CS 553 Cloud Computing
Programming Assignment # 1 – Benchmarking

Performance Evaluation

Amit Gupta – A20376501
Saloni Chauhan – A20377221

The following document contains the experimental results obtained of all the benchmarks. The experiments were performed twice to obtain the results.

1. INTRODUCTION

The experiments were performed on Chameleon. All the programs are done in C language/Cuda for one specific benchmark.

a. **For CPU Benchmark**, we performed the experiments using 1, 2, 4 and 8 threads to get the results. The processor speed has been calculated in GFLOPS and GIOPS at different thread counts taken into consideration. However, we have even implemented the code to take the results of the 60 samples for IOPS and FLOPS at the time interval of 1 sec to 10 minutes giving us the broad width of results needed to understand the benchmark and the operations occurring at the background.

b. **For Disk Benchmark,** we measured the throughput value obtained for different sequential and random read/write processes. The throughputs have been collected for the sequential read, sequential write, random read and random write. The experiment also contained different block sizes (8KB/8MB/80MB) taken into consideration along with the thread counts varying from 1, 2, 4 and 8 thread counts.

c. **For Network Benchmark,** we have measured the throughput and latency for both protocols and taken into consideration varied packet sizes and thread counts varying from 1, 2, 4 and 8 thread counts.

d. **For Memory Benchmark,** we have measured the throughput value considering the same thread count t different thread level of 1, 2, 4 and 8 thread at different block sizes of memory taken into consideration so that we can perform different experiments to obtain results.

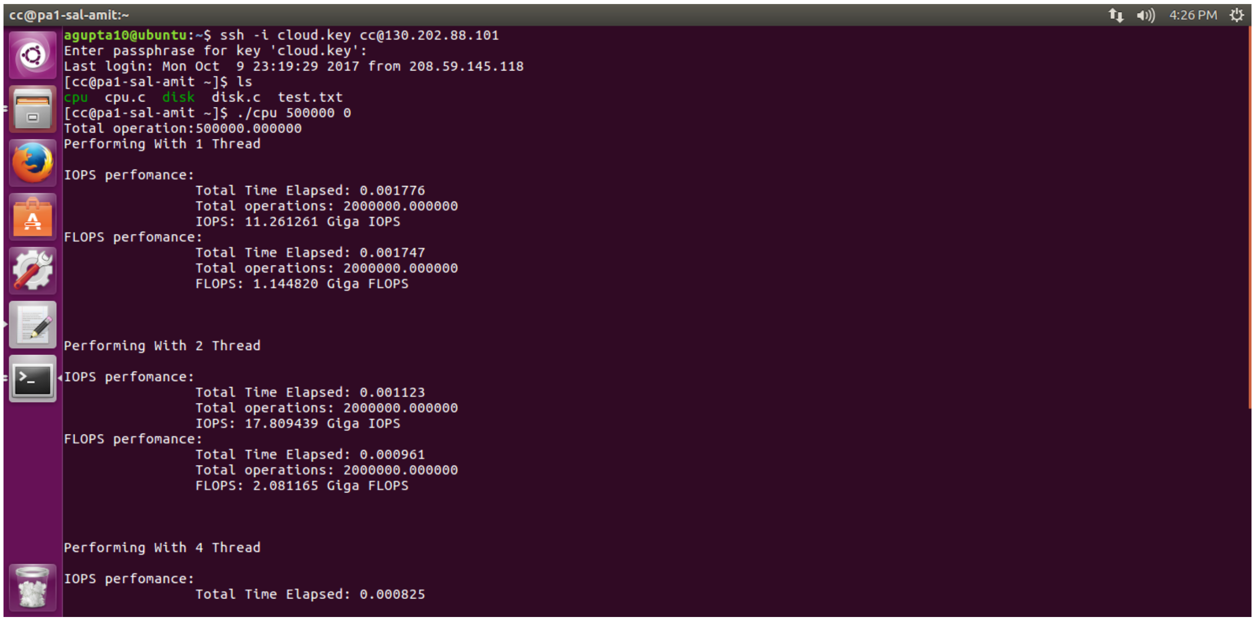
2. Experiment Results and Analysis

The following section contains the experiments performed n each benchmark and explanations to the results obtained.

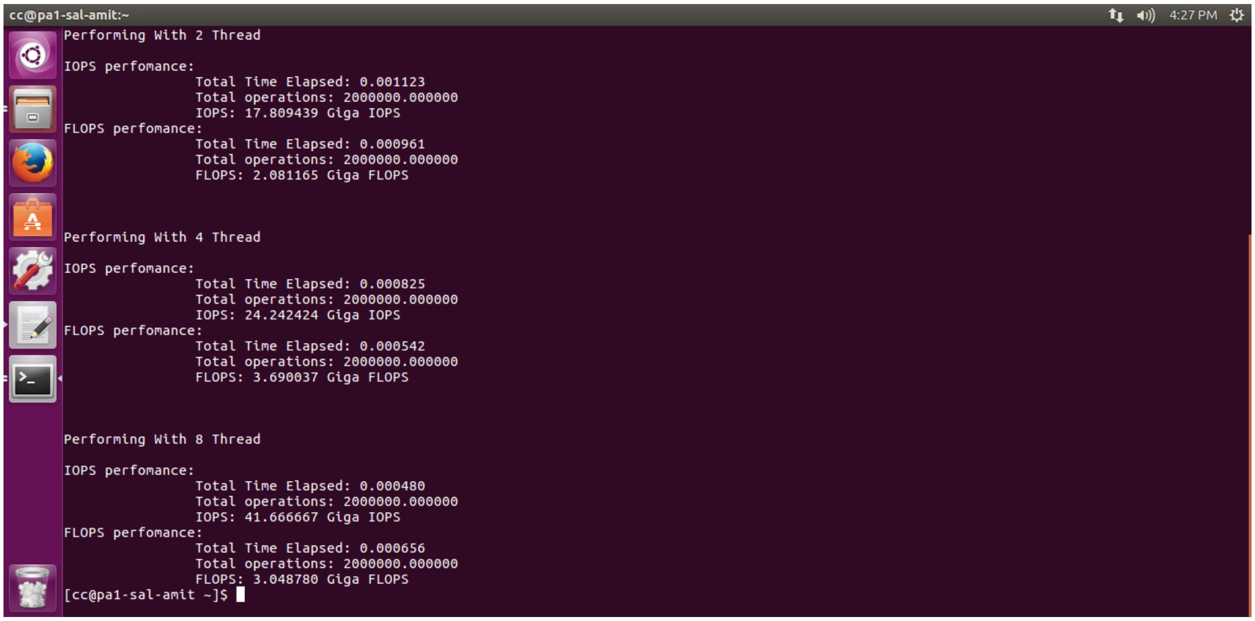
a. CPU Benchmarking

- For CPU benchmarking the observation is with respect to the different number of threads given as an input to the program so that the results are obtained. The values have been calculated in terms of GIOPS and GFLOPS.

- Following are the snapshots for the IOPS and FLOPS normal data for different thread counts.

