Benchmark Performance Evaluation

A20375099 Yuan-An Liu, A20382007 Sharel Clavy Pereira

1. Introduction

All the experiments have been carried out on on Chameleon KVM on 8 virtual core instances with 16GB of RAM and 160GB disk. GPU benchmarks have been done on a Tesla K80 GPU on Chameleon baremetal instance. All the tables and graphs are properly marked with units and brief description for easier understanding of what is going on in the experiments.

```
[cc@pa1-liu-pereira CPU]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                8
On-line CPU(s) list:   0-7
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             8
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 60
Model name:            Intel Core Processor (Haswell)
Stepping:              1
CPU MHz:               2299.998
BogoMIPS:              4599.99
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              4096K
NUMA node0 CPU(s):     0-7
Flags:                 fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmo
v pat pse36 clflush mmx fxsr sse sse2 ss syscall nx pdpe1gb rdtscp lm constant_tsc
rep_good nopl eagerfpu pni pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe
 popcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm fsgsbas
e bmi1 avx2 smep bmi2 erms invpcid xsaveopt
```

2. Experimental Results

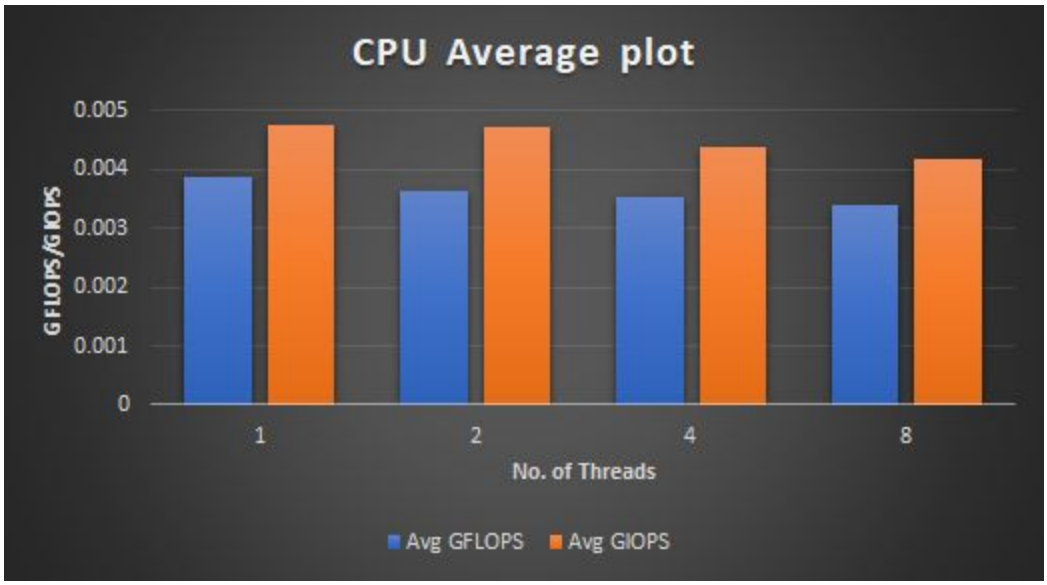
In this section we discuss the results gathered for different benchmarks along with the theoretical results and the trade offs achieved.

