

---

**CS 553**  
**Cloud Computing**  
**Programming Assignment 1**  
**Performance Document**  
**Submitted By:**  
**Sumedha Gupta (A20377983)**  
**Aditya Jadhav (A20377887)**

---

**Note:**

- CPU, Memory, Disk and Network benchmarks are performed on Chameleon Open Stack KVM cloud m1.medium instance.

Flavor Details: m1.medium	
ID	3
VCPUs	2
RAM	4GB
Size	40GB

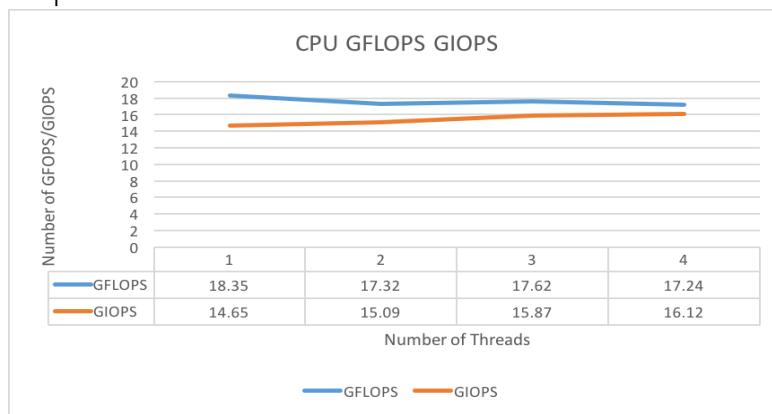
- GPU benchmarking is performed on Chameleon Bare Metal GPU M40 instance

Flavor Details: baremetal	
ID	baremetal
VCPUs	4
RAM	31.3GB
Size	100GB

- Contributions:
  - CPU, GPU and Network benchmarks are completed by Aditya Jadhav and Disk and Memory by Sumedha Gupta.

### CPU Benchmarking

- Strong Scaling is used to do all the experiments.
- In this section, the GFLOPS and GIOPS are calculated for different number of threads as shown below in the graphical representation:



- We have run the Linpack benchmark and various reading have been taken. For the comparison, average reading is considered.

```

x cc@pa1-sg:/tmp/l_mkl_p_2018.0.006/benchmarks_2018/linux/mkl/benchmarks/linpack
Current date/time: Sat Oct 7 16:37:03 2017

CPU frequency: 3.059 GHz
Number of CPUs: 2
Number of cores: 2
Number of threads: 2

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 1 1 1 1 1

Maximum memory requested that can be used=3202964416, at the size=20000

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

Size LDA Align. Time(s) GFlops Residual Residual(norm) Check
1000 1000 4 0.019 35.7829 9.394430e-13 3.203742e-02 pass
1000 1000 4 0.015 43.3941 9.394430e-13 3.203742e-02 pass
1000 1000 4 0.015 46.0414 9.394430e-13 3.203742e-02 pass
1000 1000 4 0.013 49.9148 9.394430e-13 3.203742e-02 pass
2000 2000 4 0.083 64.2622 4.085732e-12 3.554086e-02 pass
2000 2000 4 0.082 64.9343 4.085732e-12 3.554086e-02 pass
5000 5008 4 1.263 65.9969 2.262585e-11 3.154992e-02 pass
5000 5008 4 1.171 71.2238 2.262585e-11 3.154992e-02 pass
10000 10000 4 9.335 71.4385 9.187981e-11 3.239775e-02 pass
10000 10000 4 9.312 71.6111 9.187981e-11 3.239775e-02 pass
15000 15000 4 29.310 76.7816 2.219450e-10 3.495671e-02 pass
15000 15000 4 29.617 75.9849 2.219450e-10 3.495671e-02 pass
18000 18008 4 52.595 73.9360 2.886628e-10 3.161212e-02 pass
18000 18008 4 52.773 73.6859 2.886628e-10 3.161212e-02 pass
20000 20016 4 71.782 74.3101 3.669736e-10 3.248520e-02 pass
20000 20016 4 71.800 74.2912 3.669736e-10 3.248520e-02 pass

Performance Summary (GFlops)

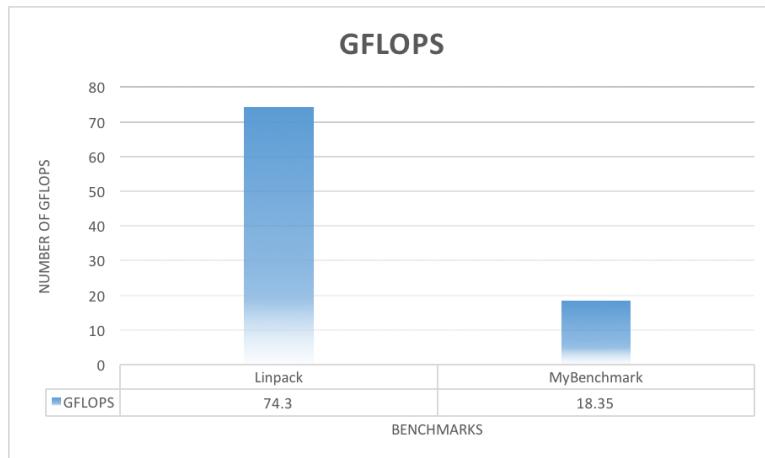
Size LDA Align. Average Maximal
1000 1000 4 43.7833 49.9148
2000 2000 4 64.5982 64.9343
5000 5008 4 68.6104 71.2238
10000 10000 4 71.5248 71.6111
15000 15000 4 76.3832 76.7816
18000 18008 4 73.8110 73.9360
20000 20016 4 74.3007 74.3101

Residual checks PASSED

End of tests

```

Below is the graphical representation of the comparison of our benchmark with Linpack:



Performance achieved by our benchmark is 18.35 GFLOPS and by linpack is 74.3 GFLOPS. It is less as linpack uses AVX instructions which provide better performance.

Performance achieved in comparison to linpack is **24.7%**.

- Theoretical performance and efficiency achieve compared to the theoretical performance?

As per the specifications of Chameleon Open Stack KVM cloud m1.medium instance.  
To check the specifications, the following command has been used:

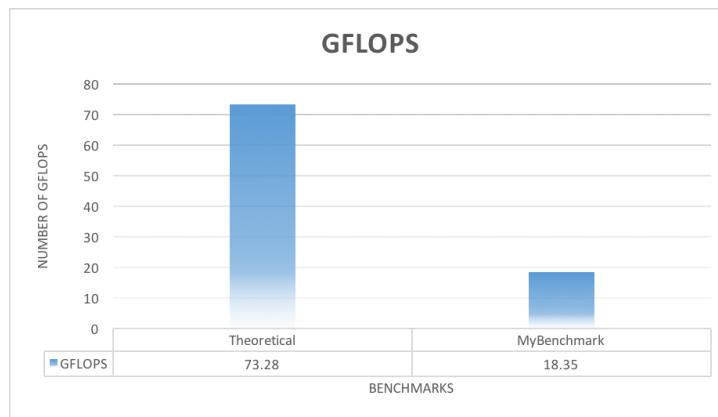
```
$ cat /proc/cpuinfo
```

IPC have been checked on the Intel website for Intel Xeon E312 (Sandy Bridge).

GFLOPS = (CPU speed in GHz) x (number of CPU cores) x (CPU instruction per cycle) x (number of CPUs per node)

$$\begin{aligned} &= (2 * 2.29 * 2 * 8) \\ &= 73.28 \end{aligned}$$

Comparison with our benchmark:

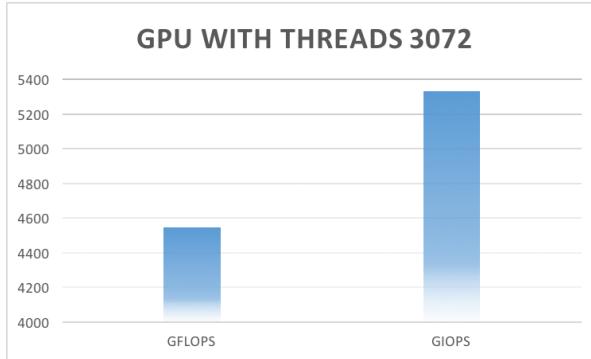


Performance achieved in comparison to theoretical performance is **25.04%**.