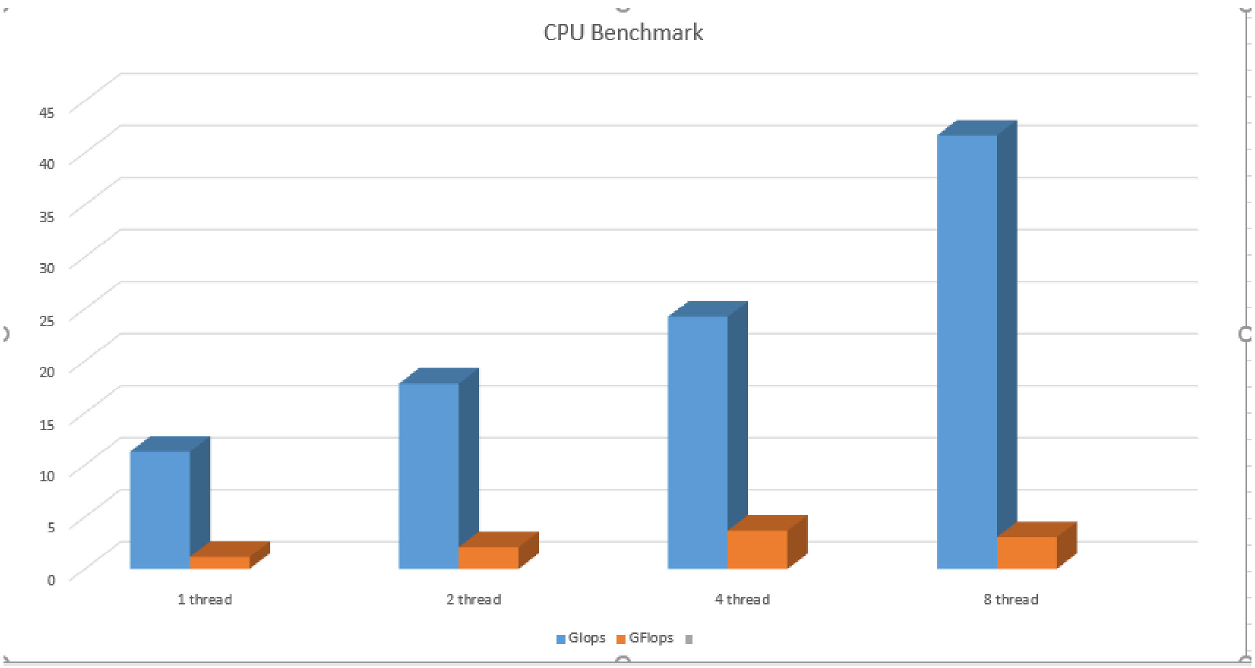


```
agupta10@ubuntu:~$ ssh -i cloud.key cc@130.202.88.101
Enter passphrase for key 'cloud.key':
Last login: Mon Oct  9 23:19:29 2017 from 208.59.145.118
[cc@pa1-sal-amit ~]$ ls
cpu  cpu.c  disk  disk.c  test.txt
[cc@pa1-sal-amit ~]$ ./cpu 500000 0
Total operation:500000.000000
Performing With 1 Thread

IOPS perfomance:
                Total Time Elapsed: 0.001776
                Total operations: 2000000.000000
                IOPS: 11.261261 Giga IOPS
FLOPS perfomance:
                Total Time Elapsed: 0.001747
                Total operations: 2000000.000000
                FLOPS: 1.144820 Giga FLOPS



Performing With 2 Thread

IOPS perfomance:
                Total Time Elapsed: 0.001123
                Total operations: 2000000.000000
                IOPS: 17.809439 Giga IOPS
FLOPS perfomance:
                Total Time Elapsed: 0.000961
                Total operations: 2000000.000000
                FLOPS: 2.081165 Giga FLOPS



Performing With 4 Thread

IOPS perfomance:
                Total Time Elapsed: 0.000825
```



```
Performing With 2 Thread

IOPS perfomance:
                Total Time Elapsed: 0.001123
                Total operations: 2000000.000000
                IOPS: 17.809439 Giga IOPS
FLOPS perfomance:
                Total Time Elapsed: 0.000961
                Total operations: 2000000.000000
                FLOPS: 2.081165 Giga FLOPS



Performing With 4 Thread

IOPS perfomance:
                Total Time Elapsed: 0.000825
                Total operations: 2000000.000000
                IOPS: 24.242424 Giga IOPS
FLOPS perfomance:
                Total Time Elapsed: 0.000542
                Total operations: 2000000.000000
                FLOPS: 3.690037 Giga FLOPS



Performing With 8 Thread

IOPS perfomance:
                Total Time Elapsed: 0.000480
                Total operations: 2000000.000000
                IOPS: 41.666667 Giga IOPS
FLOPS perfomance:
                Total Time Elapsed: 0.000656
                Total operations: 2000000.000000
                FLOPS: 3.048780 Giga FLOPS
[cc@pa1-sal-amit ~]$ 
```

- The following is the graph that we have obtained after plotting the values obtained for different number of threads. We have performed multiple experiments for 1, 2, 4 and 8 thread counts and according to which this is the result we have obtained as below.

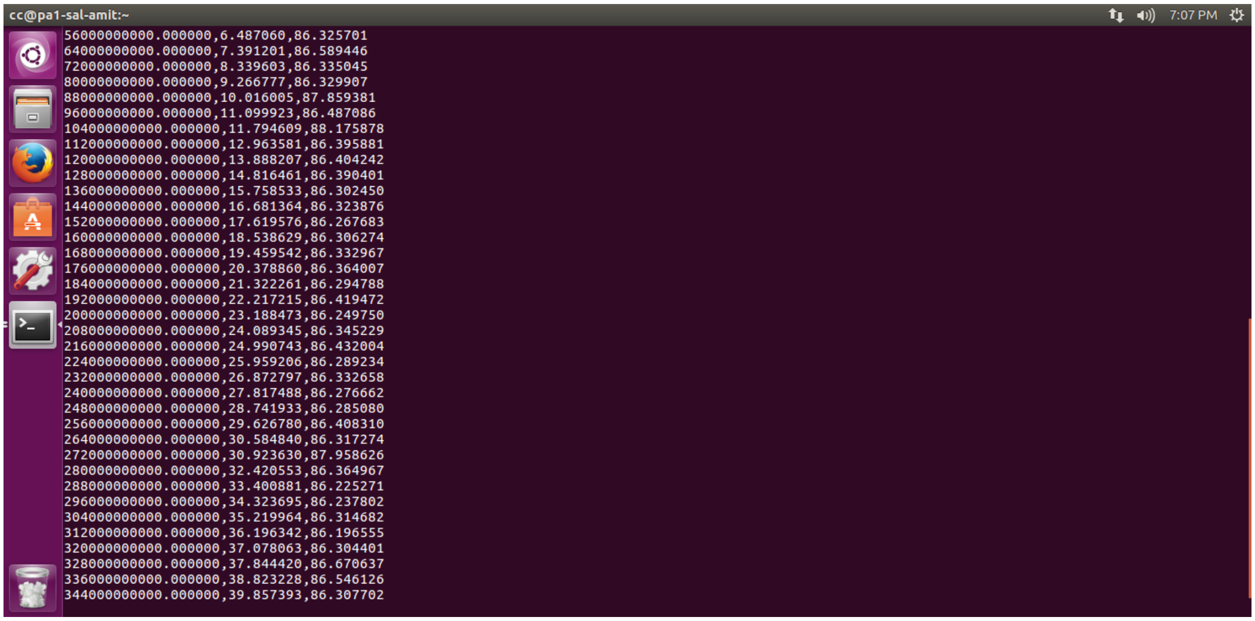


- Peak performance of the processor is given by the formula :

= CPU speed (GHz) * (number of CPU Cores) * (CPU Instruction per cycle) * (number of CPU Nodes)
= 1.8 GHz * 2 * 8 * 2
= 57.6 GFLOPS

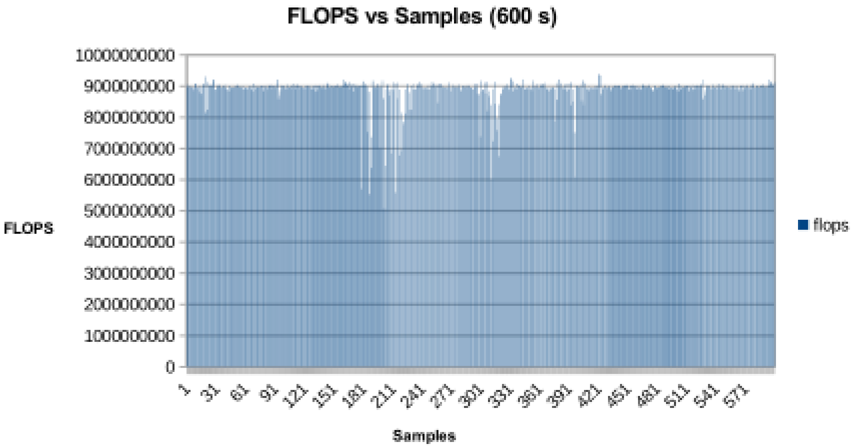
Along with this the efficiency would be calculated as = **(average value/ max performance speed) * 100**
= (30.88/57.6)*100
= 53.61%