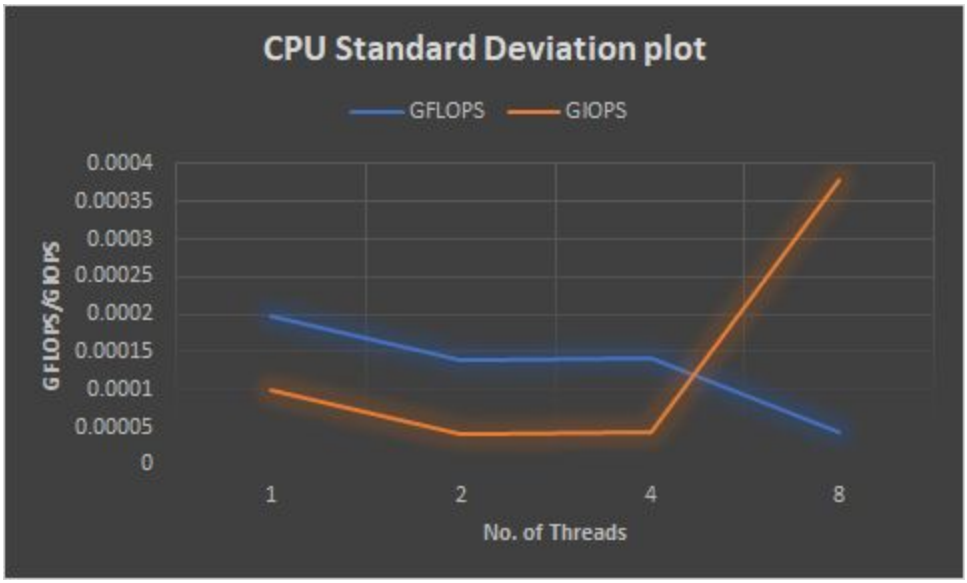
2.1 CPU Benchmark

We are measuring double precision floating point operations and integer operations, per second, in terms of GFLOPS and GIOPS using 1,2,4,8 threads.

Average and **Standard deviation** for are as follows:

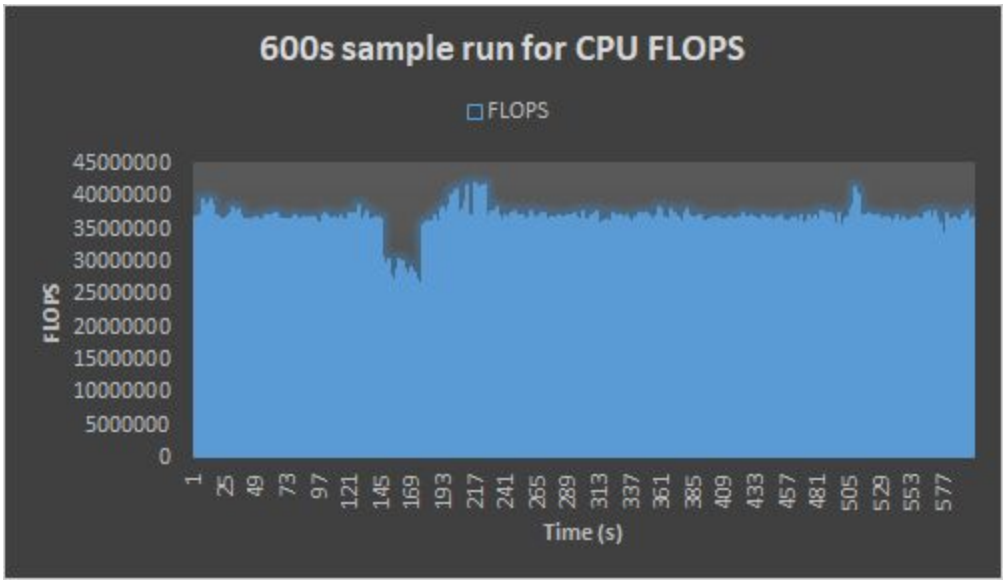
	No. of Threads	*Average*	*Standard Deviation*
	1	0.003867	0.00019798
	2	0.00363	0.000140105
	4	0.003546	0.000141643
GFLOPS	8	0.003404	4.48876E-05
	1	0.004759	9.88786E-05
	2	0.004709	4.02315E-05
	4	0.004387	4.48234E-05
GIOPS	8	0.004173	0.000377654

