CS 553 CLOUD COMPUTING PROGRAMMING ASSIGNMENT 1

PERFORMANCE EVALUATION

Abhinav Pimpalgaonkar(A20387324)

Sandeep Vuzzini(A20243379)

This assignment measures the benchmark for different components of an instance on the Chameleon Cloud, the components that were evaluated for performance are listed below:

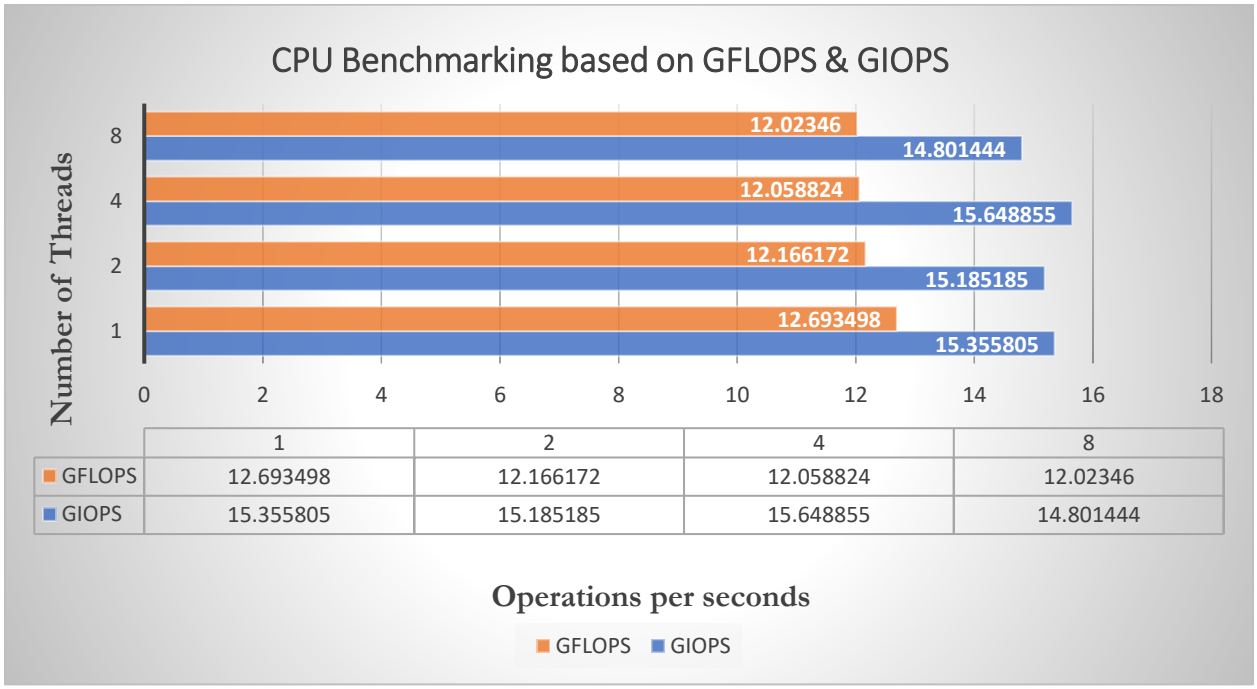
1. CPU
2. Memory
3. Disk
4. Network

I. CPU Benchmarking

For CPU Benchmarking, we used strong scaling, and measured the processor speed in terms of double precision floating point operations per second (Giga FLOPS, 10^9 FLOPS) and integer operations per second (Giga IOPS, 10^9 IOPS)

Also, we measured the processor speed at varying levels of concurrency (1 thread, 2 threads, 4 threads, 8 threads)

Please check the graph plot below for the output of our project for CPU performance.



	1	2	4	8
GFLOPS	12.693498	12.166172	12.058824	12.02346
GIOPS	15.355805	15.185185	15.648855	14.801444

Y-axis: Displays Number of Threads

X-axis: Displays Gigabit Operations per second (The operations can be either Integer/Floating). They are differed with different colors as shown in the table below the graph chart.

Theoretical Peak Performance: Number of cores*CPU Frequency*Threads*Instructions/Cycle

Conclusion: The performance of Integer operations per second is much better that performance of floating operations.