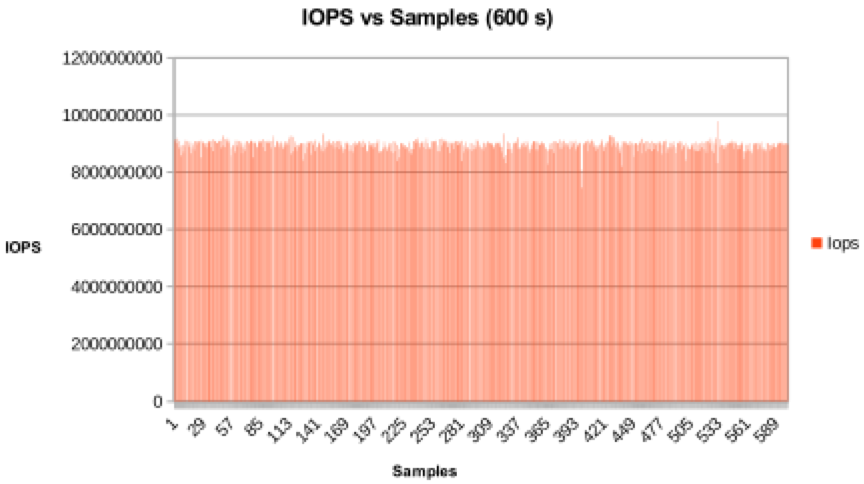
Now, we have taken the **600 samples for IOPS and FLOPS,** the following are the snapshots of program giving the data for 600 samples for both IOPS and FLOPS along with the graph being plotted for the data value obtained. The following snapshots are the data being obtained on chameleon instance for 600 samples for IOPS and FLOPS.



```
56000000000.000000,6.487060,86.325701
64000000000.000000,7.391201,86.589446
72000000000.000000,8.339603,86.335045
80000000000.000000,9.266777,86.329907
88000000000.000000,10.016005,87.859381
96000000000.000000,11.099923,86.487086
104000000000.000000,11.794609,88.175878
112000000000.000000,12.963581,86.395881
120000000000.000000,13.888207,86.404242
128000000000.000000,14.816461,86.390401
136000000000.000000,15.758533,86.302450
144000000000.000000,16.681364,86.323876
152000000000.000000,17.619576,86.267683
160000000000.000000,18.538629,86.306274
168000000000.000000,19.459542,86.332967
176000000000.000000,20.378860,86.364007
184000000000.000000,21.322261,86.294788
192000000000.000000,22.217215,86.419472
200000000000.000000,23.188473,86.249750
208000000000.000000,24.089345,86.345229
216000000000.000000,24.990743,86.432004
224000000000.000000,25.959206,86.289234
232000000000.000000,26.872797,86.332658
240000000000.000000,27.817488,86.276662
248000000000.000000,28.741933,86.285080
256000000000.000000,29.626780,86.408310
264000000000.000000,30.584840,86.317274
272000000000.000000,30.923630,87.958626
280000000000.000000,32.420553,86.364967
288000000000.000000,33.400881,86.225271
296000000000.000000,34.323695,86.237802
304000000000.000000,35.219964,86.314682
312000000000.000000,36.196342,86.196555
320000000000.000000,37.078063,86.304401
328000000000.000000,37.844420,86.670637
336000000000.000000,38.823228,86.546126
344000000000.000000,39.857393,86.307702
```

The following is the output graph generated using these 600 sample data taken into consideration.





The graph output helps here to observe that there are some glitches that can seen in the 60 samples. The reason being because the operations being performed were less in number in the time duration that is plotted against the graph as the core could have been used to perform other processes in the system. Such reasons could be one of the greater factors here that would make us observe these glitches at times while getting the output at that time period.

b. Disk Benchmarking

From practical Throughput for reading 8 MB, we are getting around throughput around = Throughput 62.023528 MBps.

The following were the entire results for different 1, 2, 4 and 8 threads with different file sizes as below obtained by running the code.

OUTPUT :

```
Performing with 8KB blocksize

With 1 thread
Sequential read:
        1000.000000 MB in time:21.670381
        Throughput 46.145935 MBps
Sequential write:
	1000.000000 MB in time:2.379779
	Throughput 420.207086 MBps
Random Read:
	1000.000000 MB in time:0.234081
	Throughput 4272.025495 MBps
Random write:
	1000.000000 MB in time:12.873473
	Throughput 77.679116 MBps
Average:
	4000.000000 MB in time:37.157714
	Throughput 107.649249 MBps

Latency: 0.000650



With 2 thread
Sequential read:
	1000.000000 MB in time:0.186645
	Throughput 5357.764741 MBps
Sequential write:
	1000.000000 MB in time:18.772712
	Throughput 53.268808 MBps
Random Read:
	1000.000000 MB in time:0.185776
	Throughput 5382.826630 MBps
Random write:
	1000.000000 MB in time:2.909487
	Throughput 343.703203 MBps
Average:
	4000.000000 MB in time:22.054620
	Throughput 181.367895 MBps

Latency: 0.000681



With 4 thread
Sequential read:
	1000.000000 MB in time:0.155321
	Throughput 6438.279434 MBps
Sequential write:
	1000.000000 MB in time:1.156635
	Throughput 864.576984 MBps
Random Read:
	1000.000000 MB in time:0.151315
	Throughput 6608.730133 MBps
Random write:
	1000.000000 MB in time:3.079716
	Throughput 324.705265 MBps
Average:
	4000.000000 MB in time:4.542987
	Throughput 880.477976 MBps
```

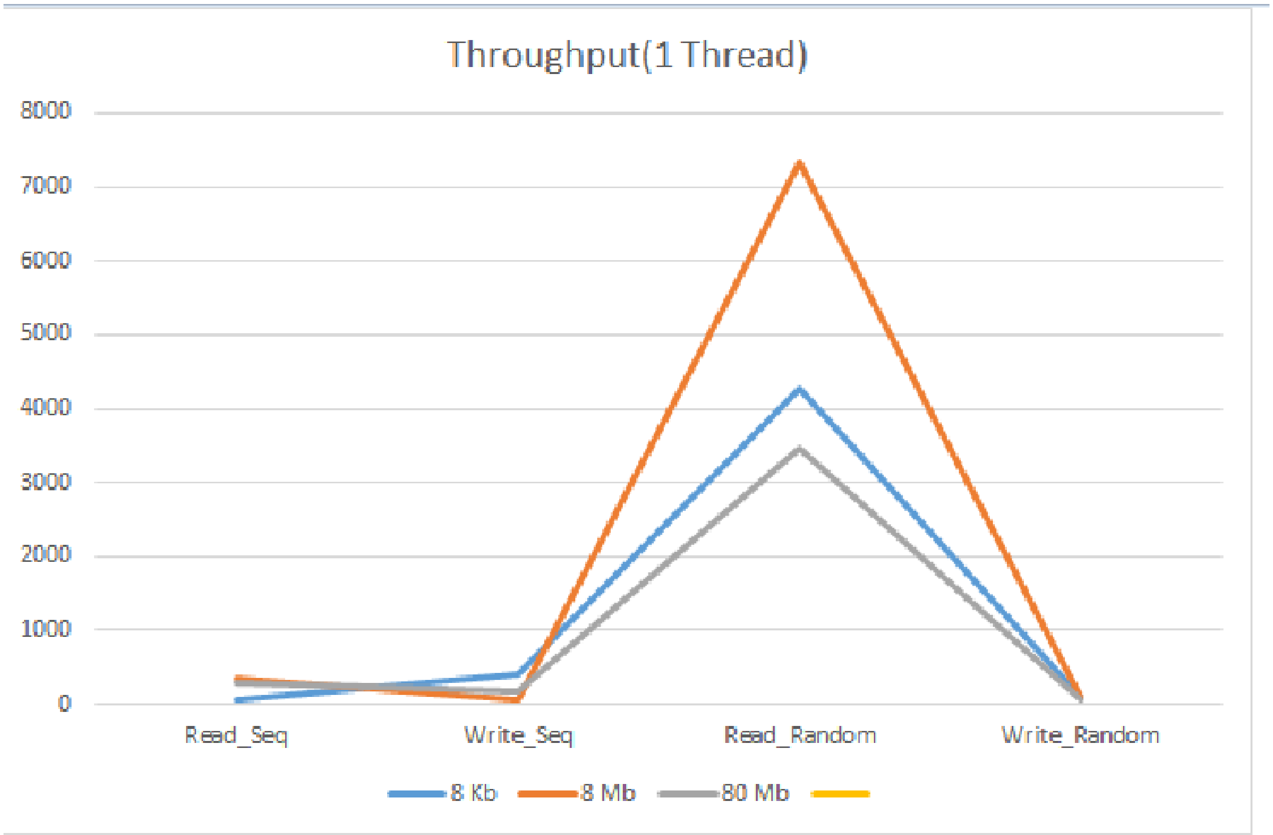
```
Latency: 0.004243



With 8 thread
Sequential read:
        1000.000000 MB in time:13.450510
        Throughput 74.346623 MBps
Sequential write:
        1000.000000 MB in time:1.208106
        Throughput 827.741937 MBps
Random Read:
        1000.000000 MB in time:0.164665
        Throughput 6072.935961 MBps
Random write:
        1000.000000 MB in time:1.135255
        Throughput 880.859366 MBps
Average:
        4000.000000 MB in time:15.958536
        Throughput 250.649558 MBps

Latency: 0.002989



Performing with 8MB blocksize

With 1 thread
Sequential read:
        1000.000000 MB in time:2.826878
        Throughput 353.747137 MBps
Sequential write:
        1000.000000 MB in time:16.122914
        Throughput 62.023528 MBps
Random Read:
        1000.000000 MB in time:0.136251
        Throughput 7339.395674 MBps
Random write:
        1000.000000 MB in time:10.137060
        Throughput 98.647931 MBps
Average:
        4000.000000 MB in time:29.223103
        Throughput 136.878004 MBps

Latency: 0.002465



With 2 thread
Sequential read:
        992.000000 MB in time:3.396930
        Throughput 292.028390 MBps
Sequential write:
        992.000000 MB in time:1.778277
        Throughput 557.843351 MBps
Random Read:
        992.000000 MB in time:0.066656
        Throughput 14882.381181 MBps
Random write:
        992.000000 MB in time:4.554913
        Throughput 217.786816 MBps
Average:
        3968.000000 MB in time:9.796776
        Throughput 405.031206 MBps

Latency: 0.000826



With 4 thread
Sequential read:
        992.000000 MB in time:2.489519
        Throughput 398.470548 MBps
Sequential write:
        992.000000 MB in time:18.799733
        Throughput 52.766707 MBps
Random Read:
        992.000000 MB in time:0.078135
        Throughput 12695.974915 MBps
Random write:
        992.000000 MB in time:1.812355
        Throughput 547.354133 MBps
Average:
        3968.000000 MB in time:23.179742
        Throughput 171.183959 MBps

Latency: 0.003025



With 8 thread
Sequential read:
        960.000000 MB in time:0.096726
        Throughput 9924.942621 MBps
Sequential write:
        960.000000 MB in time:1.073451
        Throughput 894.311897 MBps
Random Read:
        960.000000 MB in time:0.068915
        Throughput 13930.203874 MBps
Random write:
        960.000000 MB in time:6.135781
        Throughput 156.459300 MBps
Average:
	3840.000000 MB in time:7.374873
	Throughput 520.686933 MBps

Latency: 0.003079



Performing with 80MB blocksize

With 1 thread
Sequential read:
	960.000000 MB in time:3.194269
	Throughput 300.538245 MBps
Sequential write:
	960.000000 MB in time:5.624721
	Throughput 170.675132 MBps
Random Read:
	960.000000 MB in time:0.276356
	Throughput 3473.780197 MBps
Random write:
	960.000000 MB in time:16.874718
	Throughput 56.889840 MBps
Average:
	3840.000000 MB in time:25.970064
	Throughput 147.862554 MBps

Latency: 0.000670



With 2 thread
Sequential read:
	960.000000 MB in time:0.138918
	Throughput 6910.551548 MBps
Sequential write:
	960.000000 MB in time:0.877050
	Throughput 1094.578416 MBps
Random Read:
	960.000000 MB in time:0.144307
	Throughput 6652.483940 MBps
Random write:
	960.000000 MB in time:3.728169
	Throughput 257.499057 MBps
Average:
	3840.000000 MB in time:4.888444
	Throughput 785.526028 MBps

Latency: 0.000708



With 4 thread
```

```
Sequential read:
        960.000000 MB in time:0.131043
        Throughput 7325.839610 MBps
Sequential write:
        960.000000 MB in time:0.984349
        Throughput 975.263855 MBps
Random Read:
        960.000000 MB in time:0.133269
        Throughput 7203.475677 MBps
Random write:
        960.000000 MB in time:1.391682
        Throughput 689.812759 MBps
Average:
        3840.000000 MB in time:2.640343
        Throughput 1454.356498 MBps

Latency: 0.001080
```