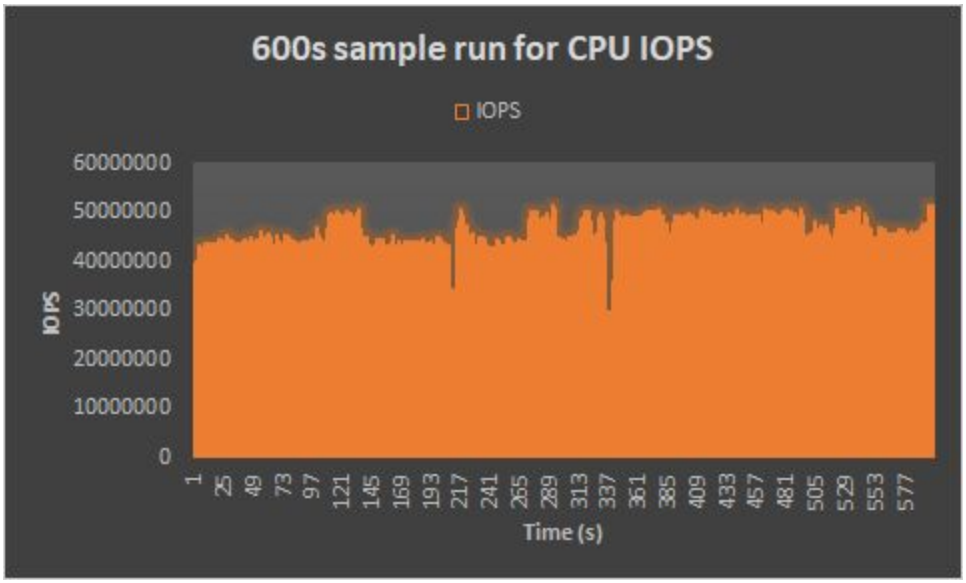




Results of 600 samples for FLOPS/IOPS with 8 threads for a 10min period:

We can notice that results are almost similar for quite a lot of time throughout the time period of 600s except for a few dips which might be caused because of different actions taking place in the operating system during the process. No process was invoked by any external process during this experiment.





We have carried out GFLOPS and GIOPS calculation for multiple thread processes. Above plots and table define the results obtained. We notice that an average of 0.003404GFLOPS and 0.004173GIOPS is achieved with multiple operations.

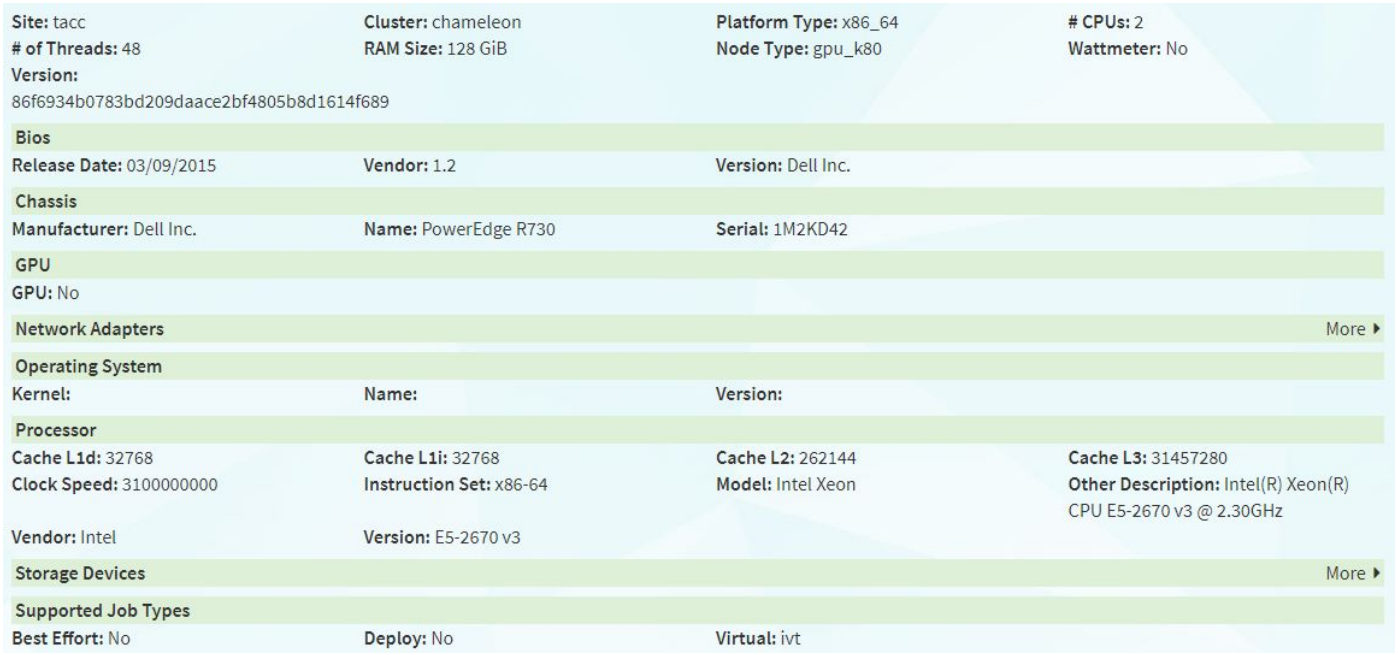
Linpack Benchmark results are as below:

