Parth Shah
A20352487

Report

Cloud Computing Assignment-1

Experiment Envornment

Amazon EC2 t2 micro
Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
cpu cores : 1
cpu MHz : 2500.040
cache size : 25600 KB

ec2-54-86-46-69.compute-1.amazonaws.com - PuTTY

```
-bash-4.2$ less /proc/cpuinfo
-bash-4.2$ less /proc/cpuinfo
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 62
model name      : Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
stepping        : 4
microcode       : 0x416
cpu MHz         : 2500.040
cache size      : 25600 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
```

Parth Shah
A20352487

1) CPU Experiments

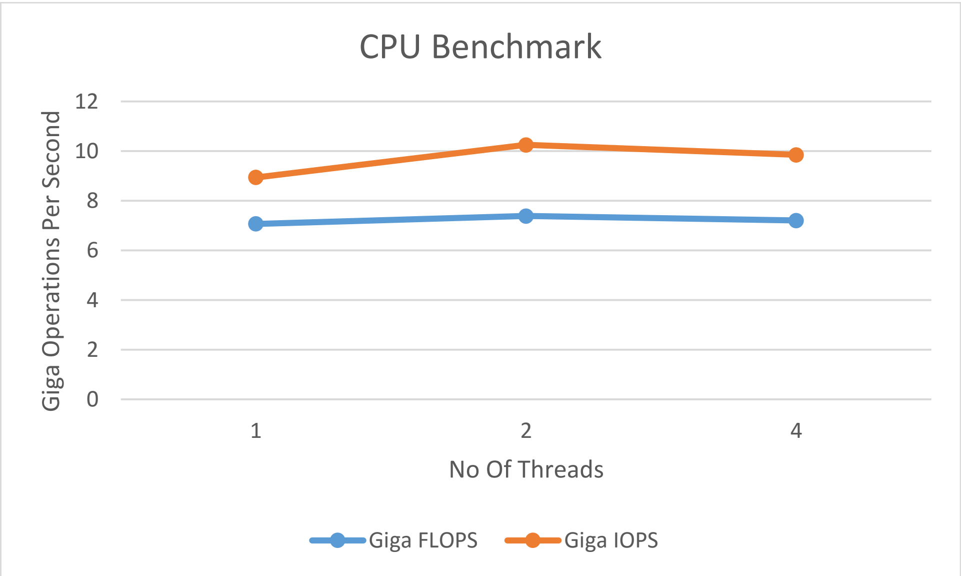
- Measuring Giga Flops and Giga Iops

In this experiment the intent was to measure Gipa Flops and Gipa Iops i.e. the number of floating point and Integer operations CPU can handle in a single second.

The processor speed was measured with with varying level of concurrency with 1,2 and 4 threads.

Each experiment was carried out three times and the average results are documented in the following table.

	Average Giga Flops	**Standard Deviation Flops**	**Average Giga Iops**	**Standard Deviation Iops**
1	7.07	0.01163	8.94	0.0652
2	7.39	0.06121	10.25	0.03621
4	7.21	0.21645	9.85	0.01939



Evaluation :

- As we can see from the above graph , we obtain more Integer operations per second than float operations per seconds as integer operations can be calculated faster by CPU.
- The pattern for operations per second almost remains constant as there is only one underlying hardware thread , which means increasing the number of threads will not make any major changes to no of operations performed in one second.