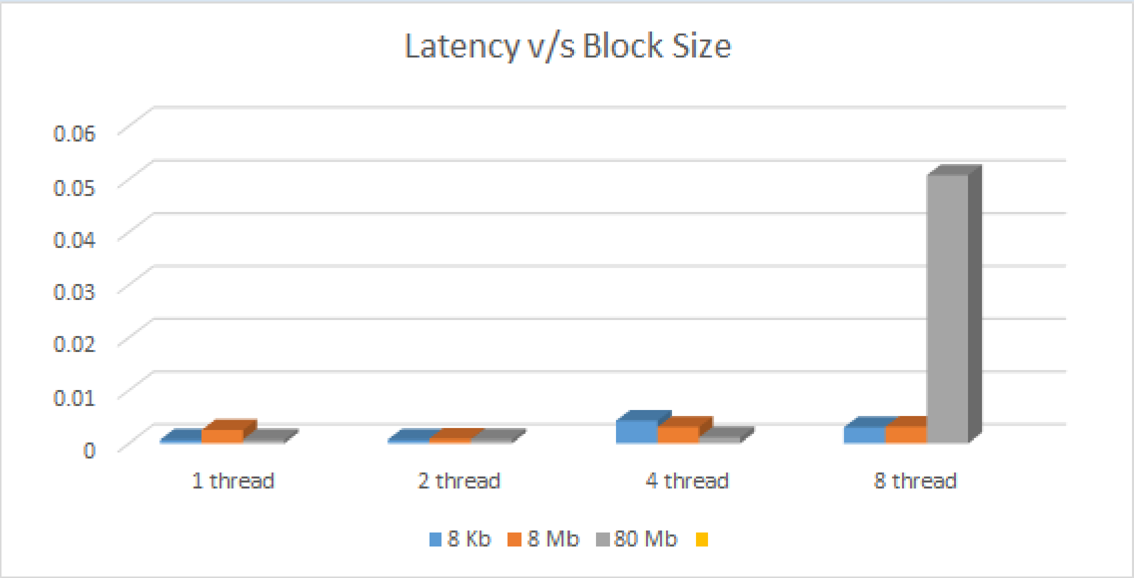
```
With 8 thread
Sequential read:
        640.000000 MB in time:2.316323
        Throughput 276.299981 MBps
Sequential write:
        640.000000 MB in time:4.265640
        Throughput 150.036102 MBps
Random Read:
        640.000000 MB in time:0.100678
        Throughput 6356.900217 MBps
Random write:
        640.000000 MB in time:3.835160
        Throughput 166.877001 MBps
Average:
        2560.000000 MB in time:10.517801
        Throughput 243.396885 MBps

Latency: 0.050754
```

- Here, we have calculated the throughput for single/multithread and different block size for sequential and random read/write.
- Here, we are getting here fewer throughputs for 2 threads when compared to 1 thread because in 2 threads, after completion of one thread, other can access the file. So 2 threads can't access a file at the same time.

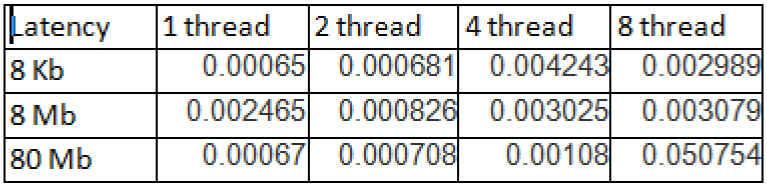
The following graph is plot against 1 thread with different block size having the following throughput values.



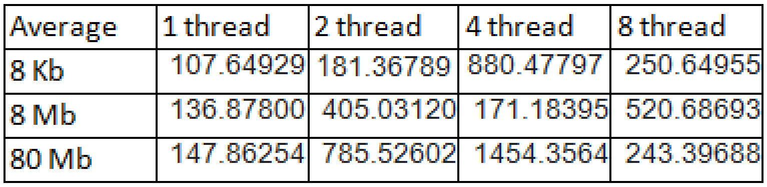
- When we see the graph here showing the values against sequential and random read/write processes we come to observe here that the graph goes upward as we are increasing the block size but drastically falls down as we increase the thread count which can be seen with the throughput value obtained. The reason being we can’t write files simultaneously as it makes it bit difficult for the system to write files simultaneously.
- Comparing the graph and the data obtained we can say that sequential read/write operations are much better in terms of performance with respect to the random read/write processes.
- Comparison also puts light on the fact that read works fine in both sequential and random read processes.
- The graph below shows the trade of f between latency obtained and the block size along with multiple threads taken into consideration here.



- The following data in the table mentioned also provides us with the data of latency and average time that was needed with increasing number of threads counts in the system in terms of throughput.