We also notice that in the Linpack results we have 1.528e-01GFLOPS . We also can conclude that results varies with multiple operations and multithreading in place. As we scale up, relative value of operations completed varies. Full Result please visit file at CPU/linpack.txt

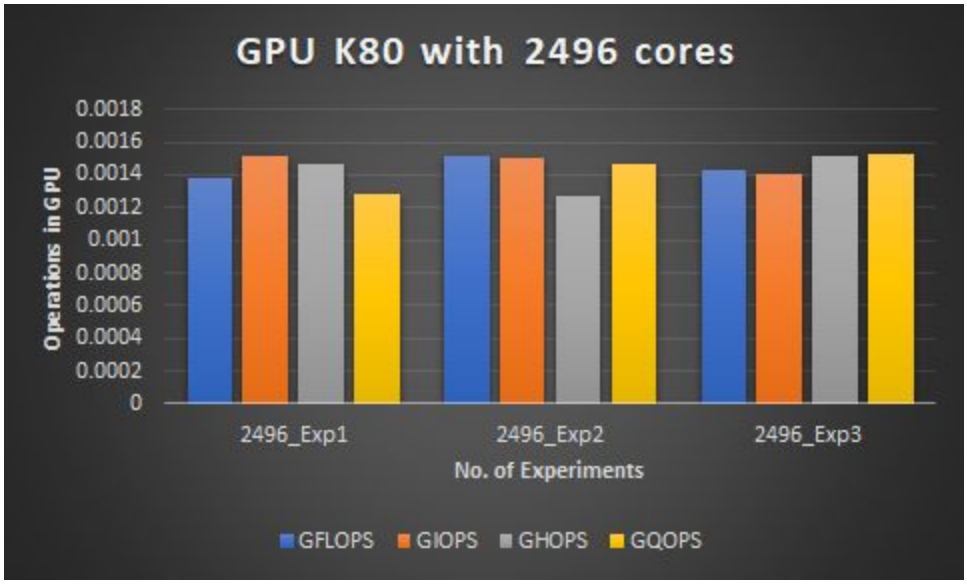
```
--------------------------------------------------------------------------------
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00R2R2          35     4     4     1               0.00             1.179e-01
HPL_pdgesv() start time Tue Sep 26 19:53:52 2017

HPL_pdgesv() end time   Tue Sep 26 19:53:52 2017

--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV-
Max aggregated wall time rfact . . . :               0.00
+ Max aggregated wall time pfact . . :               0.00
+ Max aggregated wall time mxswp . . :               0.00
Max aggregated wall time update  . . :               0.00
+ Max aggregated wall time laswp . . :               0.00
Max aggregated wall time up tr sv  . :               0.00
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0171791 ...... PASSED
Column=000000004 Fraction=11.4% Gflops=3.792e-01
Column=000000008 Fraction=22.9% Gflops=3.361e-01
Column=000000012 Fraction=34.3% Gflops=3.102e-01
Column=000000016 Fraction=45.7% Gflops=2.475e-01
Column=000000020 Fraction=57.1% Gflops=2.251e-01
Column=000000024 Fraction=68.6% Gflops=2.007e-01
Column=000000028 Fraction=80.0% Gflops=1.783e-01
Column=000000032 Fraction=91.4% Gflops=1.528e-01
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR00R2R4          35     4     4     1               0.00             1.399e-01
HPL_pdgesv() start time Tue Sep 26 19:53:52 2017

HPL_pdgesv() end time   Tue Sep 26 19:53:52 2017

--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV--VVV-
Max aggregated wall time rfact . . . :               0.00
+ Max aggregated wall time pfact . . :               0.00
+ Max aggregated wall time mxswp . . :               0.00
Max aggregated wall time update  . . :               0.00
+ Max aggregated wall time laswp . . :               0.00
Max aggregated wall time up tr sv  . :               0.00
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=        0.0168365 ...... PASSED
================================================================================

Finished    864 tests with the following results:
            864 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
================================================================================
```

2.2 GPU Benchmark

Results of multiple experiments are as follows: (Mapping one per thread per core of a 2496 core K80 GPU). Experiment is measured for double precision floating point operations , integer operations, half-precision operations and quarter-precision operations.