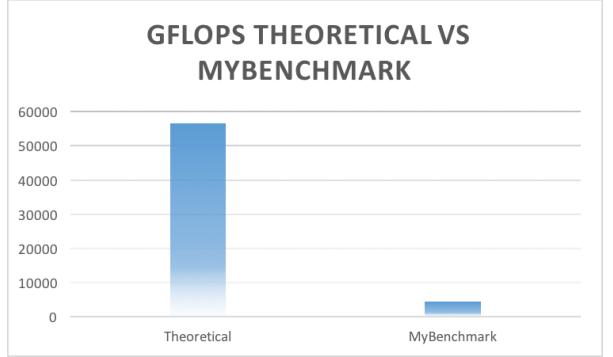
- Linpack and theoretical performance are almost same so we can conclude that their efficiency is same.

## GPU Benchmarking

- In this section, the GFLOPS and GIOPS are calculated on GPU M40 and results are shown below in the graphical representation:



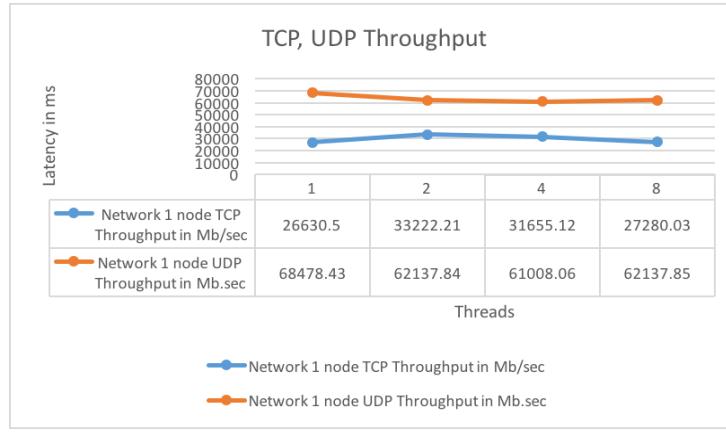
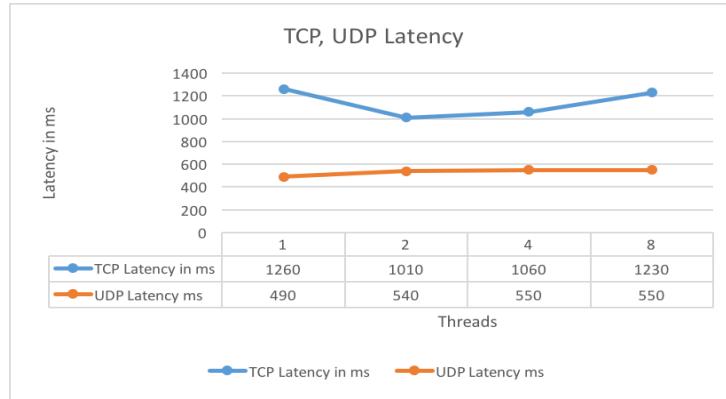
- Theoretical Performance:  $2.3 * 3072 * 8 = 56524.8$  GFLOPS



## Network Benchmarking

- The data transferred between client and server is done with packet size of 64 KB and 65536 times. So, the total data transferred is  $64 * 65536 * 2 = 8\text{GB}$
- This benchmark is implemented on 1 node as well as 2 nodes.
- Strong Scaling is used to do all the experiments.

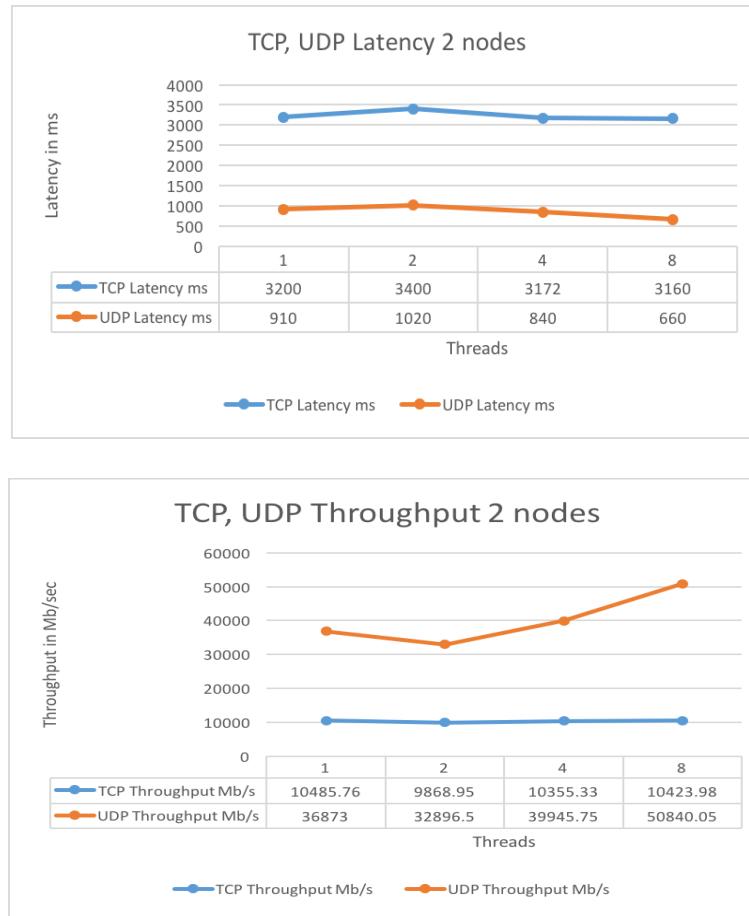
Below are the results for 1 node:



### Observations:

- In case of TCP, the fluctuations in the latency are significant as the number of threads increases while the throughput doesn't show much significant change.
- In case of UDP, there is not a significant change in both throughput and latency as the number of thread increases.
- If we compare, TCP and UDP latency and throughput, UDP has much better performance than TCP.

**Extra Credit:** Below are the results for 2 nodes:



#### Observations:

- In case of UDP, the fluctuations are significant as the number of threads increases. This may be because of the loss of data packets.
- In case of TCP, there is not a significant change in both throughput and latency as the number of thread increases.
- If we compare, TCP and UDP latency and throughput, UDP has much better performance than TCP.

#### Iperf benchmark Ouput:

- This benchmark has been tested on the Chameleon medium instance and below are the instructions to run the benchmark:

##### Server:

```
Iperf3 -s
```

##### Client TCP:

```
iperf3 -c <IP_ADDRESS> -w 64k -f Mb
```

where, -w -> window size

-f -> format to report: Kbits, Mbits, KBytes, MBytes

-c -> server ip

The output for TCP 1 node on IPerf:

```
cc@aditya-node-1:~
```

Server listening on 5201

Accepted connection from 127.0.0.1, port 56584

[ ID]	Interval	Transfer	Bandwidth
[ 5]	0.00-1.00	sec 0.20 GBytes	18901 Mbytes/sec
[ 5]	1.00-2.00	sec 0.44 GBytes	20921 Mbytes/sec
[ 5]	2.00-3.00	sec 0.19 GBytes	18812 Mbytes/sec
[ 5]	3.00-4.00	sec 0.26 GBytes	19402 Mbytes/sec
[ 5]	4.00-5.00	sec 0.60 GBytes	22333 Mbytes/sec
[ 5]	5.00-6.00	sec 0.06 GBytes	17705 Mbytes/sec
[ 5]	6.00-7.00	sec 0.42 GBytes	20781 Mbytes/sec
[ 5]	7.00-8.00	sec 0.10 GBytes	18036 Mbytes/sec
[ 5]	8.00-9.00	sec 0.65 GBytes	22730 Mbytes/sec
[ 5]	9.00-10.00	sec 0.36 GBytes	20260 Mbytes/sec
[ 5]	10.00-10.04	sec 112 MBytes	26003 Mbytes/sec

-----

[ ID]	Interval	Transfer	Bandwidth	sender	receiver
[ 5]	0.00-10.04	sec 0.00 Bytes	0.00 Mbytes/sec		
[ 5]	0.00-10.04	sec 23.4 GBytes	20011 Mbytes/sec		