Latency	1 thread	2 thread	4 thread	8 thread
8 Kb	0.00065	0.000681	0.004243	0.002989
8 Mb	0.002465	0.000826	0.003025	0.003079
80 Mb	0.00067	0.000708	0.00108	0.050754



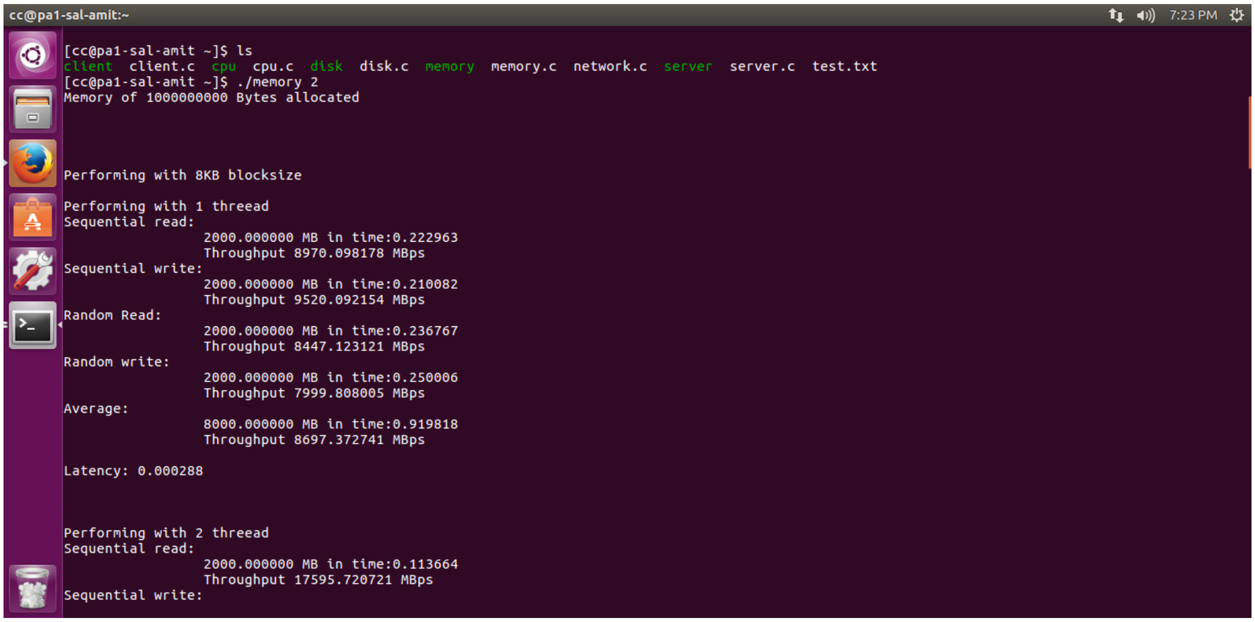
Average	1 thread	2 thread	4 thread	8 thread
8 Kb	107.64929	181.36789	880.47797	250.64955
8 Mb	136.87800	405.03120	171.18395	520.68693
80 Mb	147.86254	785.52602	1454.3564	243.39688

- The following data in the table is the output obtained with respect to the throughput obtained with different threads and different block size.

	Seq_Read				Seq_Write		
	8 Kb	8 Mb	80 Mb		8 Kb	8 Mb	80 Mb
Threads							
1	46.14594	353.7471	300.5382		420.20709	62.023528	170.67513
2	5357.764741	292.028390	6910.551548		53.268808	557.843351	1094.578416
4	6438.279434	398.470548	7325.839610		864.576984	52.766707	975.263855
8	74.346623	9924.942621	276.299981		827.741937	894.311897	150.036102

	Random_Read				Random_Write		
	8 Kb	8 Mb	80 Mb		8 Kb	8 Mb	80 Mb
Threads							
1	4272.025495	7339.395674	3473.780197		77.679116	98.64793	56.88984
2	5382.826630	14882.381181	6652.483940		343.703203	217.786816	257.4991
4	6608.730133	12695.974915	7203.475677		324.705265	547.354133	689.812759
8	6072.93596	13930.203874	6356.900217		880.859366	156.459300	166.877001

c. Memory benchmarking



```
[cc@pa1-sal-amit ~]$ ls
client  client.c  cpu  cpu.c  disk  disk.c  memory  memory.c  network.c  server  server.c  test.txt
[cc@pa1-sal-amit ~]$ ./memory 2
Memory of 1000000000 Bytes allocated



Performing with 8KB blocksize

Performing with 1 threead
Sequential read:
                 2000.000000 MB in time:0.222963
                 Throughput 8970.098178 MBps
Sequential write:
                 2000.000000 MB in time:0.210082
                 Throughput 9520.092154 MBps
Random Read:
                 2000.000000 MB in time:0.236767
                 Throughput 8447.123121 MBps
Random write:
                 2000.000000 MB in time:0.250006
                 Throughput 7999.808005 MBps
Average:
                 8000.000000 MB in time:0.919818
                 Throughput 8697.372741 MBps

Latency: 0.000288



Performing with 2 threead
Sequential read:
                 2000.000000 MB in time:0.113664
                 Throughput 17595.720721 MBps
Sequential write:
```

From practical Throughput for reading 8 KB with 1 thread as input, we are getting around throughput around = 8970.098178 MBps for sequential read of the block size.

The following were the entire results for different 1, 2, 4 and 8 threads with different file sizes as below obtained by running the code.

***Note** – The graphs generated using python code are provided at the end of the document.*

OUTPUT :

```
[cc@pa1-sal-amit ~]$ ./memory 2
Memory of 1000000000 Bytes allocated




Performing with 8KB blocksize

Performing with 1 threead
Sequential read:
 2000.000000 MB in time:0.219102
 Throughput 9128.168616 MBps
Sequential write:
 2000.000000 MB in time:0.198530
 Throughput 10074.044225 MBps
Random Read:
 2000.000000 MB in time:0.230278
 Throughput 8685.154465 MBps
Random write:
 2000.000000 MB in time:0.242529
 Throughput 8246.436509 MBps
Average:
 8000.000000 MB in time:0.890439
 Throughput 8984.332447 MBps

Latency: 0.000312



Performing with 2 threead
Sequential read:
 2000.000000 MB in time:0.111269
 Throughput 17974.458295 MBps
Sequential write:
 2000.000000 MB in time:0.113881
 Throughput 17562.192113 MBps
Random Read:
 2000.000000 MB in time:0.179047
 Throughput 11170.251387 MBps
Random write:
 2000.000000 MB in time:0.176361
 Throughput 11340.375707 MBps
Average:
 8000.000000 MB in time:0.580558
 Throughput 13779.846286 MBps

Latency: 0.000357
```

```
Performing with 4 threead
Sequential read:
 2000.000000 MB in time:0.054224
 Throughput 36884.036589 MBps
Sequential write:
 2000.000000 MB in time:0.054866
 Throughput 36452.447782 MBps
Random Read:
 2000.000000 MB in time:0.154496
 Throughput 12945.318973 MBps
Random write:
 2000.000000 MB in time:0.155750
 Throughput 12841.091493 MBps
Average:
 8000.000000 MB in time:0.419336
 Throughput 19077.780110 MBps

Latency: 0.000349
```

```
Performing with 8 threead
Sequential read:
 2000.000000 MB in time:0.027497
 Throughput 72735.207477 MBps
Sequential write:
 2000.000000 MB in time:0.029948
 Throughput 66782.422866 MBps
Random Read:
 2000.000000 MB in time:0.169891
 Throughput 11772.253975 MBps
Random write:
 2000.000000 MB in time:0.169574
 Throughput 11794.260913 MBps
Average:
 8000.000000 MB in time:0.396910
 Throughput 20155.702804 MBps

Latency: 0.001059
```

```
Performing with 8MB blocksize

Performing with 1 threead
Sequential read:
 2000.000000 MB in time:0.299954
 Throughput 6667.689046 MBps
Sequential write:
 2000.000000 MB in time:0.283628
 Throughput 7051.489980 MBps
```

```
Random Read:
 2000.000000 MB in time:0.296789
 Throughput 6738.794228 MBps
Random write:
 2000.000000 MB in time:0.280274
 Throughput 7135.874180 MBps
Average:
 8000.000000 MB in time:1.160645
 Throughput 6892.719135 MBps

Latency: 0.000646



Performing with 2 threead
Sequential read:
 1984.000000 MB in time:0.176646
 Throughput 11231.502553 MBps
Sequential write:
 1984.000000 MB in time:0.146406
 Throughput 13551.357185 MBps
Random Read:
 1984.000000 MB in time:0.164907
 Throughput 12031.023547 MBps
Random write:
 1984.000000 MB in time:0.157794
 Throughput 12573.355134 MBps
Average:
 7936.000000 MB in time:0.645753
 Throughput 12289.528659 MBps

Latency: 0.000408



Performing with 4 threead
Sequential read:
 1984.000000 MB in time:0.087693
 Throughput 22624.382790 MBps
Sequential write:
 1984.000000 MB in time:0.083255
 Throughput 23830.400577 MBps
Random Read:
 1984.000000 MB in time:0.122532
 Throughput 16191.688702 MBps
Random write:
 1984.000000 MB in time:0.103320
 Throughput 19202.477739 MBps
Average:
 7936.000000 MB in time:0.396800
 Throughput 20000.000000 MBps

Latency: 0.000451
```

```
Performing with 8 threead
Sequential read:
 1920.000000 MB in time:0.060602
 Throughput 31682.122702 MBps
Sequential write:
 1920.000000 MB in time:0.069284
 Throughput 27712.025865 MBps
Random Read:
 1920.000000 MB in time:0.090214
 Throughput 21282.727736 MBps
Random write:
 1920.000000 MB in time:0.114827
 Throughput 16720.806082 MBps
Average:
 7680.000000 MB in time:0.334927
 Throughput 22930.369902 MBps

Latency: 0.001238



Performing with 80MB blocksize

Performing with 1 threead
Sequential read:
 1920.000000 MB in time:0.301484
 Throughput 6368.497167 MBps
Sequential write:
 1920.000000 MB in time:0.360921
 Throughput 5319.723707 MBps
Random Read:
 1920.000000 MB in time:0.288555
 Throughput 6653.844154 MBps
Random write:
 1920.000000 MB in time:0.343722
 Throughput 5585.909543 MBps
Average:
 7680.000000 MB in time:1.294682
 Throughput 5931.958581 MBps

Latency: 0.000340



Performing with 2 threead
Sequential read:
 1920.000000 MB in time:0.173220
 Throughput 11084.170419 MBps
Sequential write:
 1920.000000 MB in time:0.207895
 Throughput 9235.431348 MBps
Random Read:
 1920.000000 MB in time:0.181048
 Throughput 10604.922452 MBps
Random write:
 1920.000000 MB in time:0.200933
 Throughput 9555.423947 MBps
Average:
 7680.000000 MB in time:0.763096
 Throughput 10064.264522 MBps

Latency: 0.000408



Performing with 4 threead
Sequential read:
 1920.000000 MB in time:0.099500
 Throughput 19296.482412 MBps
Sequential write:
 1920.000000 MB in time:0.112267
 Throughput 17102.086989 MBps
Random Read:
 1920.000000 MB in time:0.120234
 Throughput 15968.860722 MBps
Random write:
 1920.000000 MB in time:0.116097
 Throughput 16537.895036 MBps
Average:
 7680.000000 MB in time:0.448098
 Throughput 17139.107963 MBps

Latency: 0.000707



Performing with 8 threead
Sequential read:
 1280.000000 MB in time:0.054219
 Throughput 23607.960309 MBps
Sequential write:
 1280.000000 MB in time:0.069358
 Throughput 18454.972750 MBps
Random Read:
 1280.000000 MB in time:0.047033
 Throughput 27214.934195 MBps
Random write:
 1280.000000 MB in time:0.044085
 Throughput 29034.819099 MBps
Average:
 5120.000000 MB in time:0.214695
 Throughput 23847.784066 MBps
```