The optimal number of threads to get better performance is: **1**

Theoretical peak performance of the processor is: No. of. cores*cpu Frequency*threads*Ins/cycles

- ⇨ 1 *2.3 *1* 16
- ⇨ 41 GFLOPS

Efficiency obtained with respective to theoretical peak performance is: 48%

LINPACK Benchmark:

```
[root@pal-vuzzini-pimpalgaonkar linpack]# ./lininput_xeon64
./lininput_xeon64: line 1: syntax error near unexpected token `('
./lininput_xeon64: line 1: `Sample Intel(R) LINPACK data file (lininput_xeon64)'
[root@pal-vuzzini-pimpalgaonkar linpack]# ./xlinpack_xeon64
Input data or print help ? Type [data]/help :
data
Number of equations to solve (problem size): 5000
Leading dimension of array: 10000
Number of trials to run: 4
Data alignment value (in Kbytes): 64
Current date/time: Mon Oct  9 06:22:26 2017

CPU frequency:    2.825 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) : 5000
Leading dimension of array                  : 10000
Number of trials to run                     : 4
Data alignment value (in Kbytes)            : 64

Maximum memory requested that can be used = 400265536, at the size = 5000

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

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm)
5000   10000  64     2.233      37.3408  2.581643e-11 3.599893e-02
5000   10000  64     2.362      35.3093  2.581643e-11 3.599893e-02
5000   10000  64     2.408      34.6322  2.581643e-11 3.599893e-02
5000   10000  64     2.375      35.1157  2.581643e-11 3.599893e-02

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
5000   10000  64      35.5995  37.3408

End of tests
```

As seen from the LINPAC benchmark, the average GLOPS is 37.3408 for 2.3 seconds, this concludes to approximately 16.23 GFLOPS per second, which matches results of our code.

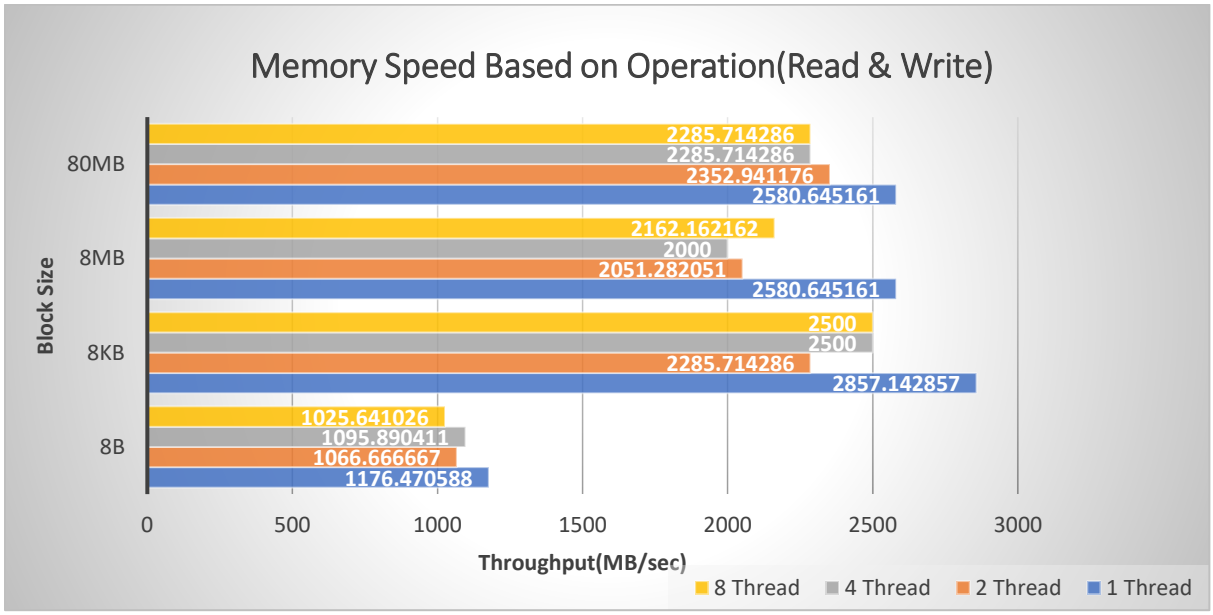
EXTRA CREDIT:

Our code is Implemented with AVX instruction and adequate evaluation was performed using bare metal provisioning.