Experiment No.	GFLOPS	GIOPS	GHOPS	GQOPS
2496_Exp1	0.00138	0.00151	0.00147	0.00128
2496_Exp2	0.00151	0.0015	0.00127	0.00147
2496_Exp3	0.00143	0.0014	0.00151	0.00153



For a **NVIDIA Tesla K80** processor with dual processing units the theoretical speed for a double precision floating point performance is 1.87 TFLOPS. We have achieved about an average of 0.0014 GFLOPS in our experiment. Since we carried out the experiments by assigning only one thread per core(for 2496 cores) of the GPU , the figurative performance obtained is relative to that of the theoretical peak value. Hence we can conclude that the experimental value obtained is debatable with that of theoretical value.

Linpack Benchmark results are as below:

```
---------------------------------------------------------------------------

- The matrix A is randomly generated for each test.
- The following scaled residual check will be computed:
      ||Ax-b||_oo / ( eps * ( || x ||_oo * || A ||_oo + || b ||_oo ) * N )
- The relative machine precision (eps) is taken to be           1.110223e-16
- Computational tests pass if scaled residuals are less than            16.0

Column=000000001 Fraction= 3.4% Gflops=8.200e-04
```

We notice that Linpack has an average of 8.200e-04 GFLOPS for the operations on the GPU. We have derived slightly different figure since we are running only one thread per core of the K80 gpu. However if we change the metrics and add in multiple threads per core with spawning multiple blocks we can probably get a different figure for comparisons.