Latency: 0.001074

- Here, we have calculated the throughput for single/multithread and different block size for sequential and random read/write.
- The same reason is one standing here too with respect to memory utilization as well. The utilization would be more in terms of sequential read/write processes and the throughput would certainly take time with increase in thread count because the system can't perform same task at one point of time.
- The following table gives the output in terms of the latency and the average time taken in the process with respect to number of threads in increasing manner.
- We can observe that the latency and average time is increasing over a period of time but gradually decreases afterwards.

Latency	1 thread	2 thread	4 thread	8 thread
8 Kb	0.000656	0.001382	0.001472	0.003663
8 Mb	0.000588	0.000792	0.001392	0.003864
80 Mb	0.000618	0.000767	0.001463	0.004078

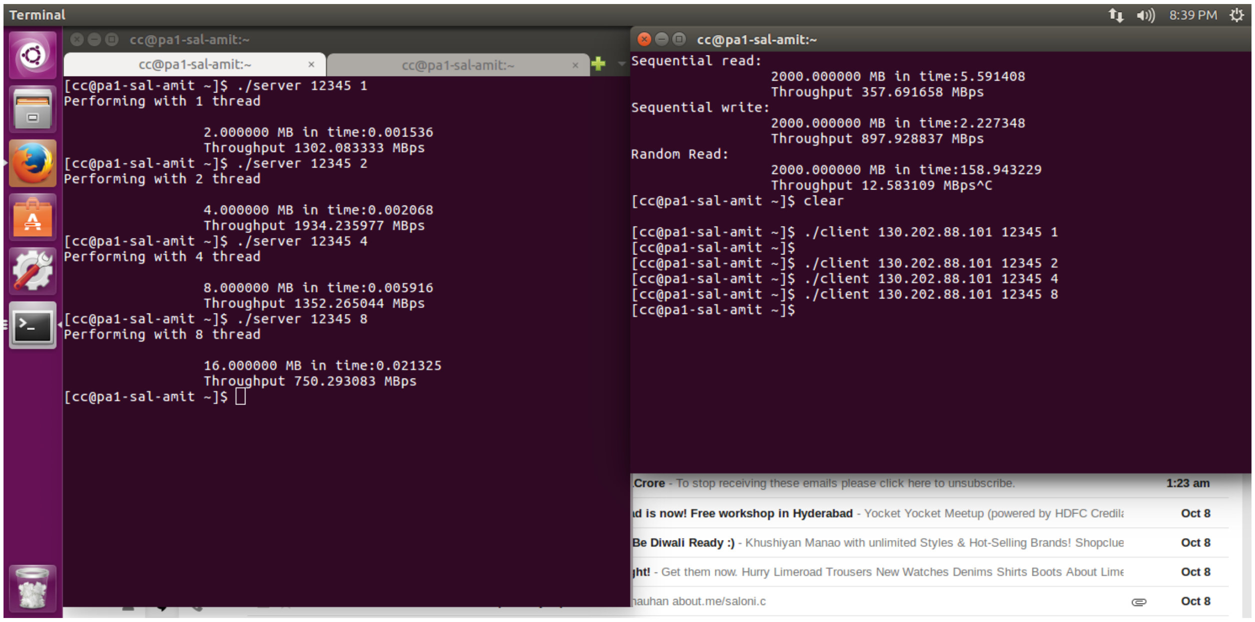
Average	1 thread	2 thread	4 thread	8 thread
8 Kb	8424.5270	13237.888	18181.157	20059.546
8 Mb	7698.7139	13006.791	20278.460	26185.793
80 Mb	6760.2456	12258.595	19941.535	31350.688

- The following table gives us the data of the throughput values obtained with different threads and different block sizes taken into consideration.

	Seq_Read				Seq_Write		
	8 Kb	8 Mb	80 Mb		8 Kb	8 Mb	80 Mb
Threads							
1	8803.041627	7965.246	8001.2535		9303.918066	7443.67741	5825.6633
2	15715.95609	13088.659	12953.210575		16697.835960	13016.835	11275.8772
4	33601.13474	20607.848	23846.902		34186.982281	22092.114	18359.045
8	75395.4419	42704.436	38954.3199		827.741937	28571.598	28054.302

	Random_Read				Random_Write		
	8 Kb	8 Mb	80 Mb		8 Kb	8 Mb	80 Mb
Threads							
1	8099.763163	7946.327	8019.96972		7676.659118	7471.864	5869.5420
2	11194.49768	13117.511	13917.82394		11196.227	12808.678	11291.075
4	12385.695513	18746.87001	22057.5564		12459.754991	19951.649	16993.165
8	11556.445145	23929.5871	37382.3041		11768.430244	19029.9483	25302.3432

d. Networking Benchmark



```
[cc@pa1-sal-amit ~]$ ./server 12345 1
Performing with 1 thread

                  2.000000 MB in time:0.001536
                  Throughput 1302.083333 MBps
[cc@pa1-sal-amit ~]$ ./server 12345 2
Performing with 2 thread

                  4.000000 MB in time:0.002068
                  Throughput 1934.235977 MBps
[cc@pa1-sal-amit ~]$ ./server 12345 4
Performing with 4 thread

                  8.000000 MB in time:0.005916
                  Throughput 1352.265044 MBps
[cc@pa1-sal-amit ~]$ ./server 12345 8
Performing with 8 thread

                  16.000000 MB in time:0.021325
                  Throughput 750.293083 MBps
[cc@pa1-sal-amit ~]$ 
```

```
Sequential read:
                  2000.000000 MB in time:5.591408
                  Throughput 357.691658 MBps
Sequential write:
                  2000.000000 MB in time:2.227348
                  Throughput 897.928837 MBps
Random Read:
                  2000.000000 MB in time:158.943229
                  Throughput 12.583109 MBps^C
[cc@pa1-sal-amit ~]$ clear

[cc@pa1-sal-amit ~]$ ./client 130.202.88.101 12345 1
[cc@pa1-sal-amit ~]$
[cc@pa1-sal-amit ~]$ ./client 130.202.88.101 12345 2
[cc@pa1-sal-amit ~]$ ./client 130.202.88.101 12345 4
[cc@pa1-sal-amit ~]$ ./client 130.202.88.101 12345 8
[cc@pa1-sal-amit ~]$
```