II. Memory Benchmarking

For memory benchmarking, we used strong scaling and considered read+write operations (used memcpy), sequential write access (used memset), random write access with varying block sizes (8B, 8KB, 8MB, 80MB), and with varied concurrency (1 thread, 2 threads, 4 threads, and 8 threads)

Throughput for Read & Write Operation on the Memory:



No. of Threads	Varying block size	Operation	Throughput (MB/sec)
1	8B	Read & Write	1176.470588
1	8KB	Read & Write	2857.142857
1	8MB	Read & Write	2580.645161
1	80MB	Read & Write	2580.645161

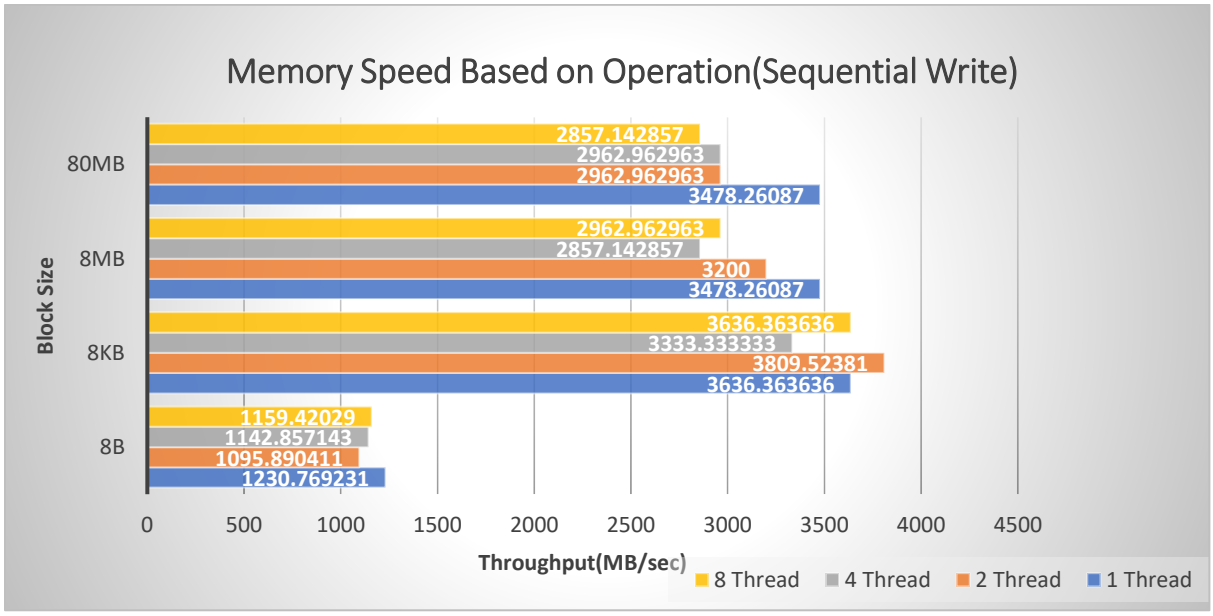
No. of Threads	Varying block size	Operation	Throughput (MB/sec)
2	8B	Read & Write	1066.666667
2	8KB	Read & Write	2285.714286
2	8MB	Read & Write	2051.282051
2	80MB	Read & Write	2352.941176

No. of Threads	Varying block size	Operation	Throughput (MB/sec)
4	8B	Read & Write	1095.890411
4	8KB	Read & Write	2500
4	8MB	Read & Write	2000
4	80MB	Read & Write	2285.714286

No. of Threads	Varying block size	Operation	Throughput (MB/sec)
8	8B	Read & Write	1025.641026
8	8KB	Read & Write	2500
8	8MB	Read & Write	2162.162162
8	80MB	Read & Write	2285.714286

Please refer to the table above for the detailed values collected for Read & Write operation across various threads.