Server listening on 5201

```
cc@aditya-node-1:~
```

[cc@aditya-node-1 ~]# iperf3 -c 127.0.0.1 -f m -w 64k

Connecting to host 127.0.0.1, port 5201

[ ID]	Interval	Transfer	Bandwidth	Retr	Cwnd
[ 4]	0.00-1.00	sec 2.31 GBytes	19813 Mbytes/sec	0	320 KBytes
[ 4]	1.00-2.00	sec 2.40 GBytes	20588 Mbytes/sec	0	320 KBytes
[ 4]	2.00-3.00	sec 2.22 GBytes	19088 Mbytes/sec	0	320 KBytes
[ 4]	3.00-4.00	sec 2.27 GBytes	19477 Mbytes/sec	0	320 KBytes
[ 4]	4.00-5.00	sec 2.60 GBytes	22354 Mbytes/sec	0	320 KBytes
[ 4]	5.00-6.00	sec 2.05 GBytes	17637 Mbytes/sec	0	320 KBytes
[ 4]	6.00-7.00	sec 2.40 GBytes	20639 Mbytes/sec	0	320 KBytes
[ 4]	7.00-8.00	sec 2.11 GBytes	18160 Mbytes/sec	0	320 KBytes
[ 4]	8.00-9.00	sec 2.65 GBytes	22794 Mbytes/sec	0	320 KBytes
[ 4]	9.00-10.00	sec 2.36 GBytes	20278 Mbytes/sec	0	320 KBytes

-----

[ ID]	Interval	Transfer	Bandwidth	Retr	sender	receiver
[ 4]	0.00-10.00	sec 23.4 GBytes	20083 Mbytes/sec	0		
[ 4]	0.00-10.00	sec 23.4 GBytes	20083 Mbytes/sec	0		

iperf Done.

```
[cc@aditya-node-1 ~]$
```

## Client UDP:

```
iperf3 -u -c <IP_ADDRESS> -w 64k -f Mb -b
```

where,-u -> UDP

-w -> window size

-f -> format to report: Kbits, Mbits, KBytes, MBytes

-c -> server ip

**-b** -> 0, will disable bandwidth limits (particularly useful for UDP tests)

The output for UDP 1 node on IPerf:

```
cc@aditya-node-1:~
```

Server listening on 5201

Accepted connection from 127.0.0.1, port 56546

```
[ 5] local 127.0.0.1 port 5201 connected to 127.0.0.1 port 38984
```

ID	Interval	Transfer	Bandwidth	Jitter	Lost/Total Datagrams
[ 5]	0.00-1.00	sec 3.86 GBytes	33093 Mbites/sec	0.345 ms	22207/[11719 (10%)
[ 5]	1.00-2.00	sec 1.74 GBytes	14956 Mbites/sec	0.002 ms	193732/[279244 (69%)
[ 5]	2.00-3.00	sec 3.85 GBytes	33064 Mbites/sec	0.001 ms	43289/[31585 (18%)
[ 5]	3.00-4.00	sec 3.39 GBytes	29118 Mbites/sec	0.066 ms	76071/[42650 (31%)
[ 5]	4.00-5.00	sec 3.72 GBytes	31961 Mbites/sec	0.119 ms	50838/[33864 (22%)
[ 5]	5.00-6.00	sec 2.86 GBytes	24570 Mbites/sec	0.002 ms	106909/[247393 (43%)
[ 5]	6.00-7.00	sec 4.13 GBytes	35494 Mbites/sec	0.002 ms	18429/[221533 (8.3%)
[ 5]	7.00-8.00	sec 1.97 GBytes	16938 Mbites/sec	0.056 ms	129543/[226541 (57%)
[ 5]	8.00-9.00	sec 4.07 GBytes	34984 Mbites/sec	0.001 ms	42234/[42325 (17%)
[ 5]	9.00-10.00	sec 3.40 GBytes	29160 Mbites/sec	0.103 ms	81923/[248910 (33%)
[ 5]	10.00-10.04	sec 79.4 MBBytes	17471 Mbites/sec	0.001 ms	7548/[11359 (66%)

```
[ ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
```

ID	Interval	Transfer	Bandwidth	Jitter	Lost/Total Datagrams
[ 5]	0.00-10.04	sec 0.00 Bytes	0.00 Mbites/sec	0.001 ms	771823/[2397090 (32%)

Server listening on 5201

```
cc@aditya-node-1:~
```

```
[cc@aditya-node-1 ~]$ iperf3 -c 127.0.0.1 -u -f m -w 64K -b 0
```

Connecting to host 127.0.0.1, port 5201

```
[ 4] local 127.0.0.1 port 38984 connected to 127.0.0.1 port 5201
```

ID	Interval	Transfer	Bandwidth	Total Datagrams
[ 4]	0.00-1.00	sec 4.57 GBytes	39287 Mbites/sec	224810
[ 4]	1.00-2.00	sec 5.65 GBytes	48555 Mbites/sec	277840
[ 4]	2.00-3.00	sec 4.69 GBytes	40291 Mbites/sec	230550
[ 4]	3.00-4.00	sec 4.94 GBytes	42420 Mbites/sec	242740
[ 4]	4.00-5.00	sec 4.80 GBytes	41207 Mbites/sec	235790
[ 4]	5.00-6.00	sec 4.95 GBytes	42501 Mbites/sec	243200
[ 4]	6.00-7.00	sec 4.53 GBBytes	38953 Mbites/sec	223890
[ 4]	7.00-8.00	sec 4.63 GBBytes	39740 Mbites/sec	227400
[ 4]	8.00-9.00	sec 4.92 GBBytes	42284 Mbites/sec	241960
[ 4]	9.00-10.00	sec 5.08 GBBytes	43675 Mbites/sec	249910

```
[ ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
```

ID	Interval	Transfer	Bandwidth	Jitter	Lost/Total Datagrams
[ 4]	0.00-10.00	sec 48.6 GBytes	41891 Mbites/sec	0.001 ms	771823/[2397090 (32%)
[ 4]	Sent 2397090 datagrams				

iperf Done.

```
[cc@aditya-node-1 ~]$
```

- What efficiency do you achieve compared to the theoretical network performance?

Theoretical Data Rate is 10000 MB/sec i.e. 10GB/sec and our is 2GB/sec. So, 20% efficiency

- Repeat the benchmark across two nodes; what efficiency do you achieve compared to the theoretical network speed?

Iperf output is approx.

The output for TCP 2 nodes on IPerf: 1091 MB/sec and is 10% of theoretical performance.

```

cc@pal-gw:~$ Server listening on 5201
-----
Accepted connection from 192.168.0.4, port 59980
[ 5] local 192.168.0.214 port 5201 connected to 192.168.0.4 port 59980
[ ID] Interval Transfer Bandwidth
[ 5] 0.00-1.00 sec 95.7 MBytes 803 Mbits/sec
[ 5] 1.00-2.00 sec 131 MBytes 1102 Mbits/sec
[ 5] 2.00-3.00 sec 117 MBytes 982 Mbits/sec
[ 5] 3.00-4.00 sec 111 MBytes 928 Mbits/sec
[ 5] 4.00-5.00 sec 130 MBytes 1093 Mbits/sec
[ 5] 5.00-6.00 sec 146 MBytes 1225 Mbits/sec
[ 5] 6.00-7.00 sec 112 MBytes 939 Mbits/sec
[ 5] 7.00-8.00 sec 127 MBytes 1067 Mbits/sec
[ 5] 8.00-9.00 sec 131 MBytes 1098 Mbits/sec
[ 5] 9.00-10.00 sec 130 MBytes 1090 Mbits/sec
[ 5] 10.00-10.04 sec 4.37 MBytes 346 Mbits/sec
----- sender
[ ID] Interval Transfer Bandwidth
[ 5] 0.00-10.04 sec 0.00 Bytes 0.00 Mbits/sec
[ 5] 0.00-10.04 sec 1.21 GBytes 1033 Mbits/sec
----- receiver
iperf Done.
[cc@aditya-node-1 ~]$ 

```

Server listening on 5201

The output for UDP 2 nodes on IPerf:

```

cc@pal-gw:~$ Server listening on 5201
-----
Accepted connection from 192.168.0.4, port 59996
[ 5] local 192.168.0.214 port 5201 connected to 192.168.0.4 port 57798
[ ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[ 5] 0.00-1.00 sec 218 MBytes 1827 Mbits/sec 0.005 ms 10943/168698 (6.5%)
[ 5] 1.00-2.00 sec 220 MBytes 1842 Mbits/sec 0.004 ms 1151/180125 (1.2%)
[ 5] 2.00-3.00 sec 196 MBytes 1646 Mbits/sec 0.006 ms 40695/182771 (2.2%)
[ 5] 3.00-4.00 sec 208 MBytes 1749 Mbits/sec 0.003 ms 30556/181504 (1.7%)
[ 5] 4.00-5.00 sec 234 MBytes 1959 Mbits/sec 0.009 ms 11112/180201 (6.1%)
[ 5] 5.00-6.00 sec 239 MBytes 2007 Mbits/sec 0.004 ms 12560/185794 (6.8%)
[ 5] 6.00-7.00 sec 247 MBytes 2073 Mbits/sec 0.025 ms 4695/183700 (2.6%)
[ 5] 7.00-8.00 sec 258 MBytes 2162 Mbits/sec 0.004 ms 1853/189464 (0.98%)
[ 5] 8.00-9.00 sec 252 MBytes 2115 Mbits/sec 0.002 ms 10039/192609 (5.2%)
[ 5] 9.00-10.00 sec 223 MBytes 1873 Mbits/sec 0.009 ms 31602/193287 (1.6%)
[ 5] 10.00-10.04 sec 10.0 MBytes 2089 Mbits/sec 0.004 ms 647/7889 (0.2%)
----- sender
[ ID] Interval Transfer Bandwidth Jitter Lost/Total Datagrams
[ 5] 0.00-10.04 sec 0.00 Bytes 0.00 Mbits/sec 0.004 ms 175853/1845044 (9.5%)
----- receiver
iperf Done.
[cc@aditya-node-1 ~]$ 

```

Server listening on 5201

- For the latency, use the ping utility to measure latency across the loopback, and across the network.
- How does it compare to the theoretical latency?