Crore - To stop receiving these emails please click here to unsubscribe. 1:23 am

d is now! Free workshop in Hyderabad - Yocket Yocket Meetup (powered by HDFC Credil Oct 8

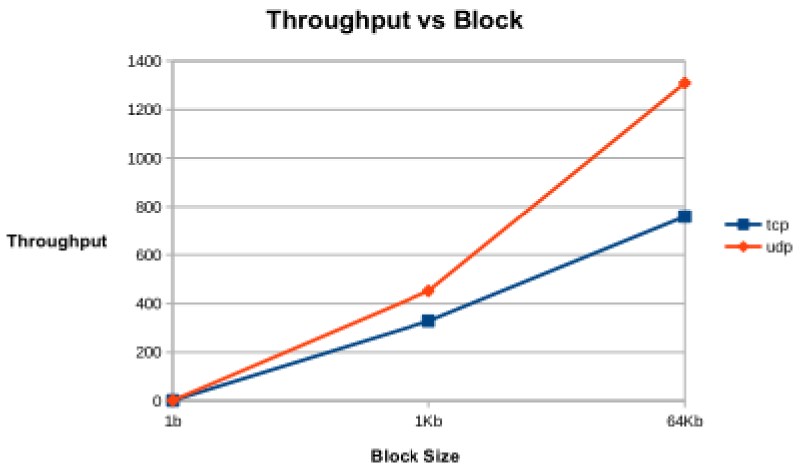
Be Diwali Ready :) - Khushiyan Manao with unlimited Styles & Hot-Selling Brands! Shopclue Oct 8

ht! - Get them now. Hurry Limeroad Trousers New Watches Denims Shirts Boots About Lime Oct 8

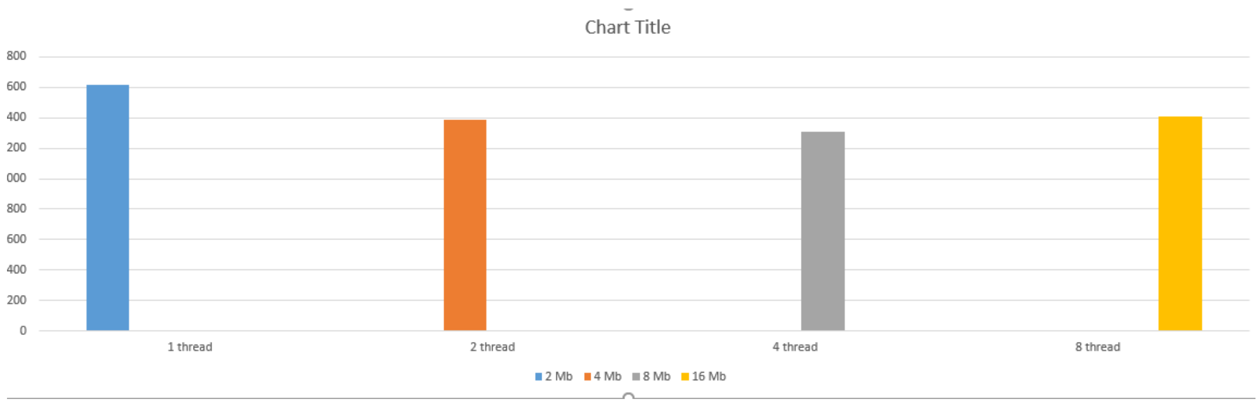
auhan about.me/saloni.c Oct 8

The following are the graphs and the observations in the output generated for TCP and UDP based protocols included based on varying thread size of 1, 2, 4 and 8 threads.

- The graph shows the increase in the plot when the block sizes being transferred between the client and server in both TCP and UDP protocols, meaning as the thread count increases the throughput is better as compared to others.



- We have even observed that the TCP throughput is lesser which can be seen in the above graph as well as compared to the UDP protocol because of the reason that UDP being a connectionless protocol which doesn't give the security of the bytes of data transferring that it will be received properly or not at the end. Thus the throughput for UDP is comparatively higher in comparison to TCP protocol.
- With respect to UDP being a connectionless protocol, it also does not provide the security of the packages transferred over the client and server will be received as well.
- One more thing to observe here that the packets can be sent in one go in the TCP protocol while in the UDP we need to send the data one by one which makes it difficult to work with no security of data being transferred.

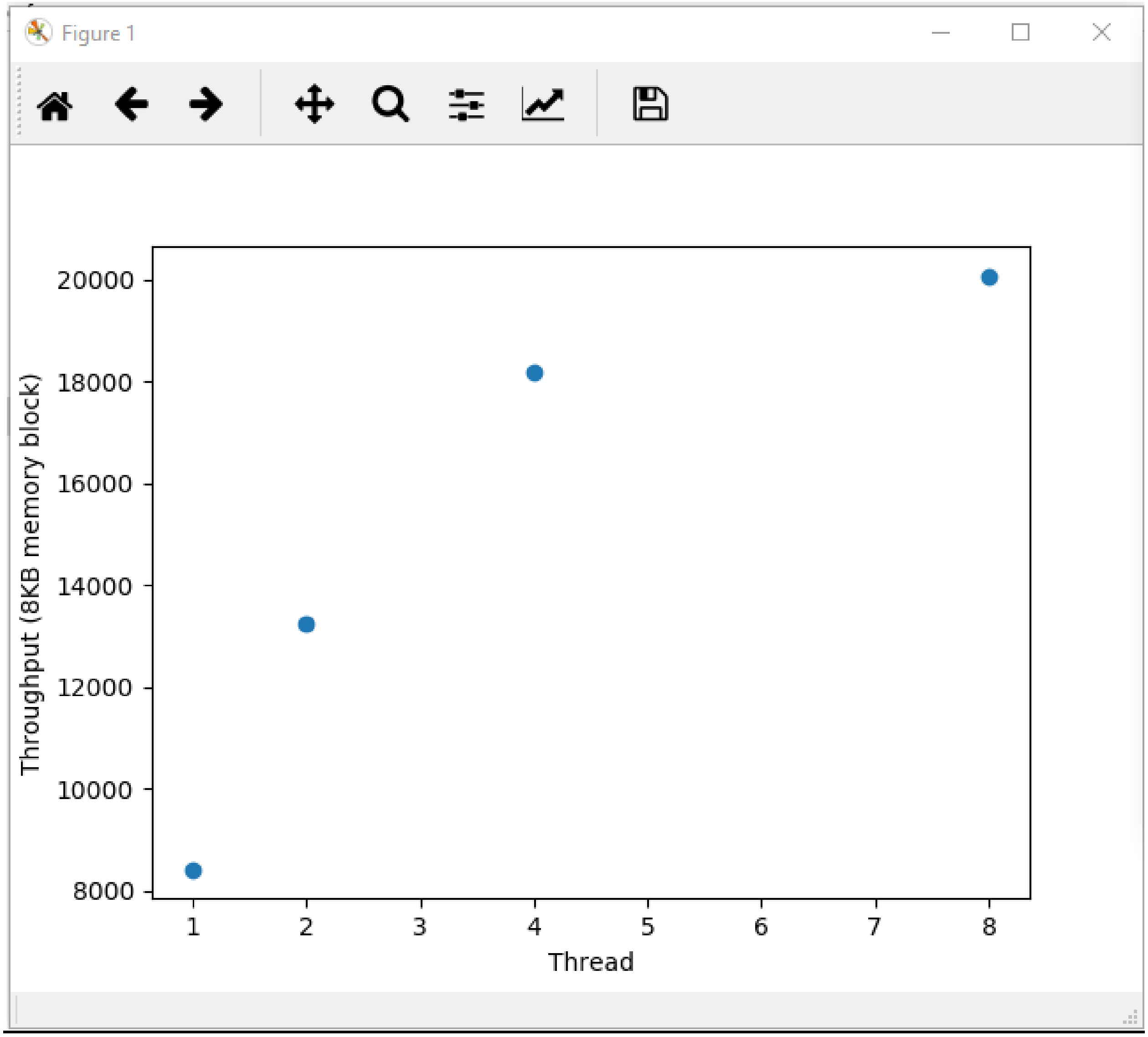


3. Notes and Important Data

- **Following are some of the graphs obtained for memory benchmark using the python code to generate the graph as required in the assignment details.** *(The graph is obtained by saving the output in txt files and then using the python script to obtain the graph. We have done for 8 KB, 8 MB and 80 MB data that we got in the output.)*
- **Comparing and reading all the benchmark data obtained on chameleon by running the code we have observed that the output for 8 threads are most efficient. Reason being multithreading increases the performance of the system. The threads run asynchronously on the system which makes it pretty much easy or the system to handle the task efficiently. However, we even see a downfall with increase I too many number of threads as it decreases system performance.**
- **Average calculations in the code itself which is being displayed in the output. To have all such values run the code and we can see the values being obtained.**
- **Latency values have also been added to the code so the values are obtained while running the program. It is calculated and displayed in the program output itself.**