Throughput for Sequential Write Operation on the Memory:



No. of Threads	Varying block size	Operation	Throughput (MB/sec)
1	8B	Sequential Write	1230.769231
1	8KB	Sequential Write	3636.363636
1	8MB	Sequential Write	3478.26087
1	80MB	Sequential Write	3478.26087

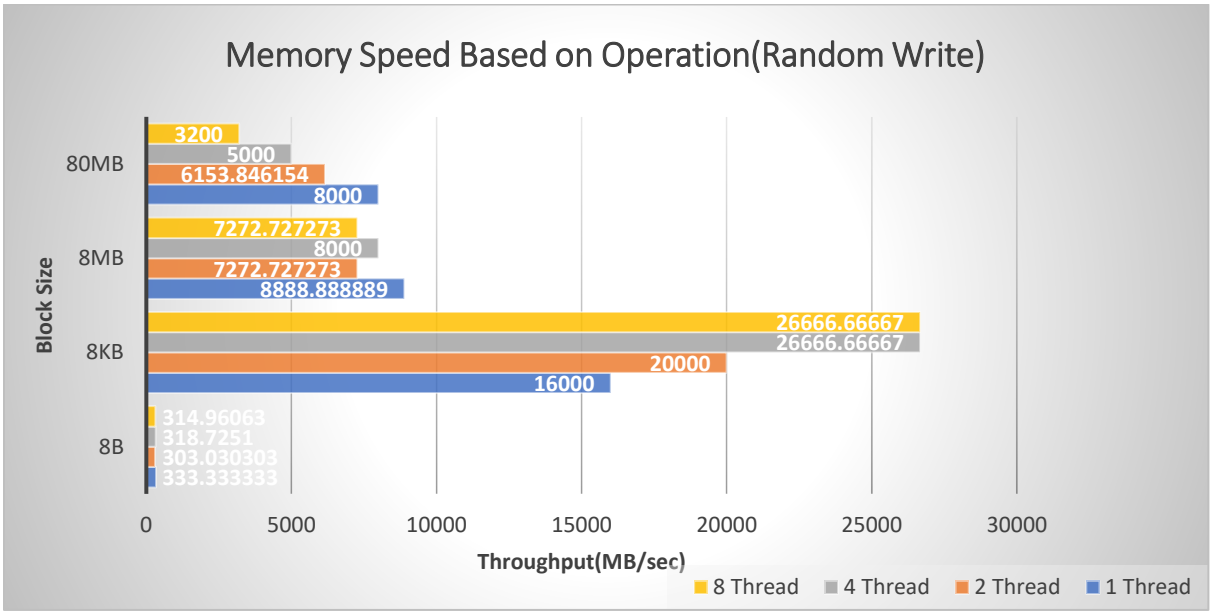
No. of Threads	Varying block size	Operation	Throughput (MB/sec)
2	8B	Sequential Write	1095.890411
2	8KB	Sequential Write	3809.52381
2	8MB	Sequential Write	3200
2	80MB	Sequential Write	2962.962963

No. of Threads	Varying block size	Operation	Throughput (MB/sec)
4	8B	Sequential Write	1142.857143
4	8KB	Sequential Write	3333.333333
4	8MB	Sequential Write	2857.142857
4	80MB	Sequential Write	2962.962963

No. of Threads	Varying block size	Operation	Throughput (MB/sec)
8	8B	Sequential Write	1159.42029
8	8KB	Sequential Write	3636.363636
8	8MB	Sequential Write	2962.962963
8	80MB	Sequential Write	2857.142857

Please refer to the table above for the detailed values collected for Sequential Write operation across various threads.

Throughput for Random Write Operation on the Memory:



No. of Threads	Varying block size	Operation	Throughput (MB/sec)
1	8B	Random Write	333.333333
1	8KB	Random Write	16000
1	8MB	Random Write	8888.888889
1	80MB	Random Write	8000

No. of Threads	Varying block size	Operation	Throughput (MB/sec)
2	8B	Random Write	303.030303
2	8KB	Random Write	20000
2	8MB	Random Write	7272.727273
2	80MB	Random Write	6153.846154

No. of Threads	Varying block size	Operation	Throughput (MB/sec)
4	8B	Random Write	318.7251
4	8KB	Random Write	26666.66667
4	8MB	Random Write	8000
4	80MB	Random Write	5000

No. of Threads	Varying block size	Operation	Throughput (MB/sec)

8	8B	Random Write	314.96063
8	8KB	Random Write	26666.66667
8	8MB	Random Write	7272.727273
8	80MB	Random Write	3200

Please refer to the table above for the detailed values collected for Random Write operation across various threads.