- Theoritical peak performance is measured by using following formula

Theoritical Performance = Ghz*IPC*Cores
= 2.5 *16*1
= 40 Giga Flops

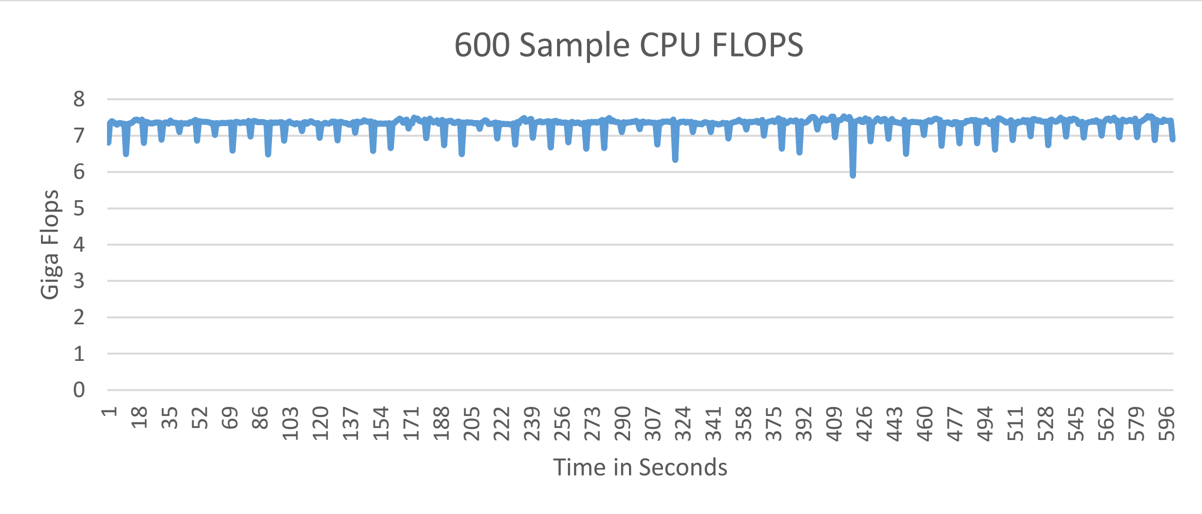
Efficiency achieved as compared to theoritical performance = (7.22/40)*100
=18.05%

Efficiency achieved = 18.05 % as compared to theoritical performance

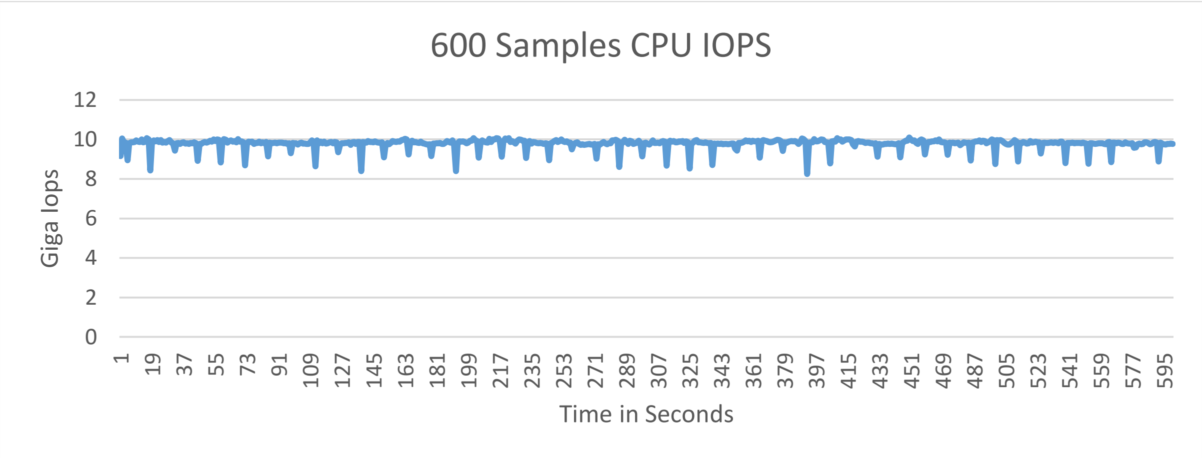
- Running the Benchmark and taking 600 Samples for FLOPS and IOPS

The CPU Benchmark was performed with 4 threads to obtain 600 samples for flops and Iops and the results obtained are shown in the below gragh. The 600 Samples are place in Samples_FLOPS_IOPS.xlsx file.