- **The experiments are done using the block sizes and file sizes of 8 KB, 8 MB and 80 MB size.**

Memory 8 KB



Memory 8 MB

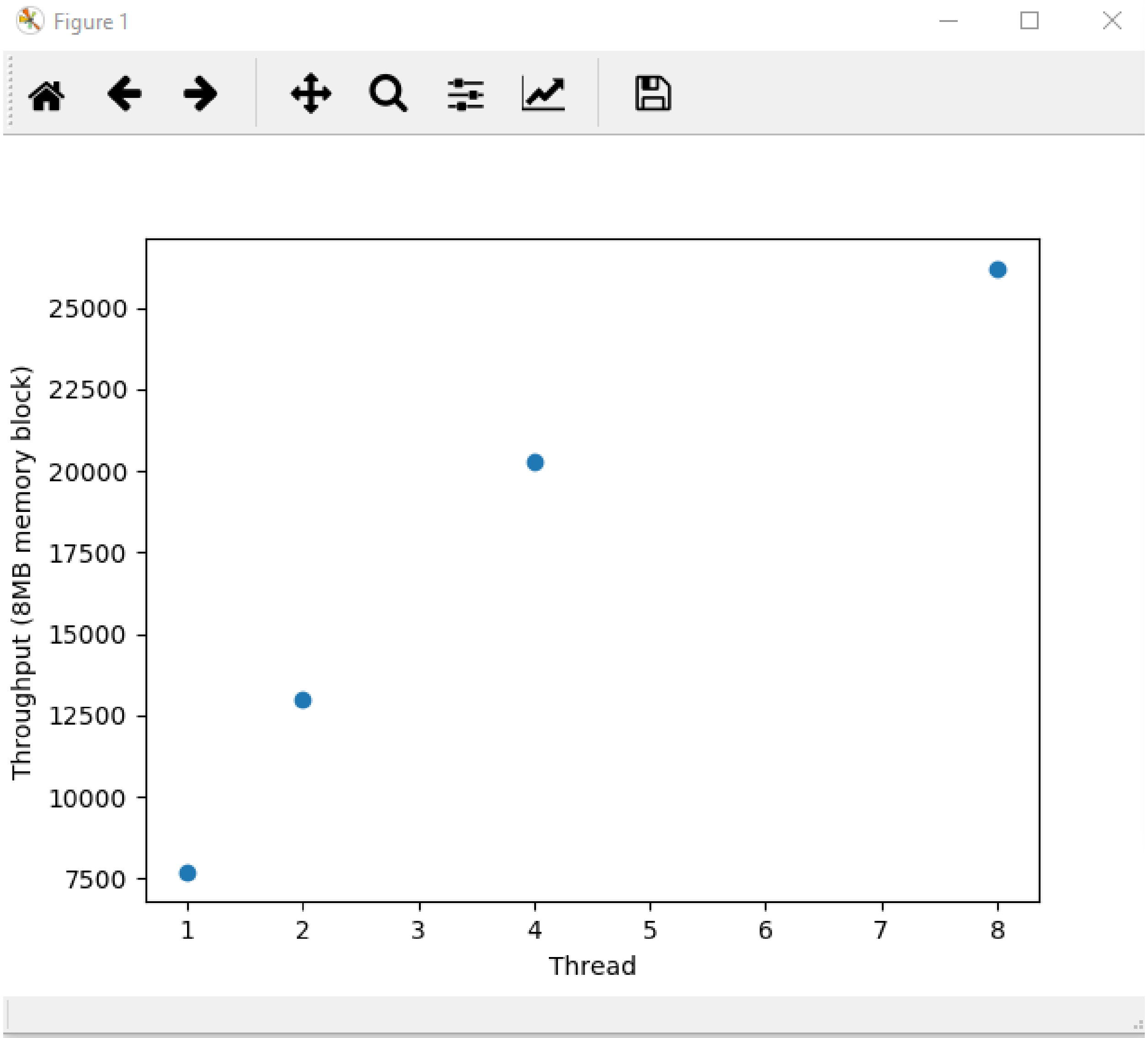


Figure 1
Throughput (8MB memory block)
25000
22500
20000
17500
15000
12500
10000
7500
1
2
3
4
5
6
7
8
Thread

Memory 80 MB

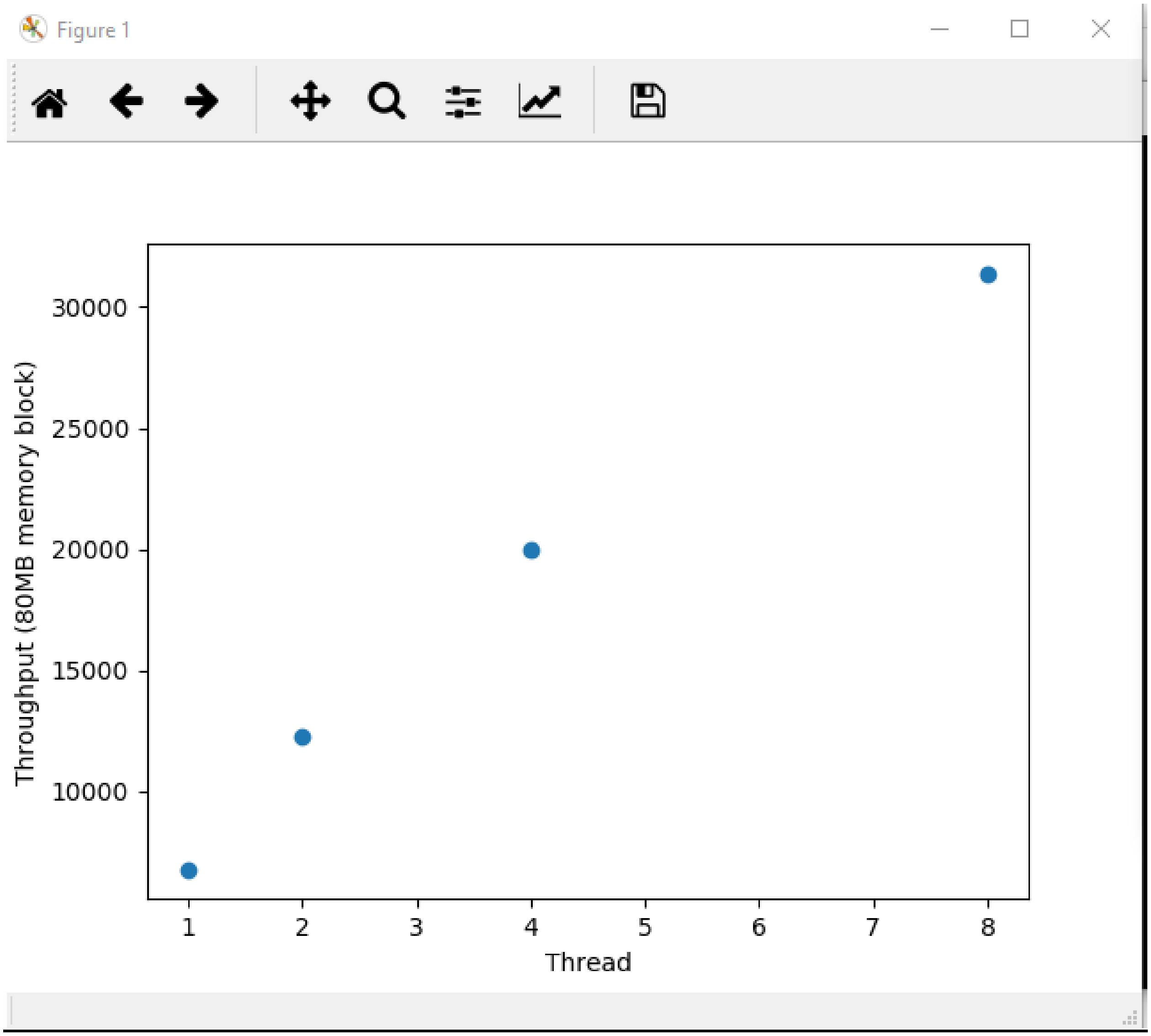


Figure 1
Throughput (80MB memory block)
30000
25000
20000
15000
10000
1
2
3
4
5
6
7
8
Thread