Theoretical peak performance of the memory is:

III. Disk Benchmarking

For Disk Benchmarking, we measured disk speed using parameter as given in the assignment: read+write operations, sequential read access, random read access, with varying block sizes (8B, 8KB, 8MB, 80MB), and varying the concurrency (1 thread, 2 threads, 4 threads, 8 threads)

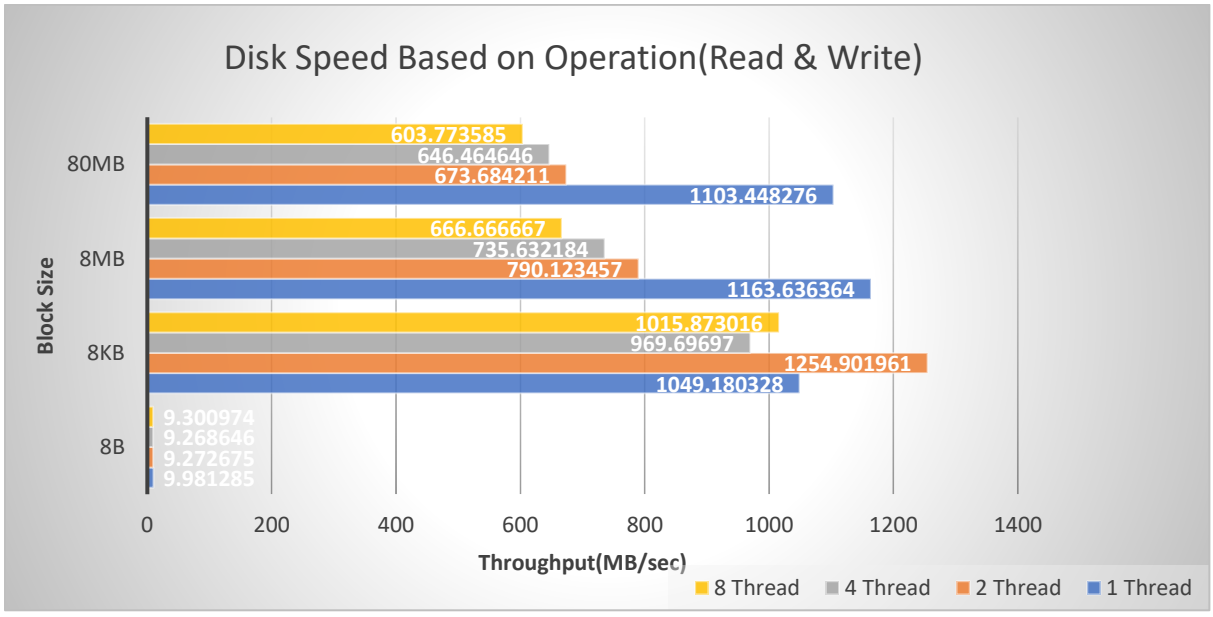
Specific to 8B block size, we measured latency, please check the table below for detailed values obtained for varying concurrency:

No. of Threads	Operation	Block Size	Latency for 8-byte Block Size in microsec
1	Read & Write	8 Byte	0.101775
1	Sequential Read	8 Byte	0.038728
1	Random Read	8 Byte	0.082642
2	Read & Write	8 Byte	0.103101
2	Sequential Read	8 Byte	0.040233
2	Random Read	8 Byte	0.085205
4	Read & Write	8 Byte	0.103593
4	Sequential Read	8 Byte	0.041604
4	Random Read	8 Byte	0.080839

