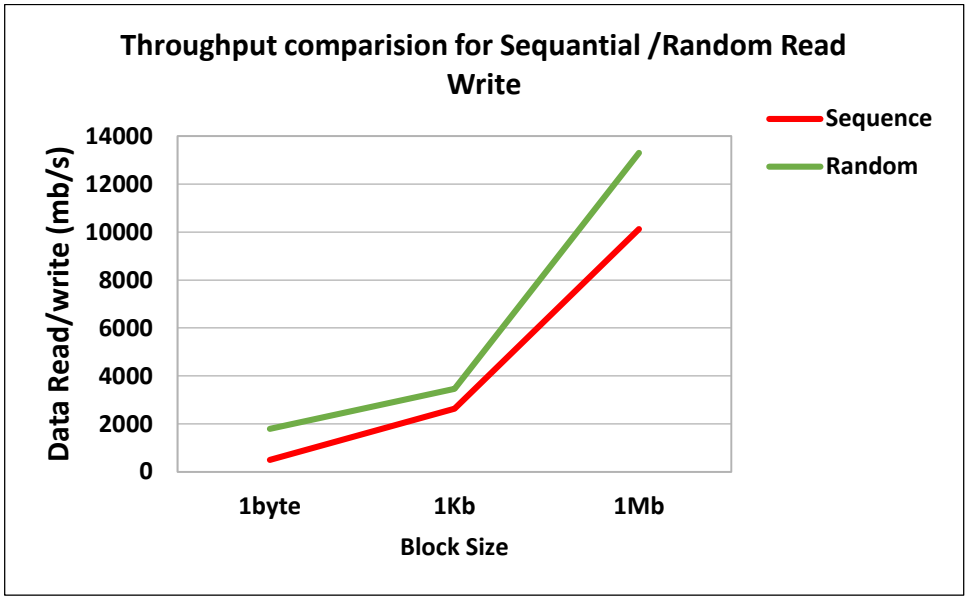


Observations:

- Throughput of a sequential read/write operations is greater than random read/write operations.
- In random read/write throughput decreases when 2 concurrent threads are run.

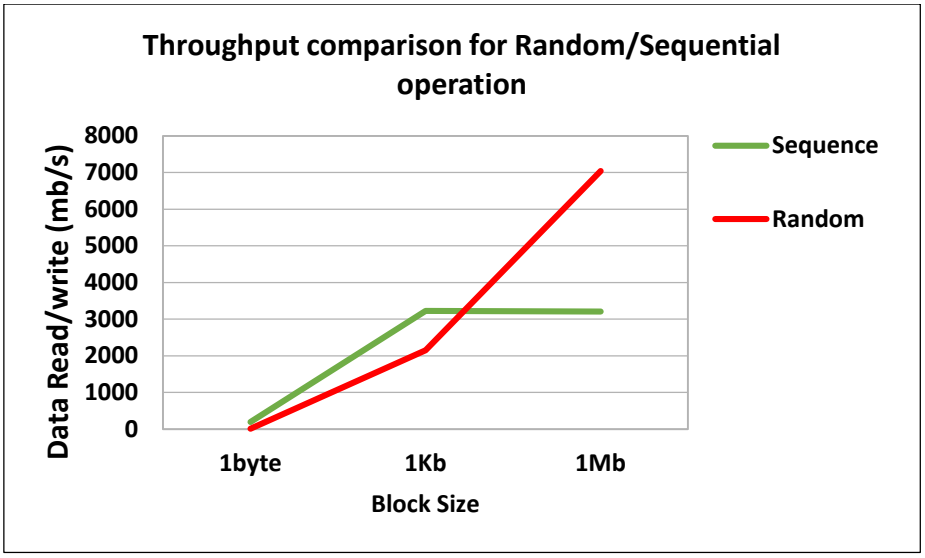
Throughput Comparison for Sequential and Random read/write operations for 1 Thread.

Bytesize	Sequence	Random
1byte	498.183	1795.6587
1Kb	2632.44	3458.0372
1Mb	10131.7	13298.153



Throughput Comparison for Sequential and Random read/write operations for 2 Thread.