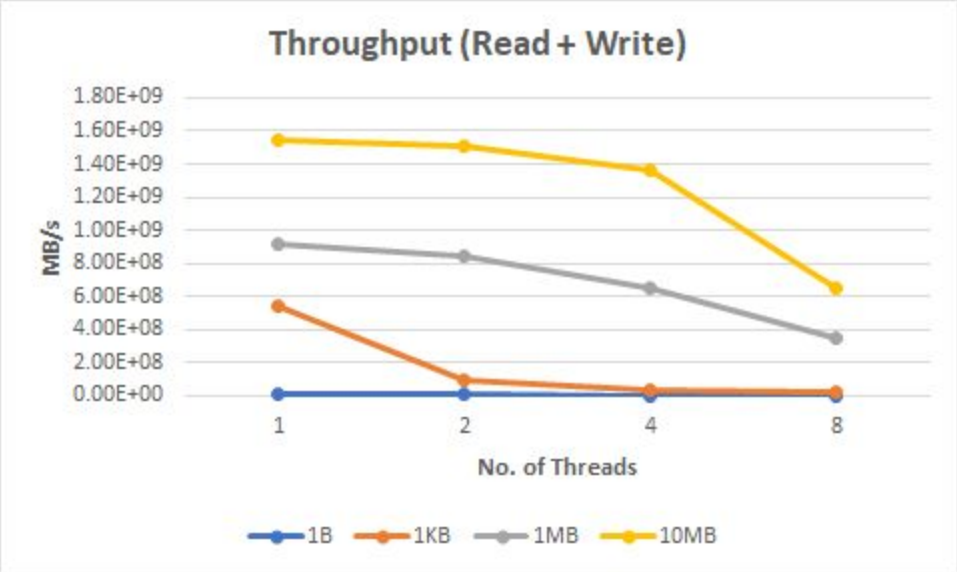
2.3 Memory Benchmark

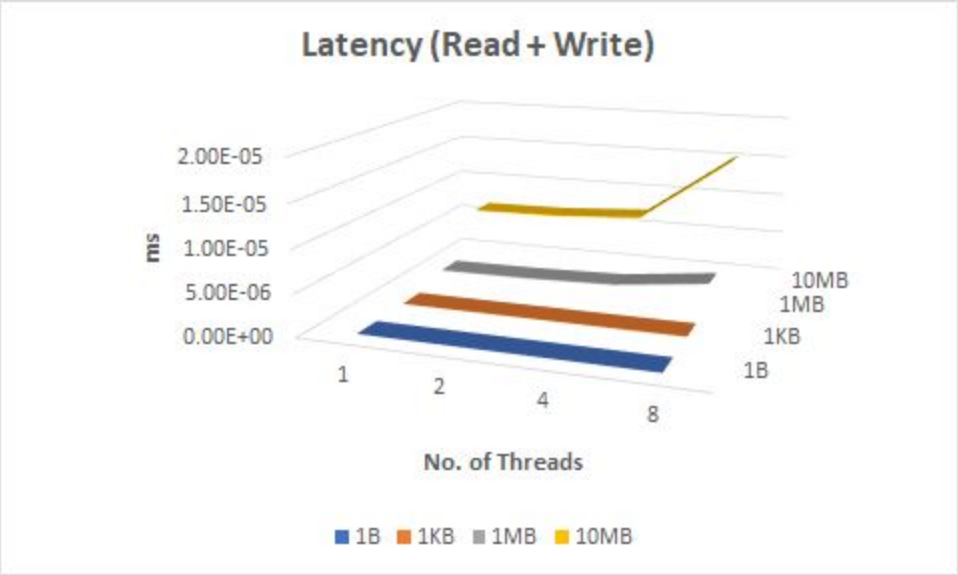
A program is created to measure the speed of the memory by performing various operations like read+write, sequential read access and random read access.Optimum concurrency is achieved by passing multiple threads and blocks of varying size (1B, 1KB, 1MB, 10MB), on a 20MB file.

```
[cc@pa1-liu-pereira ~]$ sudo dmidecode --type memory
# dmidecode 3.0
Getting SMBIOS data from sysfs.
SMBIOS 2.4 present.

Handle 0x1000, DMI type 16, 15 bytes
Physical Memory Array
        Location: Other
        Use: System Memory
        Error Correction Type: Multi-bit ECC
        Maximum Capacity: 16 GB
        Error Information Handle: Not Provided
        Number Of Devices: 1

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: 16384 MB
        Form Factor: DIMM
        Set: None
        Locator: DIMM 0
        Bank Locator: Not Specified
        Type: RAM
        Type Detail: None
```

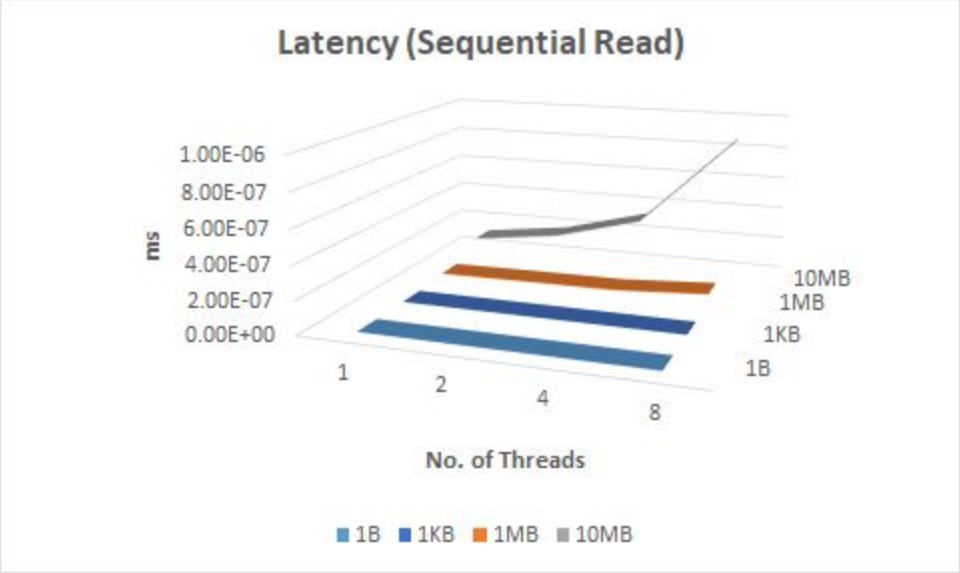
Throughput is measured in **MB/sec** and Latency in **ms** for Memory benchmarks.

Read + Write	No. of Threads	1B	1KB	1MB	10MB
	1	9.06E+06	5.42E+08	9.21E+08	1.55E+09
	2	5.33E+06	8.95E+07	8.40E+08	1.51E+09
	4	1.73E+06	3.87E+07	6.55E+08	1.37E+09
Throughput	8	650956	2.16E+07	3.54E+08	6.53E+08
	1	1.10E-10	1.84E-09	1.09E-06	6.46E-06
	2	1.88E-10	1.12E-08	1.19E-06	6.62E-06
	4	5.77E-10	2.58E-08	1.53E-06	7.32E-06
Latency	8	1.54E-09	4.64E-08	2.83E-06	1.53E-05