FLOPS



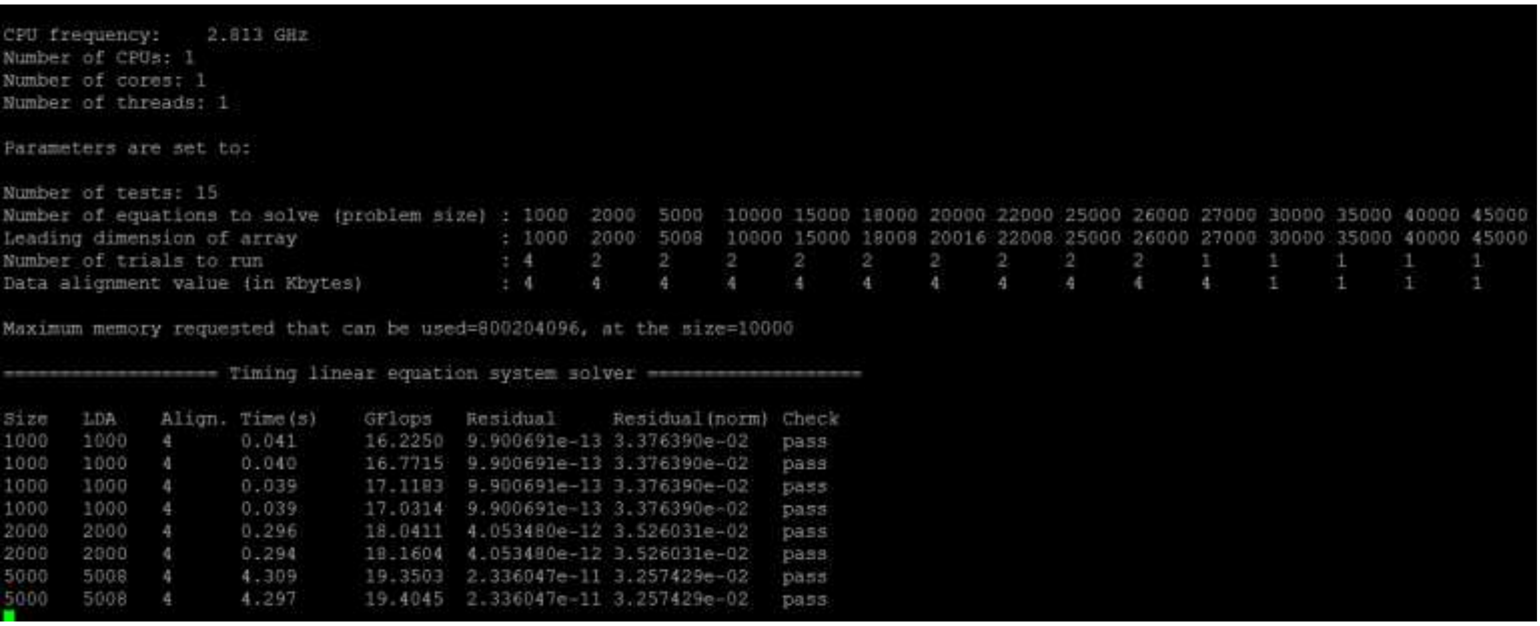
IOPS



Evaluation : As we see from the above graphs we obtain a constant pattern of number of floating point operations per second & Integer operations per second when carried out with 4 threads . The few negative spikes which we see in the graphs gives us the indication that some part of CPU might be running many background processes, because of which there is a slight dip in IOPS and Flops at some intervals.

Parth Shah
A20352487

- Linpack Benchmark



```
CPU frequency:    2.813 GHz
Number of CPUs: 1
Number of cores: 1
Number of threads: 1

Parameters are set to:

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 18008 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
1000   1000   4      0.041      16.2250   9.900691e-13 3.376390e-02   pass
1000   1000   4      0.040      16.7715   9.900691e-13 3.376390e-02   pass
1000   1000   4      0.039      17.1183   9.900691e-13 3.376390e-02   pass
1000   1000   4      0.039      17.0314   9.900691e-13 3.376390e-02   pass
2000   2000   4      0.296      18.0411   4.053480e-12 3.526031e-02   pass
2000   2000   4      0.294      18.1604   4.053480e-12 3.526031e-02   pass
5000   5008   4      4.309      19.3503   2.336047e-11 3.257429e-02   pass
5000   5008   4      4.297      19.4045   2.336047e-11 3.257429e-02   pass
```

- The best Performance achieved using linpack Benchmark was around 19.5 Giga Flops.
- Linpack Benchmark efficiency achieved as compared to theoritical performance is

  = (Linpack Performance /Theoritical Performance) *100

  = (19.5/40)*100

  = 48.75 %

2) Disk Experiments

- For disk experiments , we need to perform sequential and Random read write operations using varying block sizes of (1B,1KB,1MB) along with varying concurrency of 1 and 2 threads.