Latency has been measured using ping utility, below is the output on both localhost and network.

```

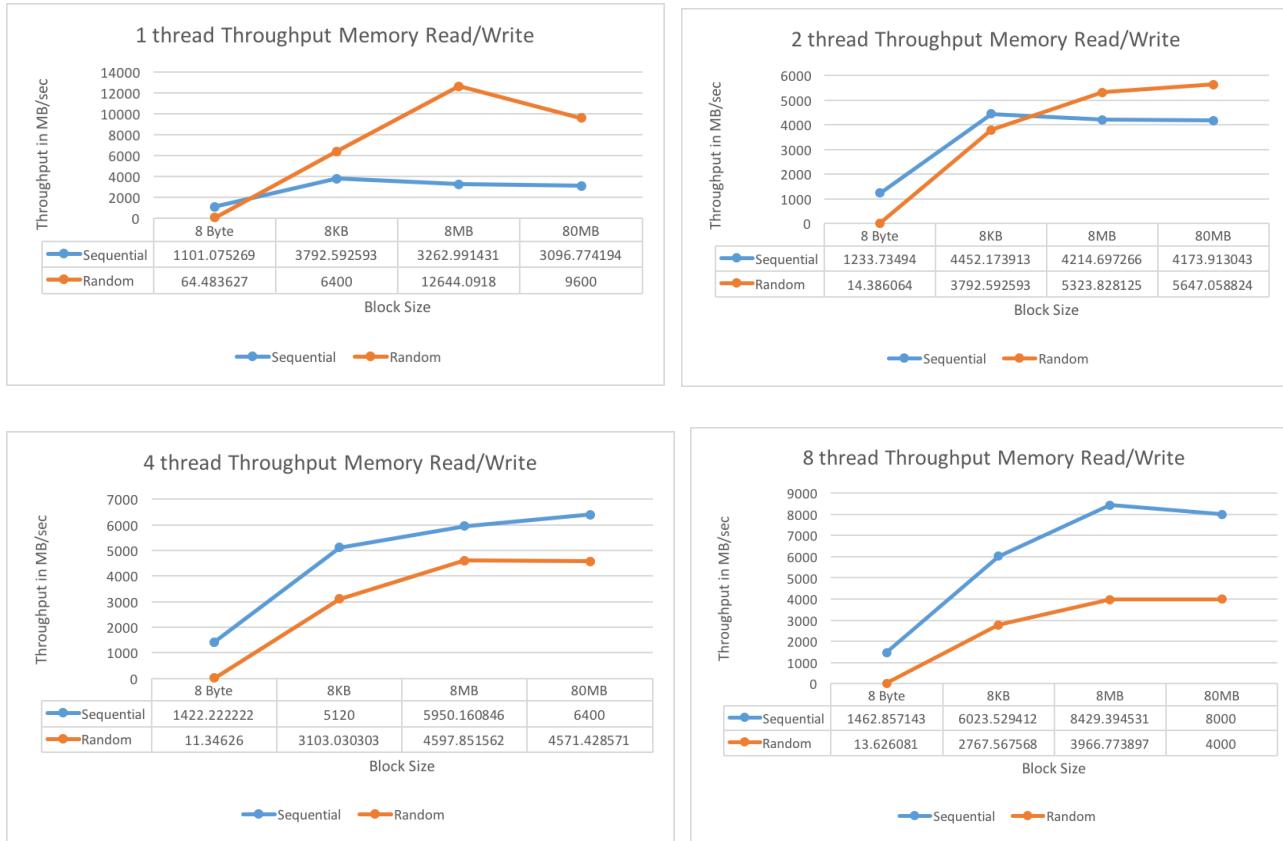
cc@aditya-node-2:~$ ping 192.168.0.150 -s 64000 -c 15
PING 192.168.0.150 (192.168.0.150) 64000(64028) bytes of data.
64008 bytes from 192.168.0.150: icmp_seq=1 ttl=64 time=3.38 ms
64008 bytes from 192.168.0.150: icmp_seq=2 ttl=64 time=1.80 ms
64008 bytes from 192.168.0.150: icmp_seq=3 ttl=64 time=1.70 ms
64008 bytes from 192.168.0.150: icmp_seq=4 ttl=64 time=1.79 ms
64008 bytes from 192.168.0.150: icmp_seq=5 ttl=64 time=1.75 ms
64008 bytes from 192.168.0.150: icmp_seq=6 ttl=64 time=1.66 ms
64008 bytes from 192.168.0.150: icmp_seq=7 ttl=64 time=1.65 ms
64008 bytes from 192.168.0.150: icmp_seq=8 ttl=64 time=1.75 ms
64008 bytes from 192.168.0.150: icmp_seq=9 ttl=64 time=1.71 ms
64008 bytes from 192.168.0.150: icmp_seq=10 ttl=64 time=1.65 ms
64008 bytes from 192.168.0.150: icmp_seq=11 ttl=64 time=1.73 ms
64008 bytes from 192.168.0.150: icmp_seq=12 ttl=64 time=1.70 ms
64008 bytes from 192.168.0.150: icmp_seq=13 ttl=64 time=1.70 ms
64008 bytes from 192.168.0.150: icmp_seq=14 ttl=64 time=1.59 ms
64008 bytes from 192.168.0.150: icmp_seq=15 ttl=64 time=1.55 ms
--- 192.168.0.150 ping statistics ---
15 packets transmitted, 15 received, 0% packet loss, time 14022ms
rtt min/avg/max/mdev = 1.553/1.811/3.382/0.428 ms
[cc@aditya-node-2 ~]$ 

```

No. of data transferred is 64008 bytes \* 15 packets, the average latency is captures as **0.068 ms on localhost and 1.811 ms on network**.

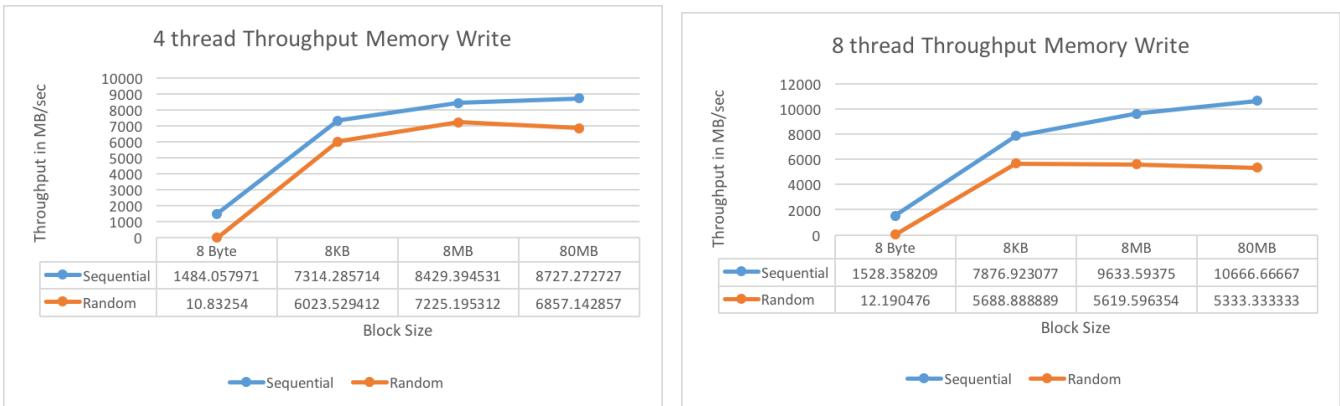
## Memory Benchmarking

- Strong Scaling is used to do all the experiments.
- A large piece of memory has been allocated to perform the benchmark. The block sizes of 8B, 8KB, 8MB, 80MB with threads varying as 1, 2, 4 and 8 have been taken.
- Memory read/write sequentially, randomly and write sequentially and randomly functions have been implemented.
- Below are the results for various operations with varying level of concurrency:



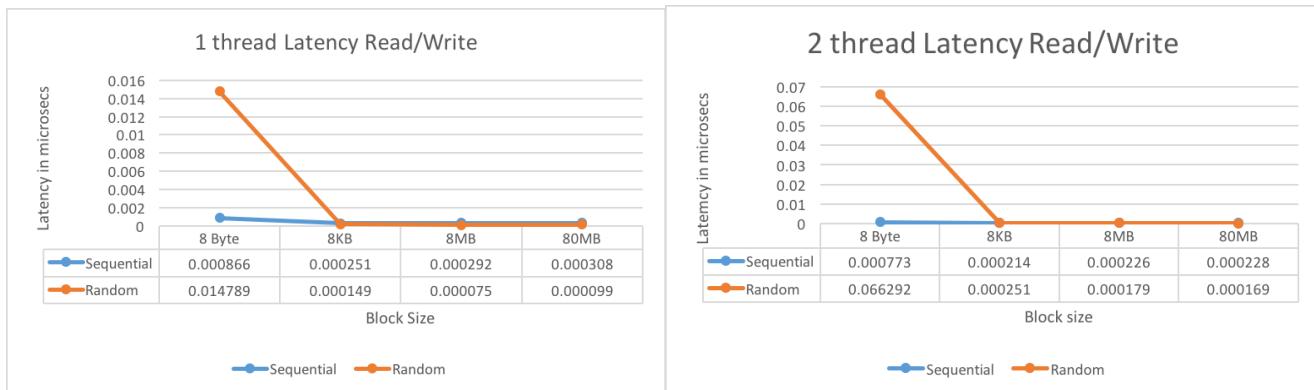
### Observations:

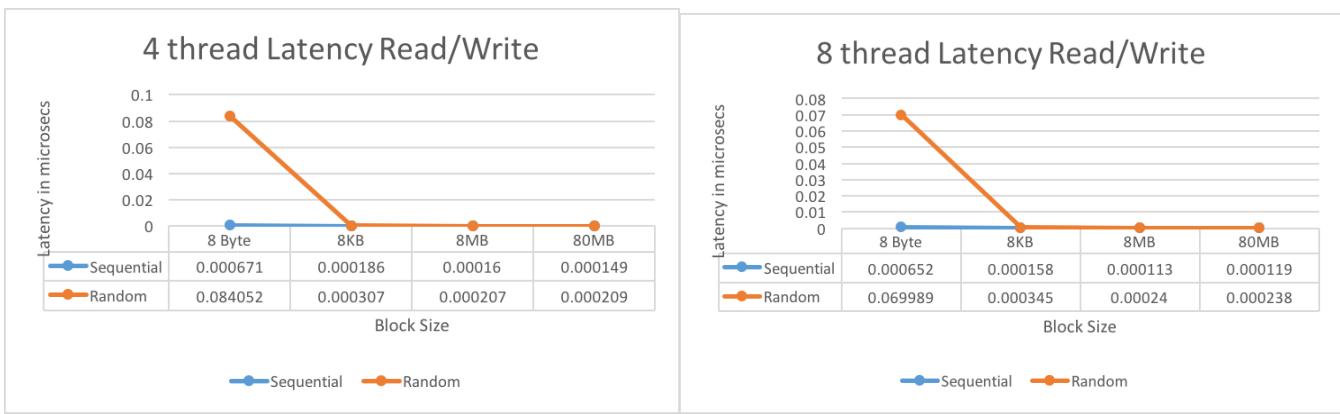
- The throughput for sequential read and write operations increases by a significant amount as the number of thread increases and as block size varies.
- The performance of random operation is much less than the sequential except for one thread the performance by random operations is higher than sequential ones.
- **Optimal Concurrency:**
  - 8 threads with 8 MB block size



### Observations:

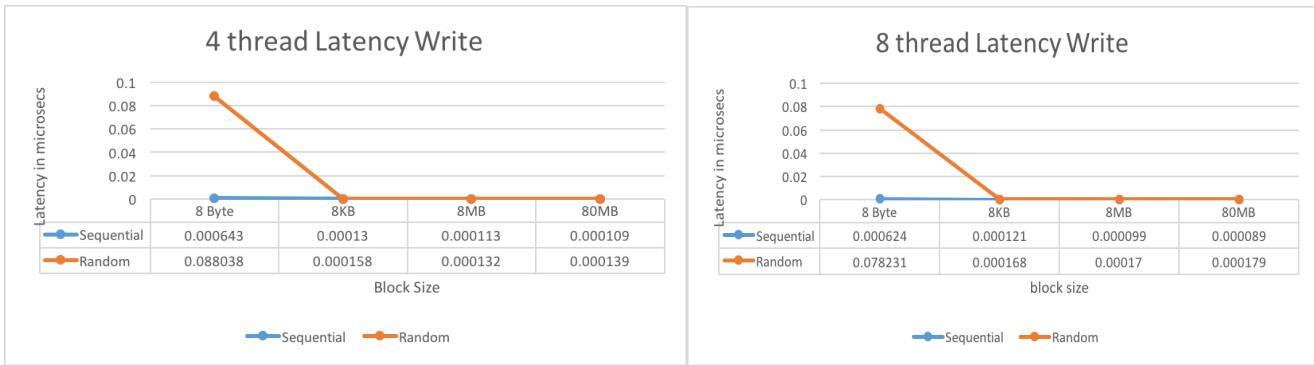
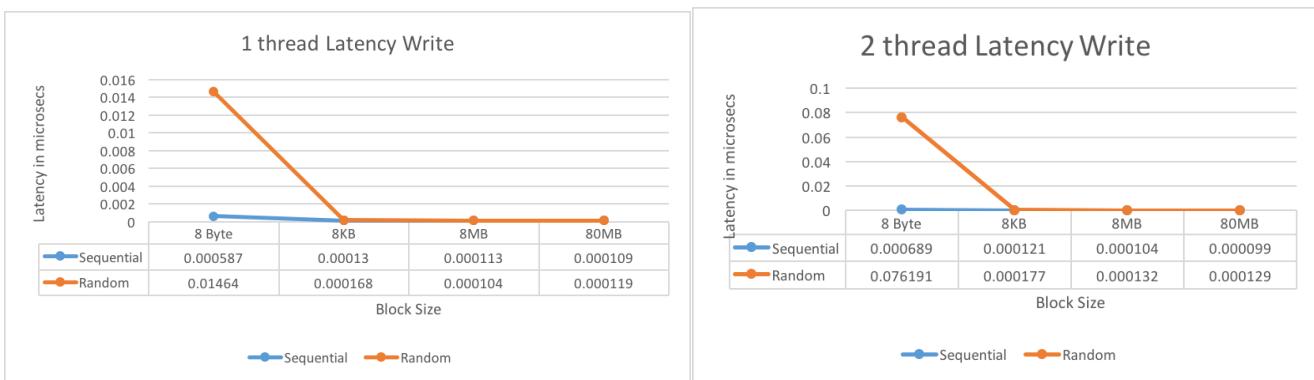
- The throughput for sequential memory write operations increases as the number of threads increases and the size of block increases.
- The throughput of random operations is much less than the sequential ones.
- Optimal Concurrency:**
  - 8 threads with 80 MB block size





### Observations:

- The latency for sequential memory read and write operations doesn't change significantly but decreases by a little amount as the number of thread increases and block size varies.
- The latency for random operations decreases from one thread as the number of thread increases and block size varies.
- Optimal Concurrency:**
  - 8 threads with 8 MB block size



### Observations:

- The latency for sequential memory write operations doesn't change significantly but decreases by a little amount as the number of thread increases and block size varies.
- The latency for random operations decreases from one thread as the number of thread increases and block size varies.
- Optimal Concurrency:**
  - 8 threads with 80 MB block size