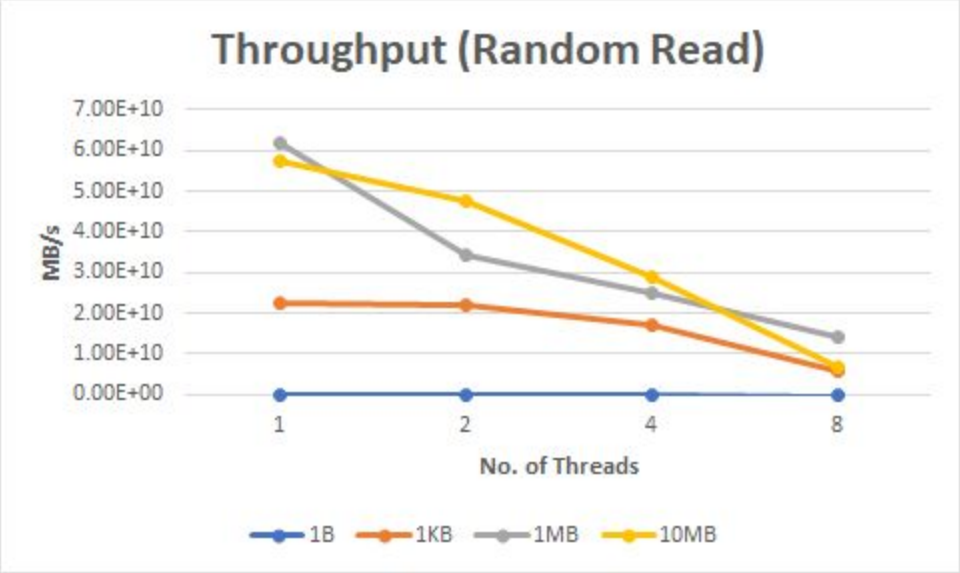


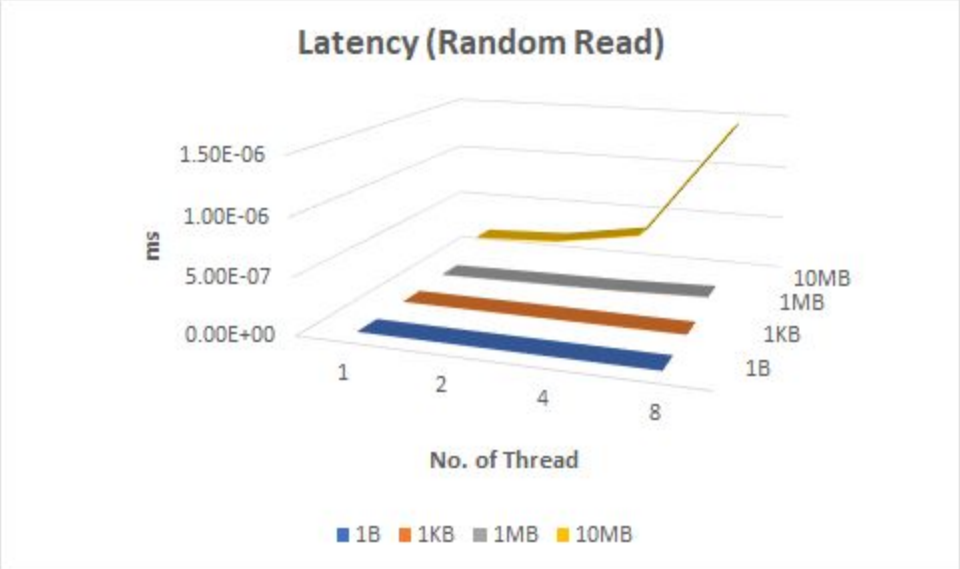
Sequential Read	No. of Threads	1B	1KB	1MB	10MB
	1	1.03E+08	4.39E+10	5.36E+10	9.00E+10
	2	9.41E+07	2.99E+10	3.91E+10	5.53E+10
	4	9.18E+07	1.70E+10	3.32E+10	3.05E+10
Throughput	8	8.94E+07	9.79E+09	1.54E+10	1.16E+10
	1	9.67E-12	2.28E-11	1.87E-08	1.11E-07
	2	1.06E-11	3.34E-11	2.55E-08	1.81E-07
	4	1.09E-11	5.87E-11	3.01E-08	3.27E-07
Latency	8	1.12E-11	1.02E-10	6.50E-08	8.62E-07