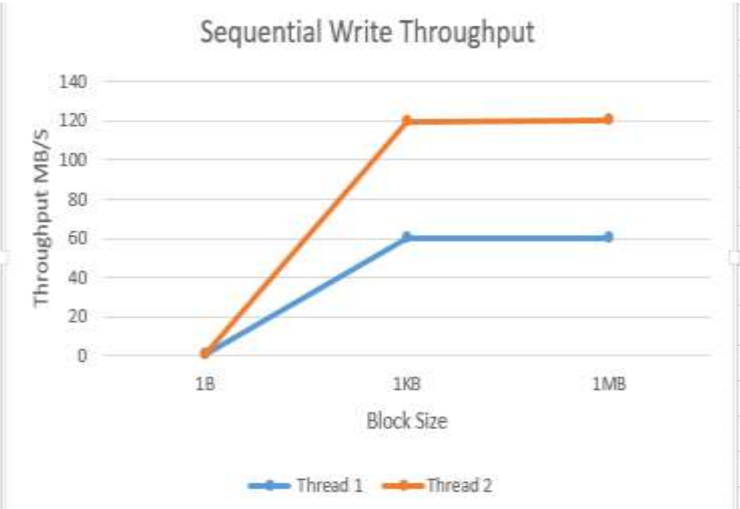
Sequential Write

The following results are obtained when sequential write operations are performed for varying block sizes of (1B,1KB,1MB) and varying concurrency of 1 and 2 threads.

- For 1Byte Block total data written is **50 MB**.
- For 1 KB and 1MB Block data written in **5 GB** to carry out the experiments precisely.

	Throughput(MB/Second)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 1MB
1	0.76 MB/s	60.19 MB/s	60.34 MB/s
2	1.00 MB/s	119.7 MB/s	120.33 MB/s

	Latency (ms)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 1MB
1	0.00124 ms	0.0162 ms	16.5 ms
2	0.00094 ms	0.0081 ms	8.3 ms



Evaluation :

- From the above graph for sequential write we observe that better performance for throughput was achieved using 2 threads.
- While using 1Byte buffer block the throughput achieved was very less as compared to the throughput achieved by performing the experiments using 1KB and 1MB block sizes.
- As the throughput increases the latency i.e. the time required to transfer each block size decreases.

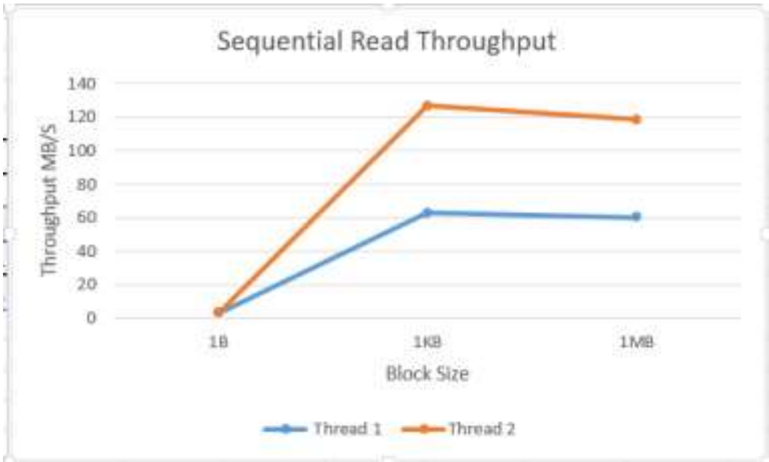
Sequential Read

The following results are obtained when sequential read operations are performed for varying block sizes of (1B,1KB,1MB) and varying concurrency of 1 and 2 threads.

- For 1B, 1KB and 1MB block sizes total data read is **1GB text file** to carry out the experiments precisely.