Bytesize	Sequence	Random
1byte	194.847	8.16941
1Kb	3221.65	2151.3936
1Mb	3203.64	7037.7929



Observations:

- In above graph Sequential read/write is fast for 1byte and 1Kbyte data. However for 1MB data random read/write gives more throughput.
- When multiple threads are used throughput increases by small amount.

Theatrical value of Memory is calculated by multiplying

  - Clock frequency
  - Number of data transferred per clock (2 for DDR3)
  - Memory bus interface width
  - Number of interfaces (typically 2 for modern computers).
- The above values can be found out by executing following command.

  sudo lshw -c memory

- It gives value in MegaBits/ Second. Dividing by 8 we get Memory Theoretical value as

  =1600*2*(64/8)*2

  = 51200 MHz

  = **51.2 Ghz**

Extra Credit: Stream Benchmarking system

- I have evaluated STREAM benchmarking system on amazon t2.micro instance.
- Also I have compared these benchmarking value with my implementation.
- Following are the results observed after running the Stream benchmark system.

```
-------------------------------------------------------------
ubuntu@ip-172-31-45-63:~$ ./mystream
-------------------------------------------------------------
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 29111 microseconds.
   (= 29111 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:           5510.4     0.029233     0.029036     0.029602
Scale:          5404.9     0.029803     0.029603     0.030062
Add:            7749.9     0.031151     0.030968     0.031438
Triad:          7280.3     0.033177     0.032966     0.033316
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
```

My System Throughput in MB/S	**Stream Throughput in MB/S**
3298.5	6485.75

- **Efficiency gained from a system is calculated as :**

  = 3298.5/6485.75

  = 50.85 %

Network Benchmarking:

- 1000 packets of varying size is transferred over a TCP and UDP network.
- Same experiment is repeated for two concurrent clients.
- Following results are captured while performing the benchmark experiment. And based on these observations graphs are plotted.