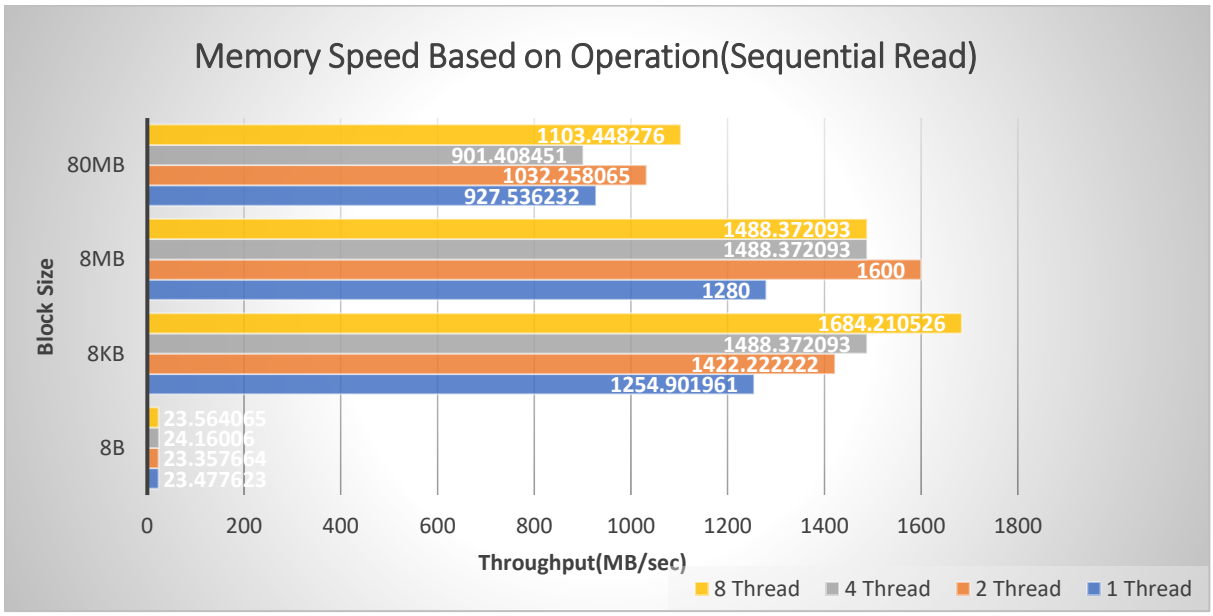
A sample screenshot for capturing latency statistics is shown below:

```
[root@pal-vuzzini-pimpalgaonkar cc]# ./Disk_Benchmarking
Perform Disk Benchmarking on
1. 1 Thread
2. 2 Thread
3. 4 Thread
4. 8 Thread
4
Number of Threads 8Choose Operation to be performed
1. Read and Write
2. Sequential Read
3. Random Read
3
Choose an option of  block size
1. 8 Byte
2. 8 Kilobyte
3. 8 Megabyte
4.80 Megabyte
1
Disk Benchmark Execution based on operation choosed,number of threads and block size

Creating a file for reading

Starting Benchmarking

Time Spent=54.250000
Throughput for read_disk_random = 11.797235 MB/s

Latency for block size of 8 byte =0.646710 microseconds

Latency for block size of 1 byte =0.080839 microseconds
```

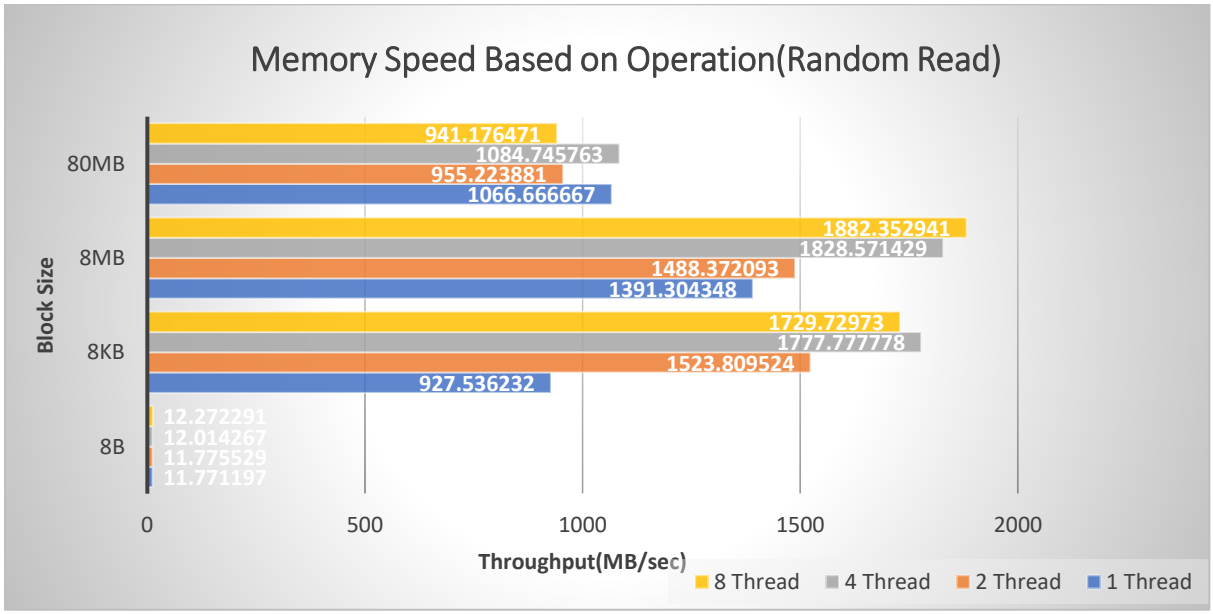
The below graph is shown for the Disk speed based on Read & Write Operation:



The below graph is shown for the Disk speed based on Sequential Read Operation:



The below graph is shown for the Disk speed based on Random Read Operation:



IOZONE Benchmarking:

1. Configuration of IOZONE

```
Run began: Mon Oct  9 02:27:23 2017

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.
```

2. I captured stats calculated by IOZONE for a sample of 8 MB disk file

KB	reclen	write	rewrite	read	reread	random read	random write	bkwd read	record rewrite	stride read	fwrite	frewrite	fread	freread
8192	4	834808	1794073	3407755	4063449	3483062	1865166	3084134	2578629	3497955	1942566	1694366	4257814	4331893
8192	8	932705	2093471	4164410	4667931	4638315	2260389	3213651	3672165	4493345	2251207	1861428	1306165	5392464

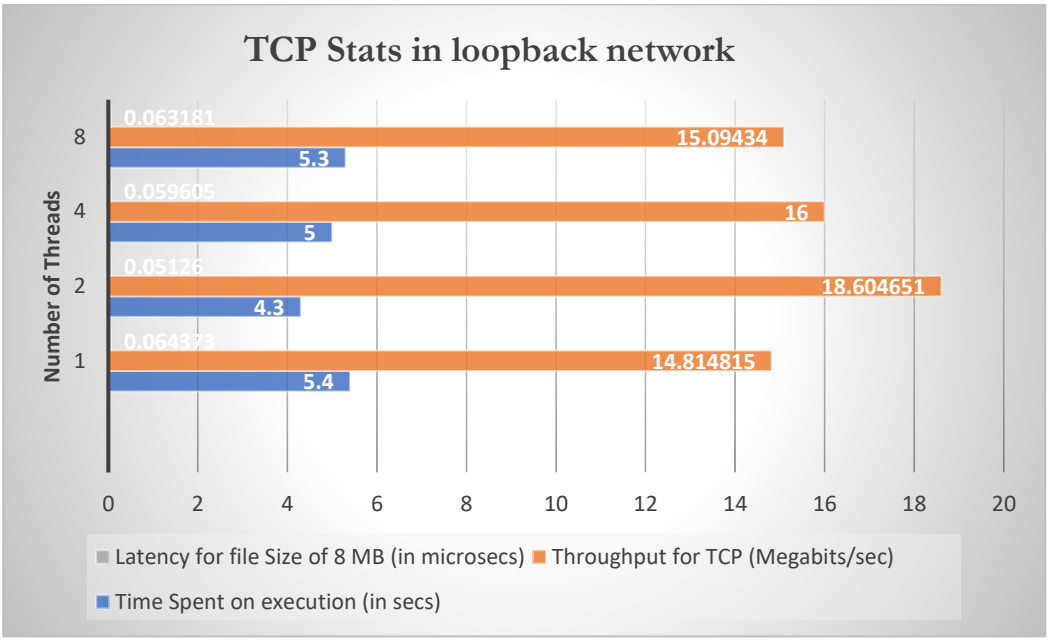
As output is in KBytes per second, for an 8 MB file size on the disk, and a chunk on 4 (which is eq. to number of threads in our experiment), the random read for an example is taking 3483062 KBytes/sec, which is equivalent to 3401 MBytes/sec throughput, this is more than my code output of 1888 MBytes/sec.