	Throughput(MB/Second)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 1MB
1	3.11 MB/s	63.00 MB/s	60.13/ MB/s
2	3.08 MB/s	127.05 MB/s	118.6 MB/s

	Latency (ms)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 1MB
1	0.0003 ms	0.0154 ms	16.6 ms
2	0.0003 ms	0.0076 ms	8.4 ms



Evaluation :

- From the above graph for sequential read we observe that better performance for throughput was achieved using 2 threads.
- While using 1Byte buffer block the throughput achieved was very less as compared to the throughput achieved by performing the experiments using 1KB and 1MB block sizes.

Parth Shah
A20352487

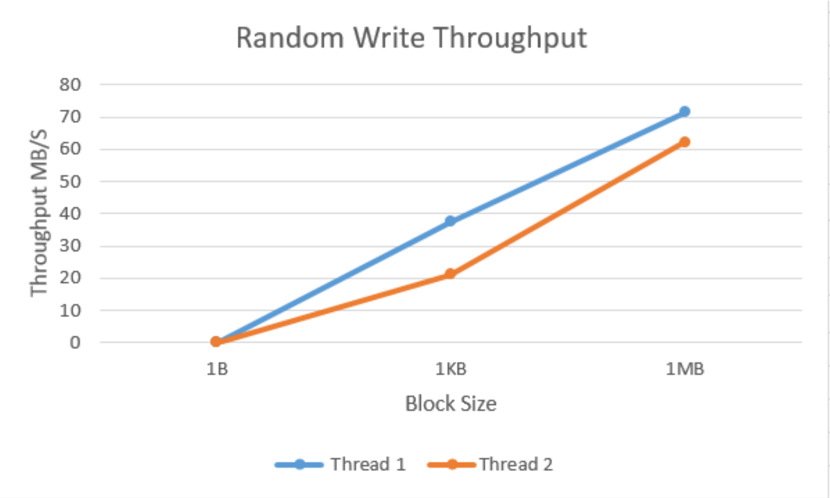
Random Write

The following results are obtained when Random write operations are performed for varying block sizes of (1B,1KB,1MB) and varying concurrency of 1 and 2 threads.

- For 1Byte Block total data written is **10 MB**.
- For 1 KB and 1MB Block data written in **1 GB** to carry out the experiments precisely.

	Throughput(MB/Second)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 1MB
1	0.0257 MB/s	37.56 MB/s	71.40 MB/s
2	0.0366 MB/s	21.22 MB/s	62.17 MB/s

	Latency (ms)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 1MB
1	0.037 ms	0.026 ms	14 ms
2	0.030 ms	0.046ms	16 ms



Evaluation :

- From the above graph for Random Write as the size of the buffer block increases there is constant increase in the throughput achieved.
- Moreover better performance was achieved using 2 threads as compared to 1 thread.

Parth Shah
A20352487

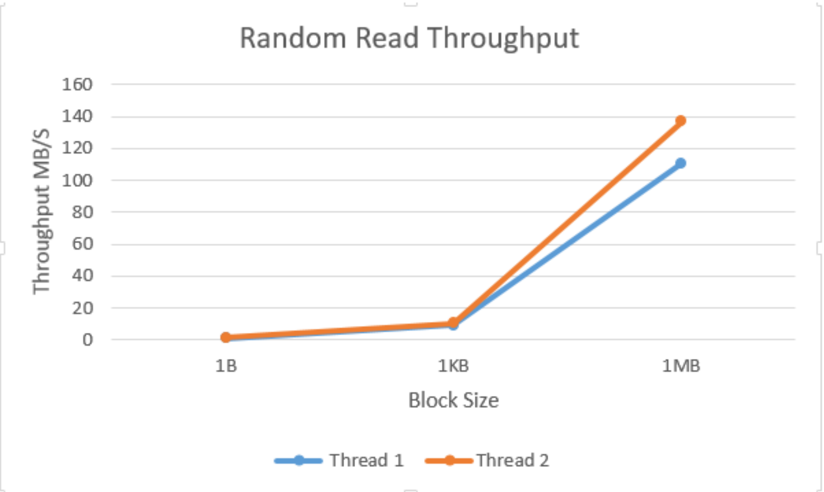
Random Read

The following results are obtained when Random read operations are performed for varying block sizes of (1B,1KB,1MB) and varying concurrency of 1 and 2 threads.