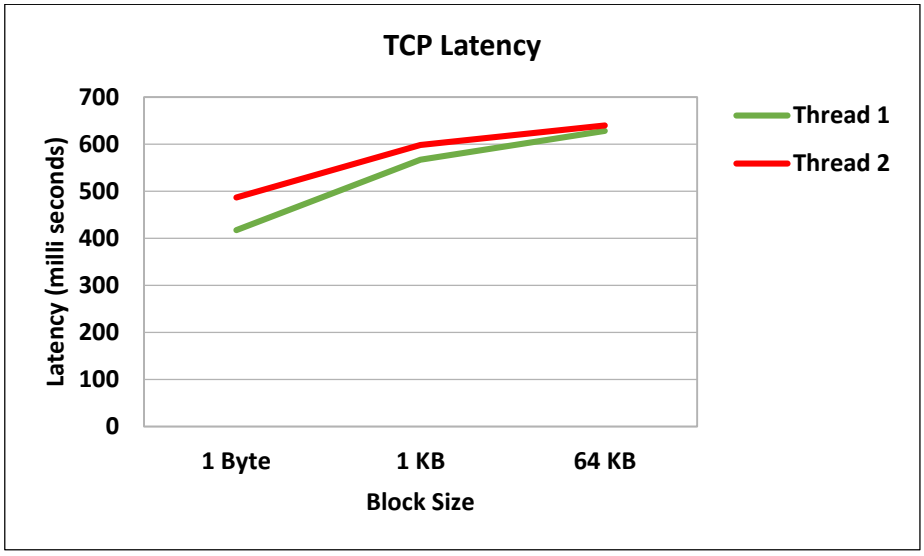
TCP Readings

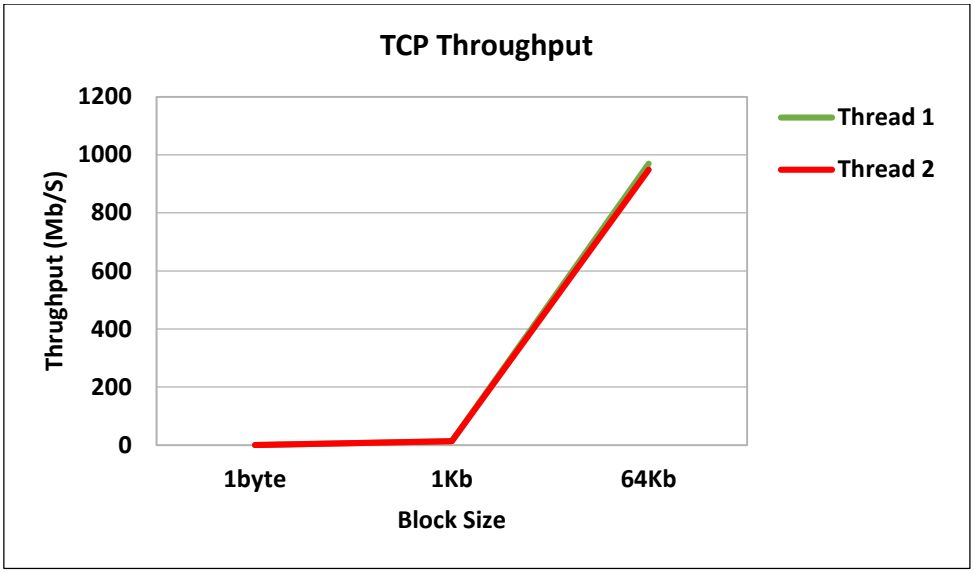
Single Client

Byte Size	Latency in milli sec	Throughput in Mb/s
1Byte	420	0.01296
1 Kilo Byte	547	14.0969
64 Kilo Byte	613	969.697

Multiple Client

Byte Size	Latency Client 1	Latency Client 2	Average Latency In milli sec	Throughput Client 1	Throughput client 2	Average Throughput in Mb/S
1Byte	417	486	451.5	0.00173	0.00168	0.0136
1 Kilo Byte	567	598.5	582.7	1.71233	1.7094	13.6869
64 Kilo Byte	628	640	634	118.409	118.519	949.168





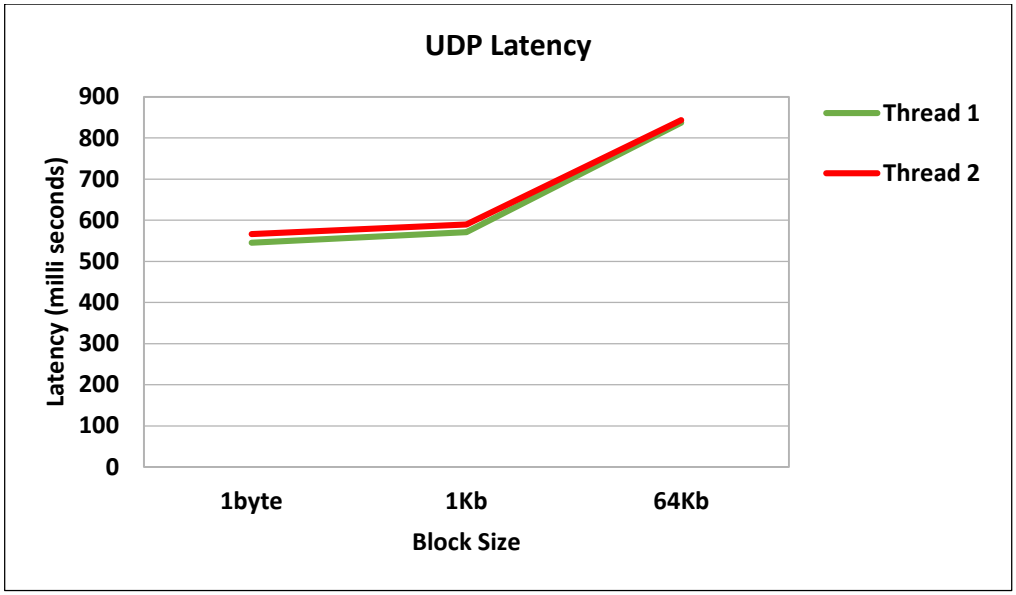
UDP Readings:

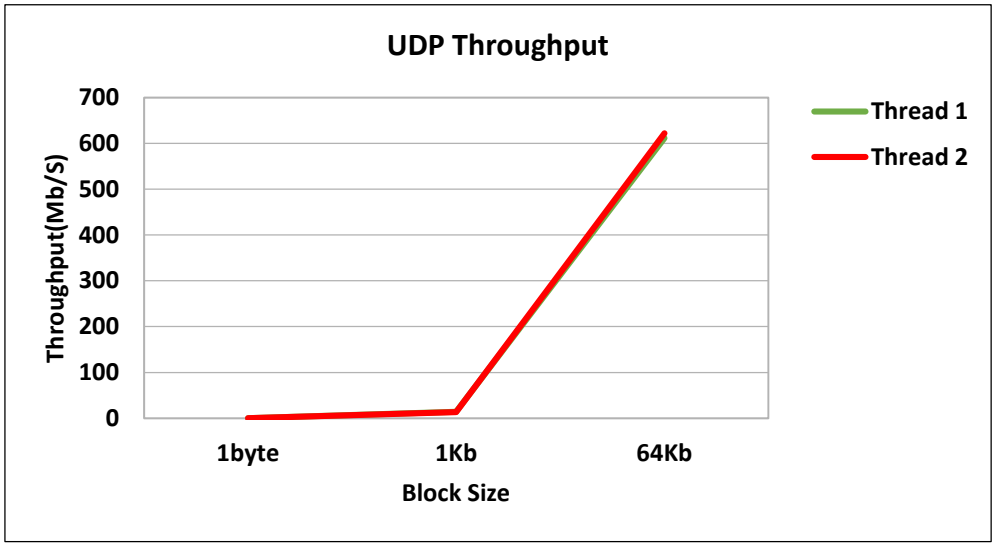
Single Client

Byte Size	Latency in MS	Throughput
1Byte	535	0.01345
1 Kilo Byte	563	14.0105
64 Kilo Byte	822	611.343

Multiple Client

Byte Size	Latency Client 1	Latency Client 2	Average Latency in milli Seconds	Throughput Client 1	Throughput client 2	Average Throughput in Mb/S
1Byte	547	566.5	566.75	0.00177	0.00176	0.01408
1 Kilo Byte	571	589.5	580.25	1.73461	1.65837	13.5719
64 Kilo Byte	837	843	840	77.8589	77.6228	621.926





Observations:

- Throughput of a system is been calculated in Mega bits per Second.
- As packet size increases throughput of a system also increases for both TCP and UDP systems.
- Latency of a system increases gradually but there is no much significant change.
- As packet size increases from 1KB to 64 KB throughput of TCP and UDP systems increases by a great amount.

Theoretical Performance of a Network:

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

Extra Credit: Iperf Benchmarking system

- I have evaluated Iperf benchmarking system on amazon t2.micro instance.
- Also I have compared these benchmarking value with my implementation.
- Following are the results observed after running the Iperf benchmark for 64 Kb Window size**.**