- Theoretical memory bandwidth of your memory
 
$$\begin{aligned}
 &= \text{cpu clock cyle} * \text{type of DDR} * 8 \text{ IPC} \\
 &= 2.29 * 3 * 8 \\
 &= 54.96 \text{ GB/sec}
 \end{aligned}$$

```

[cc@pa1-sg current]$ sudo dmidecode --type 17 | more
# dmidecode 3.0
Scanning /dev/mem for entry point.
SMBIOS 2.4 present.

Handle 0x1100, DMI type 17, 21 bytes
Memory Device
    Array Handle: 0x1000
    Error Information Handle: 0x0000
    Total Width: 64 bits
    Data Width: 64 bits
    Size: 4096 MB
    Form Factor: DIMM
    Set: None
    Locator: DIMM 0
    Bank Locator: Not Specified
    Type: RAM
    Type Detail: None

[cc@pa1-sg current]$ dmidecode --type 17 | grep -i speed
/dev/mem: Permission denied
[cc@pa1-sg current]$ sudo dmidecode --type 17 | grep -i speed
[cc@pa1-sg current]$ sudo lshw -short -C memory
H/W path          Device      Class           Description
=====
/0/0                  memory      96KiB BIOS
/0/1000                memory      4GiB System Memory
/0/1000/0              memory      4GiB DIMM RAM

```

- Stream Benchmark Results:
  - Best performance achieved: The best rate for stream without the openmp flag is 5221.1 MB/sec. We have achieved 8429 MB/sec.
  - With openmp flag: the best rate for stream is 10642 MB/sec and we have achieved 79% as compared to that

```

cc@pal-sg:~ 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 36094 microseconds.
(= 36094 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: 5221.1 0.031904 0.030645 0.038294
Scale: 5189.8 0.033716 0.030830 0.051713
Add: 6877.8 0.036953 0.034895 0.045938
Triad: 6586.5 0.037120 0.036438 0.040493

Solution Validates: avg error less than 1.000000e-13 on all three arrays
[cc@pal-sg ~]$
```

```

cc@pal-sg:~ will be used to compute the reported bandwidth.

Number of Threads requested = 2
Number of Threads counted = 2

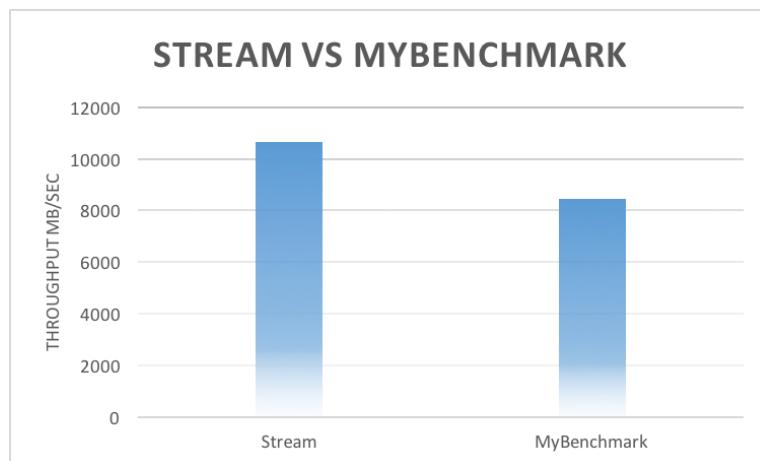
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 22158 microseconds.
(= 22158 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.

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: 10642.6 0.016111 0.015034 0.022340
Scale: 10306.5 0.016060 0.015524 0.018060
Add: 13846.4 0.017804 0.017333 0.018832
Triad: 13109.1 0.018859 0.018308 0.019612

Solution Validates: avg error less than 1.000000e-13 on all three arrays
[cc@pal-sg ~]$
```

○



- What efficiency do you achieve compared to the theoretical performance?
  - Our benchmark performance is approx. 8GB/sec while theoretical memory bandwidth of your memory is 54.96 GB/sec.
  - Efficiency Achieved: 14.5 %

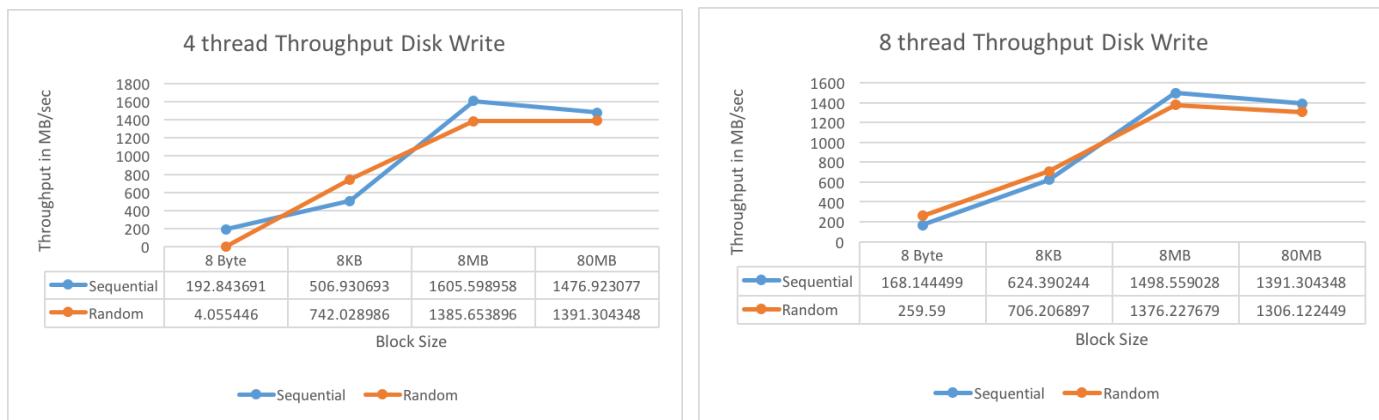
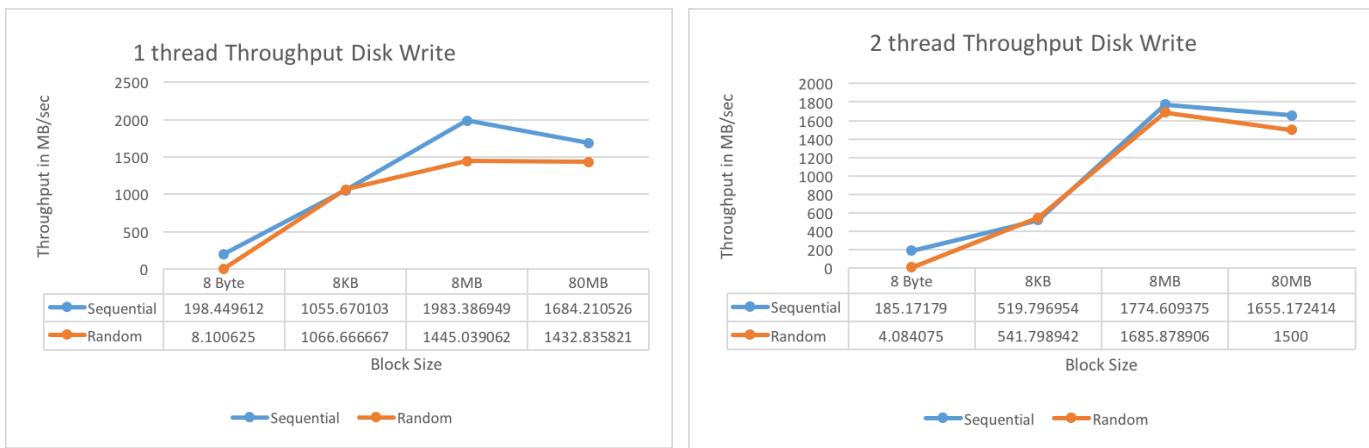
## Disk Benchmarking

- Strong Scaling is used to do all the experiments.
- A large file has been allocated to perform the benchmark. The block sizes of 8B, 8KB, 8MB, 80MB with threads varying as 1, 2, 4 and 8 have been taken.
- File read/write sequentially, randomly and write sequentially and randomly functions have been implemented.
- After exploring different APIs to read and write in a disk, fread and fwrite have been identified for the best performance.
- Below are the results for various operations with varying level of concurrency:



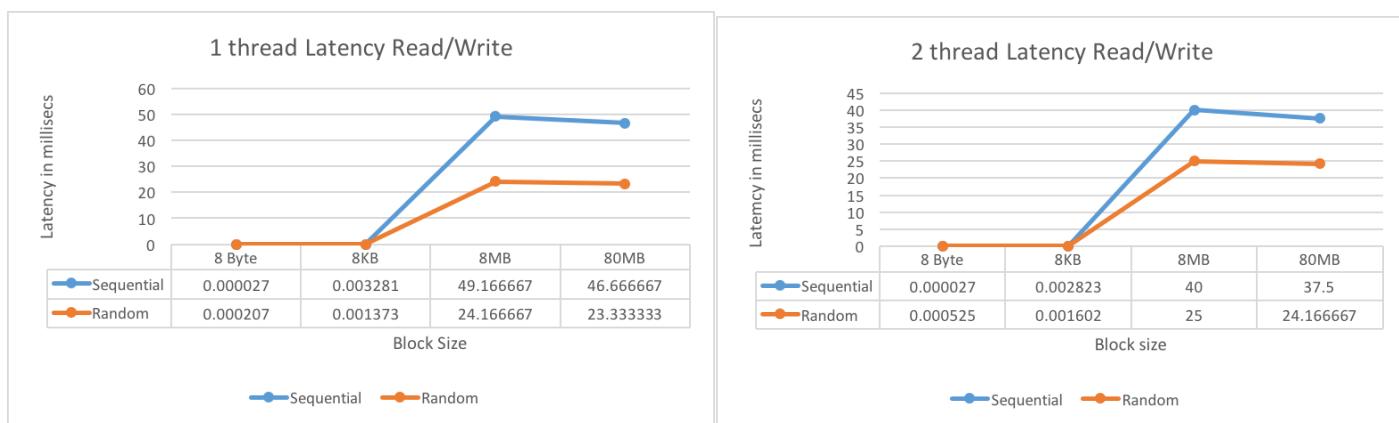
### Observations:

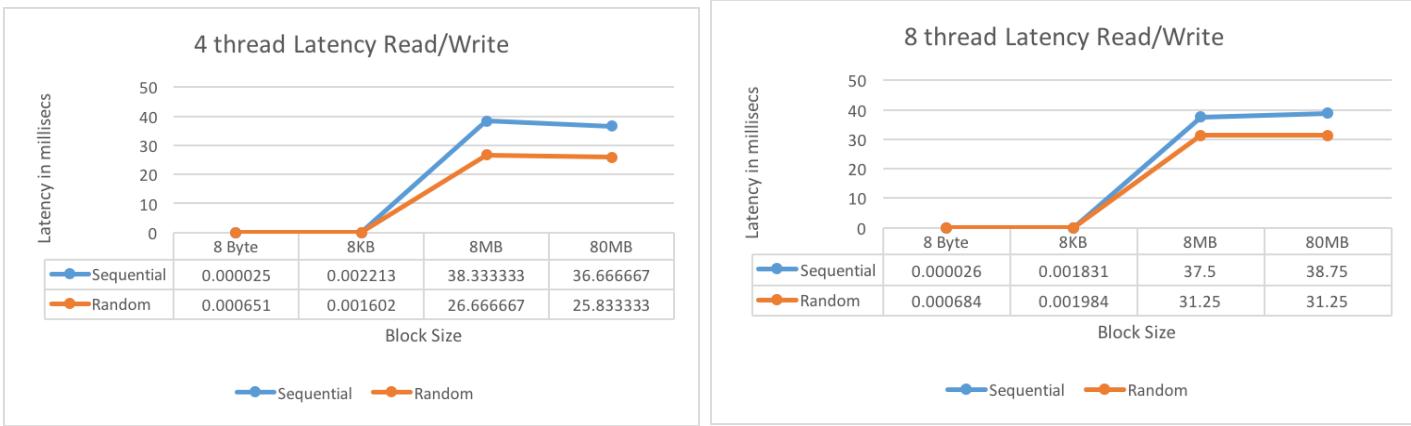
- The throughput for sequential read and write operations increases as the number of thread increases and as block size varies.
- The performance of random operation is high than the sequential except for one thread the performance by random operations is less than sequential ones. Also, the random operation performance is less in case of 8 threads and large block size in comparison to sequential.
- **Optimal Concurrency:**
  - 8threads with 8 KB block size



### Observations:

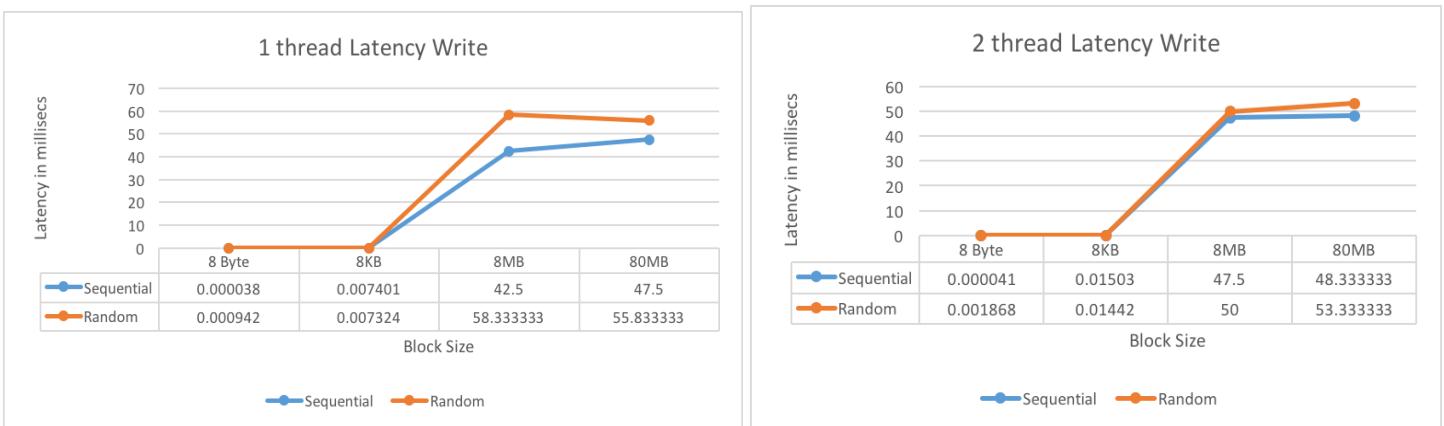
- The throughput for sequential read and write operations increases as the number of thread increases and as block size varies.
- The performance of random operation is a little less than the sequential except for eight thread the performance by random operations is a little high than sequential ones.
- Optimal Concurrency:**
  - 1 thread with 8 MB block size





### Observations:

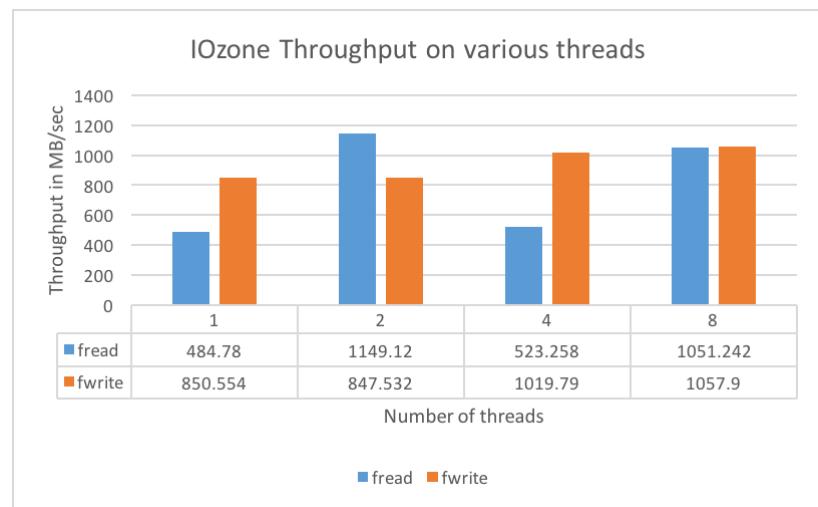
- The latency for sequential disk read and write operations change significantly and increases as the number of thread from 2 to 4 and 8 and block size varies.
- The latency for random operations also increases as the number of thread increases from 2 to 4 and 8 and block size varies.
- The latency of random operations is much less than the sequential ones.
- Optimal Concurrency:**
  - 8 threads with 8B block size



### Observations:

- The latency for sequential disk write operations change significantly and increases as the number of thread from 2 to 4 and 8 increases and block size varies.