- For 1B block sizes total data read is **50MB .**
- For 1KB and 1MB block sizes total data read is **1GB text file** to carry out the experiments precisely.

	Throughput(MB/Second)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 1MB
1	1 MB/s	9.07 MB/s	110.75 MB/s
2	1.34 MB/s	10.32 MB/s	136.89 MB/s

	Latency (ms)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 1MB
1	0.0009 ms	0.026 ms	9.0 ms
2	0.0008 ms	0.046ms	7.3 ms

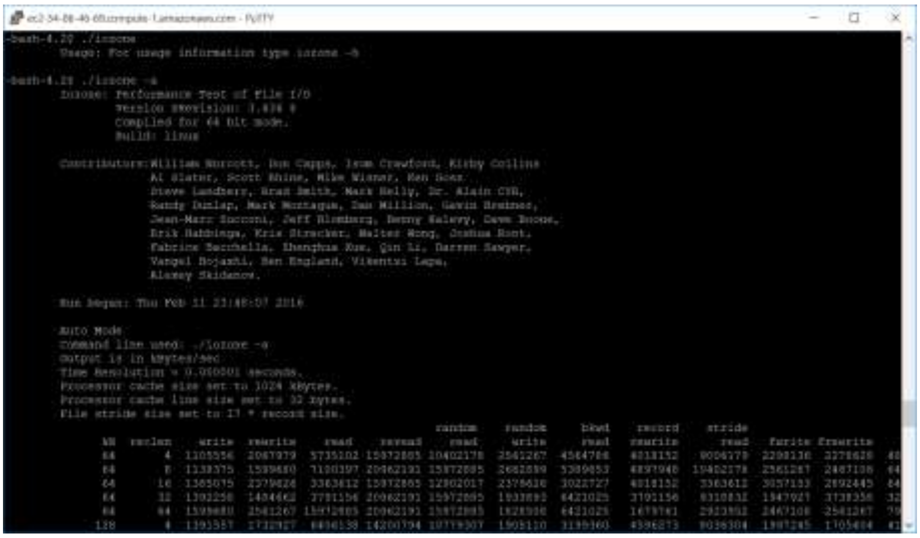


Evaluation :

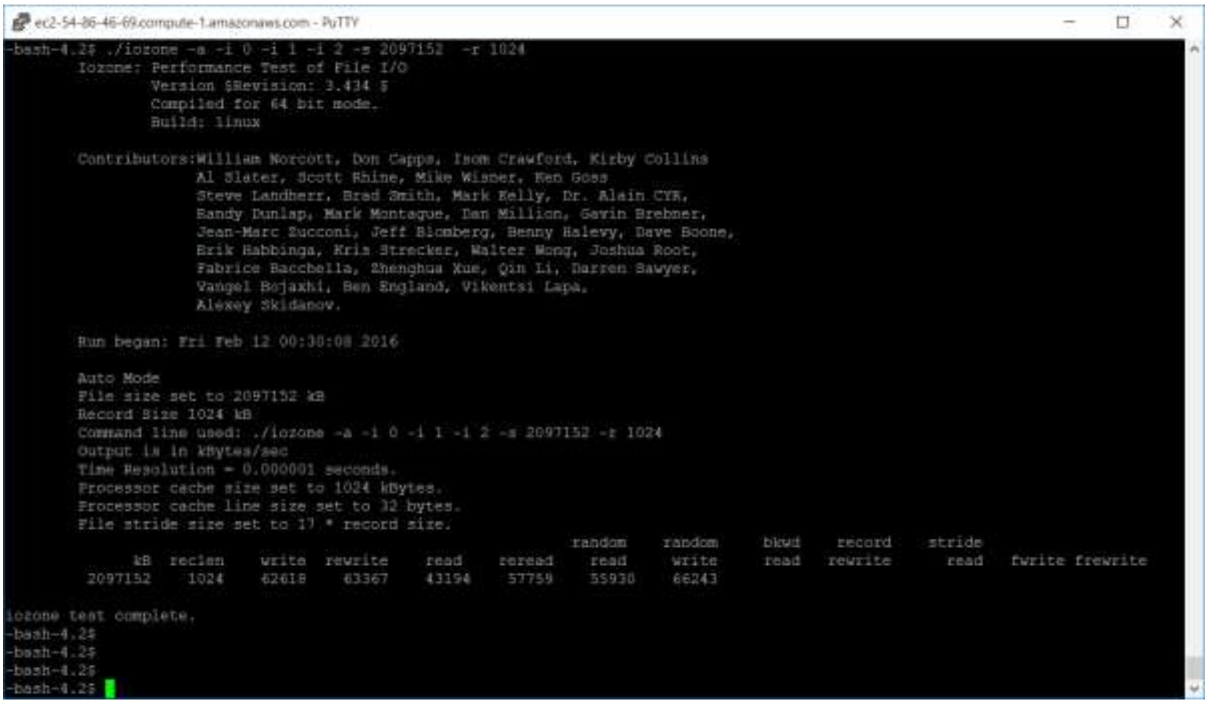
- While performing random read experiments with 1B and 1KB blocks we were not able to achieve high throughputs .
- While running the same experiment using 1MB block sizes higher values of throughput was achieved.
- The performance achieved using 1 thread and 2 threads was almost same.