- For installing Iperf

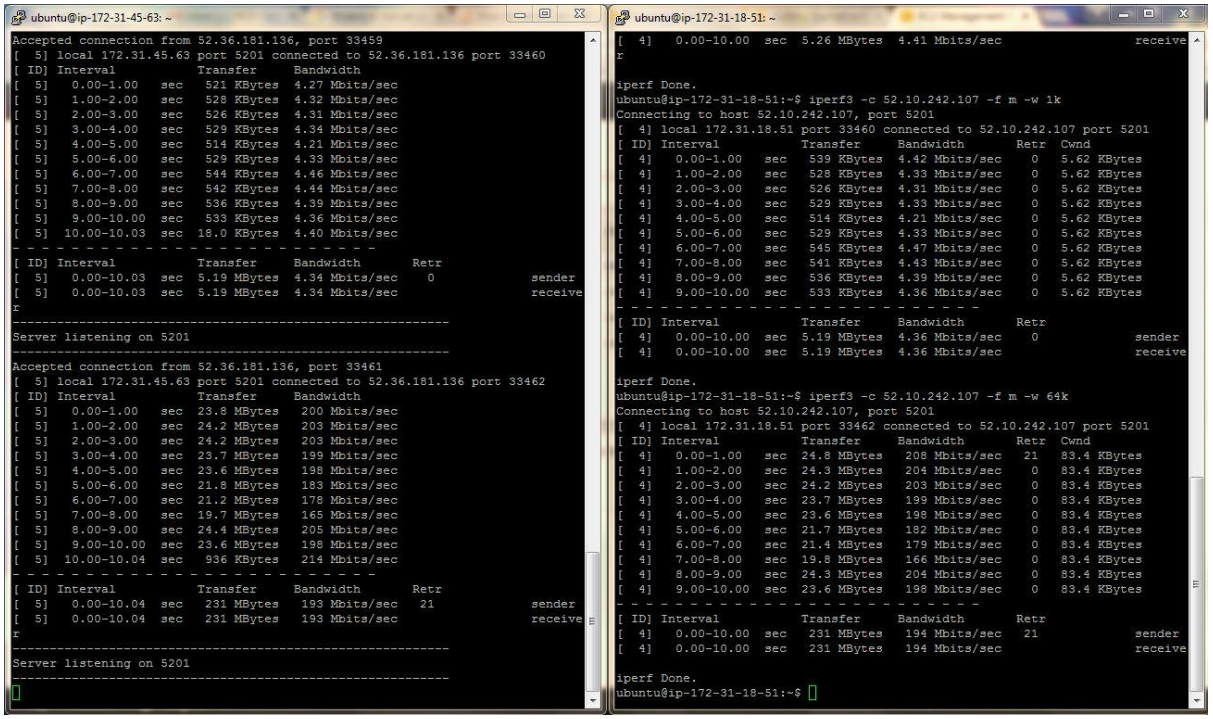
  Sudo apt-get install iperf3

- For TCP :
- For running a Server :

  Iperf3 –s

- For running a client :

  Iperf3 –c @"public_ip"–f m –w 64k

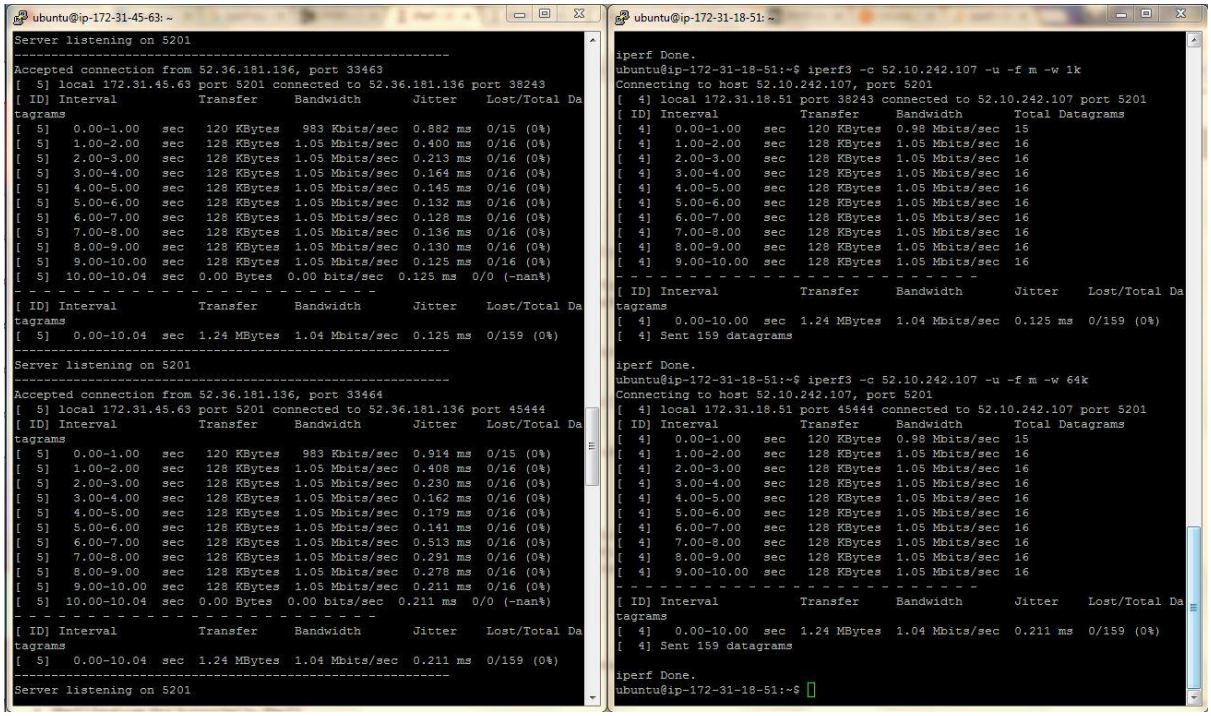


- **For UDP:**
- For running a Server :

  Iperf3 –s

- For running a client :

  Iperf3 –c @"public_ip" –u –f m –w 64k .



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