- The latency for random operations also increases as the number of thread increases from 2 to 4 and 8 and block size varies.
  - The latency of random operations and sequential ones are almost similar.
  - Optimal Concurrency:**
    - 8threads with 8B block size
- Explain your findings, putting in perspective the hardware you are testing; can you tell if the disk you are evaluating is a spinning hard drive (HDD) or a solid-state memory (SSD) disk?
- As per the latency values which depicts very good performance, we have concluded that it's a solid-state memory (SSD) disk
- Run the IOZone benchmark
- Best performance achieved for 2 threads for fread: 1149 MB/sec
  - Best performance achieved for 3 threads for fwrite: 1019 MB/sec



- Comparison with our benchmark: We are getting much better performance than IOzone.
- The IOzone benchmark is done for 1GB file same as our benchmark & got the following output:

```

× cc@pa1-sg:~/sumedha/iozone3_394/src/current
[cc@pa1-sg current]$ ./iozone -a -s 1048576
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: Mon Oct  9 21:32:01 2017

Auto Mode
File size set to 1048576 KB
Command line used: ./iozone -a -s 1048576
Output is in Kbytes/sec
Time Resolution = 0.0000001 seconds
Processor cache size set to 1024 Kbytes.
Processor cache line size set to 32 bytes.
File stride size set to 17 * record size.

          KB  reclen  write  rewrite   read  reread   read  write  read  readwrt  record  stride   fwrite  frewrite  fread  freread
1048576      4  1549928  5058355  5523700  4084815  6463155  5275690  332214  4941315  8233809  4897139  1107727  370537  5506528  5190122
1048576      8  1412261  1618937  5806938  6463155  5275690  332214  4941315  8233809  4897139  1107727  370537  5506528  5901653
1048576     16  1466691  1052770  5945351  6634797  5961272  360906  5150447  8097006  6317100  1163014  483520  4988209  5944544
1048576     32  740487  1639827  6473057  7651440  6933976  3684606  6220788  8774530  6841983  297642  1635362  5691298  7323915
1048576     64  1435011  1564186  6989526  7998804  7691449  485933  6417116  10359781  7804336  424171  715808  7732804  7866580
1048576    128  1705159  925933  6773657  7203754  7274695  416711  5356916  1072810  6320568  450092  497938  6354770  7704114
1048576    256  422311  637762  6105423  521347  5785755  1417891  5210185  8140674  5357363  410085  896469  5136783  5611944
1048576    512  427520  5674015  6007958  6953332  6944744  1976442  5325312  8035239  563944  982520  462802  5916133  6678015
1048576   1024  722866  30015  6178541  7249848  7539709  506106  6125593  833740  6163200  427314  1204288  6425882  7118217
1048576   2048  1345100  828521  6365734  6475163  5899626  35579  6164595  8185613  6672494  1195511  1494899  6681333  7547355
1048576   4096  503648  294474  6507642  7119462  6447266  1325579  6095106  7838068  6600127  397017  852302  6405121  7294590
1048576   8192  1227477  439698  6501707  6102077  7075742  565807  6612114  8150676  7060781  455991  1217305  6431088  7294144
1048576  16384  281137  1174338  5373899  5855678  5824711  1067169  5345844  4035935  5158745  430962  744893  5276278  5555370

iozone test complete.

```

Below is the output for throughput for 1,2,4 8 threads separately:

<pre>[cc@pal-sg current]\$ ./iozone -i 0 -t 1 I ozone: 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: Mon Oct 9 21:43:09 2017  Command line used: ./iozone -i 0 -t 1 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. Throughput test with 1 process Each process writes a 512 Kbyte file in 4 Kbyte records  Children see throughput for 1 initial writers = 850554.06 KB/sec Parent sees throughput for 1 initial writers = 1564.81 KB/sec Min throughput per process = 850554.06 KB/sec Max throughput per process = 850554.06 KB/sec Avg throughput per process = 850554.06 KB/sec Min xfer = 512.00 KB  Children see throughput for 1 rewriters = 484780.56 KB/sec Parent sees throughput for 1 rewriters = 4776.17 KB/sec Min throughput per process = 484780.56 KB/sec Max throughput per process = 484780.56 KB/sec Avg throughput per process = 484780.56 KB/sec Min xfer = 512.00 KB  ozone test complete.</pre>	<pre>[cc@pal-sg current]\$ ./iozone -i 0 -t 2 I ozone: 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: Mon Oct 9 21:42:08 2017  Command line used: ./iozone -i 0 -t 2 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. Throughput test with 2 processes Each process writes a 512 Kbyte file in 4 Kbyte records  Children see throughput for 2 initial writers = 847532.88 KB/sec Parent sees throughput for 2 initial writers = 786.70 KB/sec Min throughput per process = 847532.88 KB/sec Max throughput per process = 847532.88 KB/sec Avg throughput per process = 423766.44 KB/sec Min xfer = 512.00 KB  Children see throughput for 2 rewriters = 2348392.50 KB/sec Parent sees throughput for 2 rewriters = 1321.04 KB/sec Min throughput per process = 1149120.75 KB/sec Max throughput per process = 1199171.75 KB/sec Avg throughput per process = 1174146.25 KB/sec Min xfer = 492.00 KB  ozone test complete.</pre>
<pre>[cc@pal-sg current]\$ ./iozone -i 0 -t 4 I ozone: 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: Mon Oct 9 21:42:31 2017  Command line used: ./iozone -i 0 -t 4 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. Throughput test with 4 processes Each process writes a 512 Kbyte file in 4 Kbyte records  Children see throughput for 4 initial writers = 1903794.50 KB/sec Parent sees throughput for 4 initial writers = 2541.09 KB/sec Min throughput per process = 884002.56 KB/sec Max throughput per process = 1019791.94 KB/sec Avg throughput per process = 475948.62 KB/sec Min xfer = 412.00 KB  Children see throughput for 4 rewriters = 1029644.44 KB/sec Parent sees throughput for 4 rewriters = 2309.29 KB/sec Min throughput per process = 0.00 KB/sec Max throughput per process = 523258.66 KB/sec Avg throughput per process = 257411.11 KB/sec Min xfer = 0.00 KB  ozone test complete.</pre>	<pre>[cc@pal-sg current]\$ ./iozone -i 0 -t 8 I ozone: 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: Mon Oct 9 21:42:50 2017  Command line used: ./iozone -i 0 -t 8 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. Throughput test with 8 processes Each process writes a 512 Kbyte file in 4 Kbyte records  Children see throughput for 8 initial writers = 1907639.88 KB/sec Parent sees throughput for 8 initial writers = 491.10 KB/sec Min throughput per process = 0.00 KB/sec Max throughput per process = 1057975.00 KB/sec Avg throughput per process = 238454.98 KB/sec Min xfer = 0.00 KB  Children see throughput for 8 rewriters = 2094782.50 KB/sec Parent sees throughput for 8 rewriters = 1898.97 KB/sec Min throughput per process = 0.00 KB/sec Max throughput per process = 1051242.25 KB/sec Avg throughput per process = 261847.81 KB/sec Min xfer = 0.00 KB  ozone test complete.</pre>

- The theoretical performance is generally advertised by the manufacturer.  
After trying many commands, this data I got which just specifies the size of the disk.

```
[cc@pa1-sg sumedha]$ sudo lshw -class disk -class storage
*-ide
  description: IDE interface
  product: 82371SB PIIX3 IDE [Notoma/Triton II]
  vendor: Intel Corporation
  physical id: 1.1
  bus info: pci@0000:00:01.1
  version: 00
  width: 32 bits
  clock: 33MHz
  capabilities: ide bus_master
  configuration: driver=ata_pmix latency=0
  resources: irq=8 iport:1f0(size=8) iport:3f6 iport:170(size=8) iport:3f6 iport:c0@0(size=16)
*-scsi
  description: SCSI storage controller
  product: Virtio block device
  vendor: Red Hat, Inc
  physical id: 4
  bus info: pci@0000:00:04.0
  version: 00
  width: 32 bits
  clock: 33MHz
  capabilities: scsi msix bus_master cap_list
  configuration: driver=virtio-pci latency=0
  resources: irq:11 iport:c0@0(size=64) memory:feb2000-febd2fff
*-virtio1
  description: Virtual I/O device
  physical id: 0
  bus info: virtio@1
  logical name: /dev/vda
  size: 40GB (42GB)
  capabilities: partitioned partitioned:dos
  configuration: driver=virtio_blk logicalsectorsize=512 sectorsize=512 signature=000389fa
*-ppn0@0
  product: PnP device PNP0700
  physical id: 4
  capabilities: pnp
[cc@pa1-sg dev]$ sudo lshw -short -C disk
H/W path      Device      Class      Description
=====
/0/100/4/0    /dev/vda    disk      42GB Virtual I/O device
```

The performance that we found for Seagate ST9250610NS is:

Average Latency: 4.16 ms, 7200 RPM and 115 MB/sec data transfer rate.