Parth Shah
A20352487

- IOZONE Benchmark



- I have run the iozone benchmark to test the Disk Speed.



- Results obtained on iozone to test a **file of 2GB using block size of 1MB**.

Sequential Write MB/s	Sequential Read MB/s	Random write MB/s	Randon Read MB/s
63 MB/s	42.18 MB/s	64.69 MB/s	54.6 MB/s

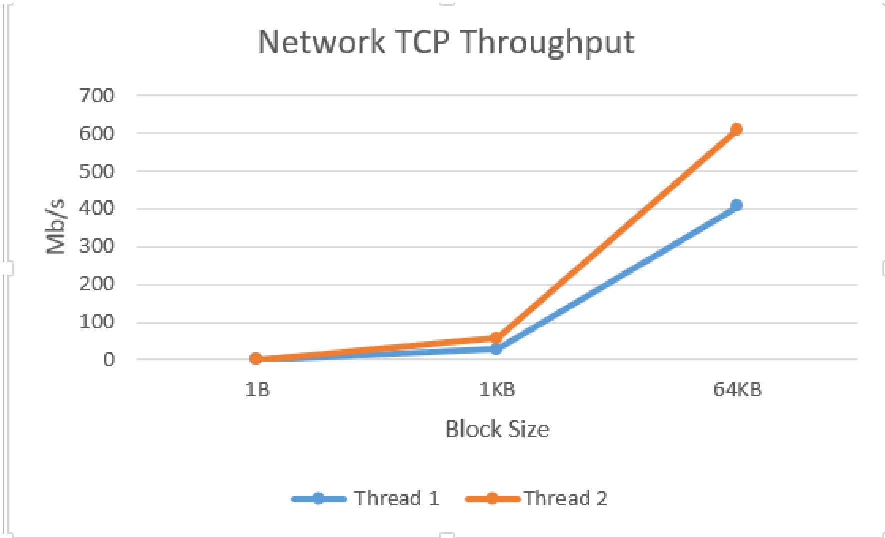
3) Network Experiments

For Network experiments , we need to measure the network speed between two instances using varying buffer size of 1B, 1KB and 64 KB with varying level of concurrency using 1 and 2 threads.

TCP

	Throughput (Mb/Second)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 64KB
1	0.01 Mb/s	28.67 Mb/s	407.77 Mb/s
2	0.017 Mb/s	57 Mb/s	611 Mb/s

	Latency (ms)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 64KB
1	0.008 ms	0.04 ms	2.4 ms
2	0.045 ms	0.031 ms	1.9 ms



Evaluation :