Throughput achieved by running my code when compared to IOZONE benchmark for Sequential read is approximately 64%, Random read is approximately 52%.

IV. Network Benchmarking

Graph Plot for TCP on Loopback network

Below is a graph plot for TCP statistics between two processes (server & client) on a loopback node.

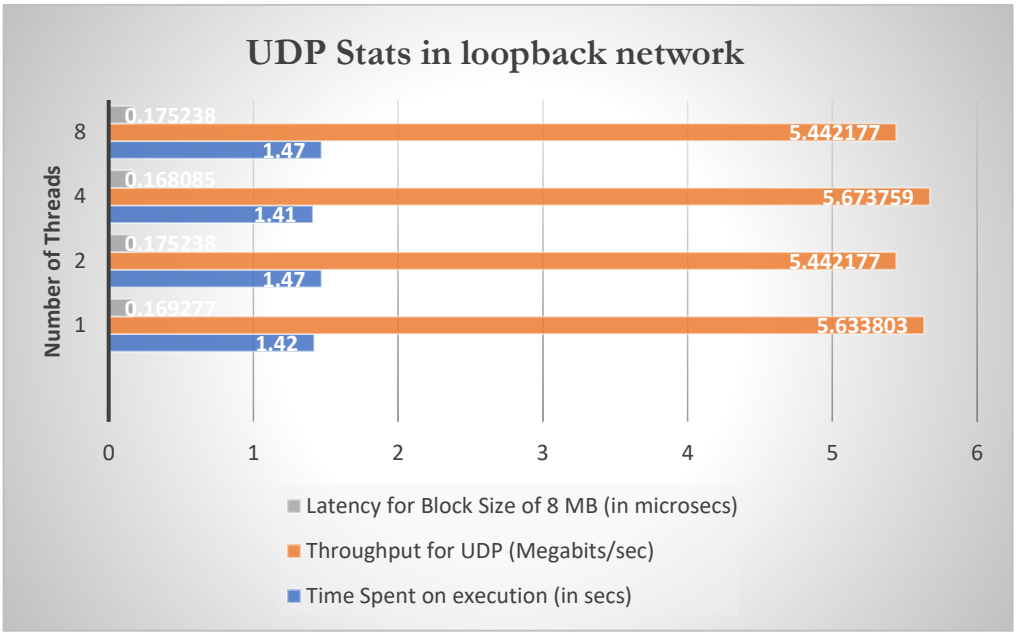


The collected stats for the above displayed graph are properly documented in the table below:

Number of Threads	Time Spent on execution (in secs)	Throughput for TCP (Megabits/sec)	Latency for file Size of 8 MB (in microsecs)
1	5.4	14.814815	0.064373
2	4.3	18.604651	0.05126
4	5	16	0.059605
8	5.3	15.09434	0.063181

Graph Plot for UDP on Loopback Network

Below is a graph plot for UDP statistics between two processes (server & client) on a loopback node.



The collected stats for the above displayed graph are properly documented in the table below:

Number of Threads	Time Spent on execution (in secs)	Throughput for UDP (Megabits/sec)	Latency for Block Size of 8 MB (in microsecs)
1	1.42	5.633803	0.169277
2	1.47	5.442177	0.175238
4	1.41	5.673759	0.168085
8	1.47	5.442177	0.175238

iPERF Statistics

This is the TCP statistics for iPERF

```
------------------------------------------------------------
Client connecting to 127.0.0.1, TCP port 5001
TCP window size: 2.50 MByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 40506 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  64.2 GBytes  55.1 Gbits/sec
```

As seen from the above output of iperf the bandwidth is 55 Gbits/sec, this appr. 80% faster than our throughput observed with code.

This is the UDP statistics for iPERF

```
Client connecting to 127.0.0.1, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 127.0.0.1 port 38635 connected with 127.0.0.1 port 5001
[ ID] Interval       Transfer     Bandwidth
[  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   0.005 ms    0/  893 (0%)
```

As seen from the above output of iperf the bandwidth is 1.05 Mbits/sec, this is approximately equal to the output observed in our code.

----------END of DOCUMENT----------