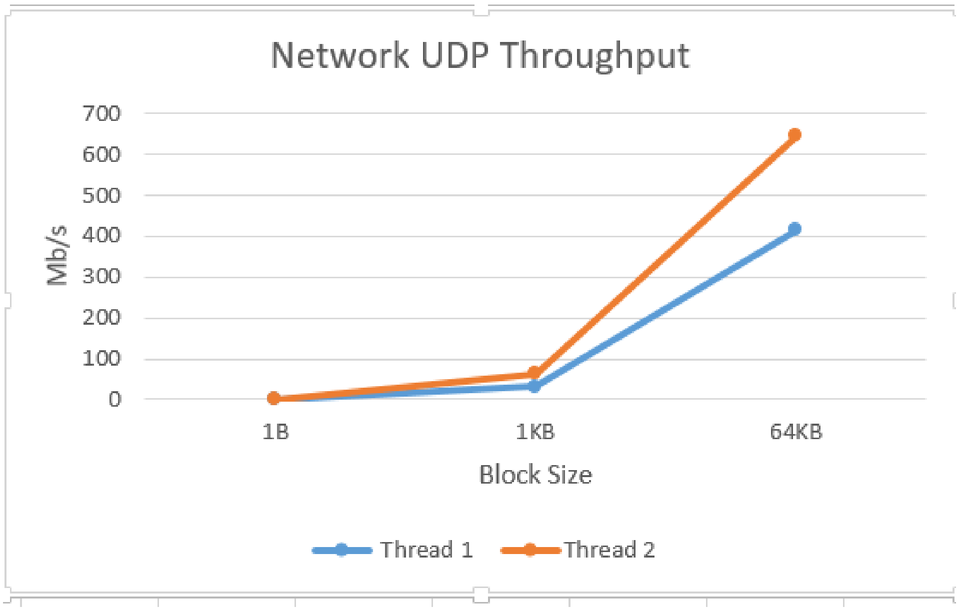
- From the above graph we observe that as the Block Size increases there is a steep growth in throughput achieved.
- For the block sizes of 1B and 1KB the throughput achieved was low as compared to the throughput achieved with block sizes of 64KB.
- The performance achieved with 2 threads was better than the performance achieved with single thread.

Parth Shah
A20352487

UDP

	Throughput (Mb/Second)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 64KB
1	0.02 Mb/s	31.84 Mb/s	414.91 Mb/s
2	0.03 Mb/s	63.67 Mb/s	645.37 Mb/s

	Latency (ms)		
No of Threads	Block Size 1B	Block Size 1KB	Block Size 64KB
1	0.006 ms	0.03 ms	2.1 ms
2	0.003 ms	0.025 ms	1.8 ms



Evaluation :

- The result obtained after running the experiments using UDP protocol was slighltly faster than TCP Protocol.
- As in TCP , in UDP too there was a steep growth in throughput achieved as the buffer block size increases.

Parth Shah
A20352487

- IPERF Benchmark

- Execute command
  ./iperf –s –p 12345

This command starts the server instance with IP 12345

ec2-54-86-46-69.compute-1.amazonaws.com - PuTTY

```
------------------------------------------------------------
Server listening on TCP port 12345
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 172.31.22.111 port 12345 connected with 54.172.40.85 port 46244
[  4]  0.0-30.1 sec  1.91 GBytes    546 Mbits/sec
^C-bash-4.2$ ./iperf -s -p 12345
------------------------------------------------------------
Server listening on TCP port 12345
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 172.31.22.111 port 12345 connected with 54.172.40.85 port 46245
[  4]  0.0-30.0 sec  1.97 GBytes    564 Mbits/sec
^C-bash-4.2$ ./iperf -s -p 12345
------------------------------------------------------------
Server listening on TCP port 12345
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 172.31.22.111 port 12345 connected with 54.172.40.85 port 46247
[  4]  0.0- 1.5 sec    100 MBytes    578 Mbits/sec
[  5] local 172.31.22.111 port 12345 connected with 54.172.40.85 port 46248
[  5]  0.0-16.0 sec  1000 MBytes    523 Mbits/sec
[  4] local 172.31.22.111 port 12345 connected with 54.172.40.85 port 46249
[  4]  0.0-30.0 sec  1.90 GBytes    544 Mbits/sec
```

Parth Shah
A20352487

References

1) http://www.binarytides.com/udp-socket-programming-in-java/

2) http://tutorials.jenkov.com/java-nio/index.html

3) https://www.cs.uic.edu/~troy/spring05/cs450/sockets/socket.html

4) http://www.iozone.org/docs/IOzone_msword_98.pdf

5)https://www.veritas.com/support/en_US/article.HOWTO64302