Random Read	No. of Threads	1B	1KB	1MB	10MB
	1	9.29E+07	2.26E+10	6.20E+10	5.75E+10
	2	9.80E+07	2.20E+10	3.42E+10	4.74E+10
	4	9.36E+07	1.72E+10	2.49E+10	2.88E+10
Throughput	8	6.94E+07	5.80E+09	1.42E+10	6.93E+09

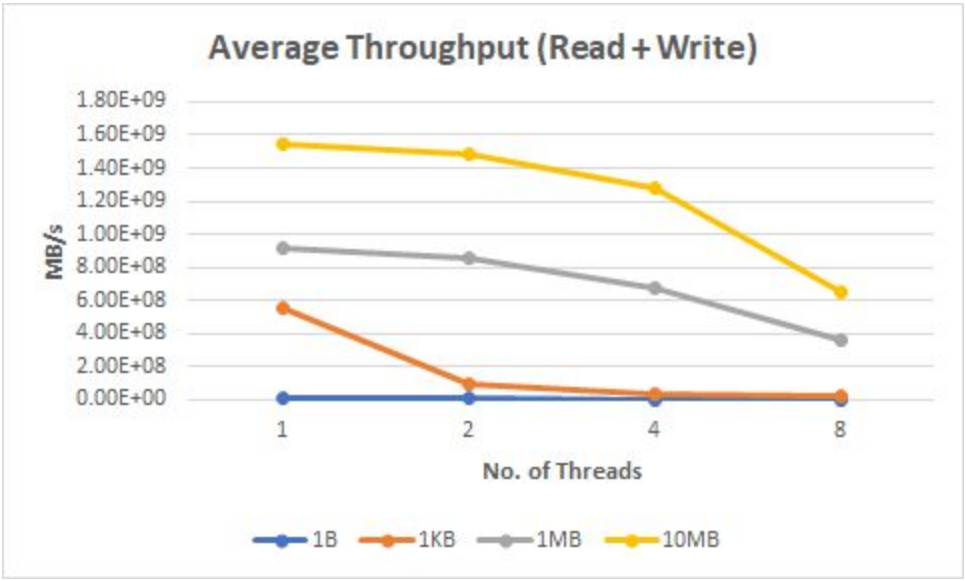
	1	1.08E-11	4.42E-11	1.61E-08	1.74E-07
	2	1.02E-11	4.55E-11	2.93E-08	2.11E-07
	4	1.07E-11	5.82E-11	4.02E-08	3.47E-07
Latency	8	1.44E-11	1.72E-10	7.02E-08	1.44E-06

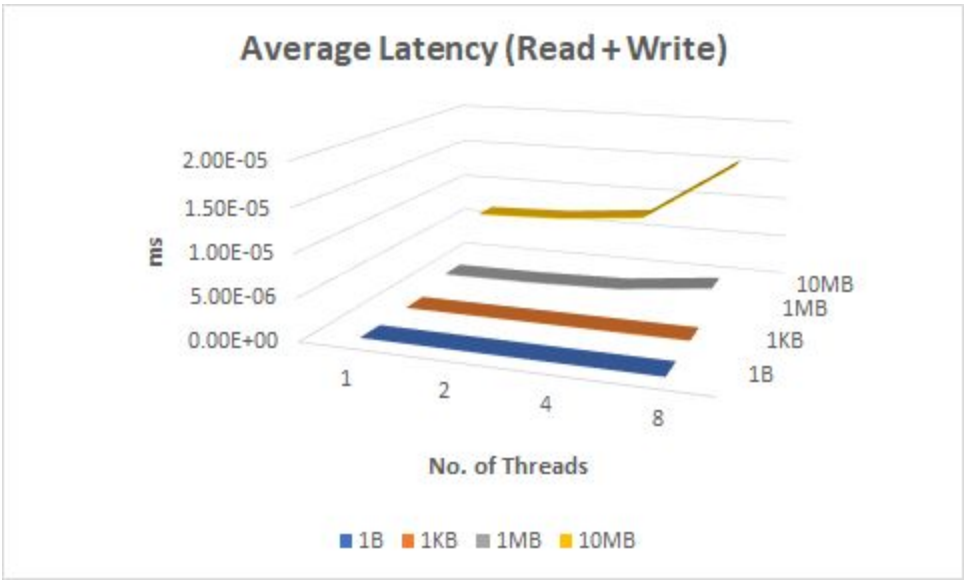




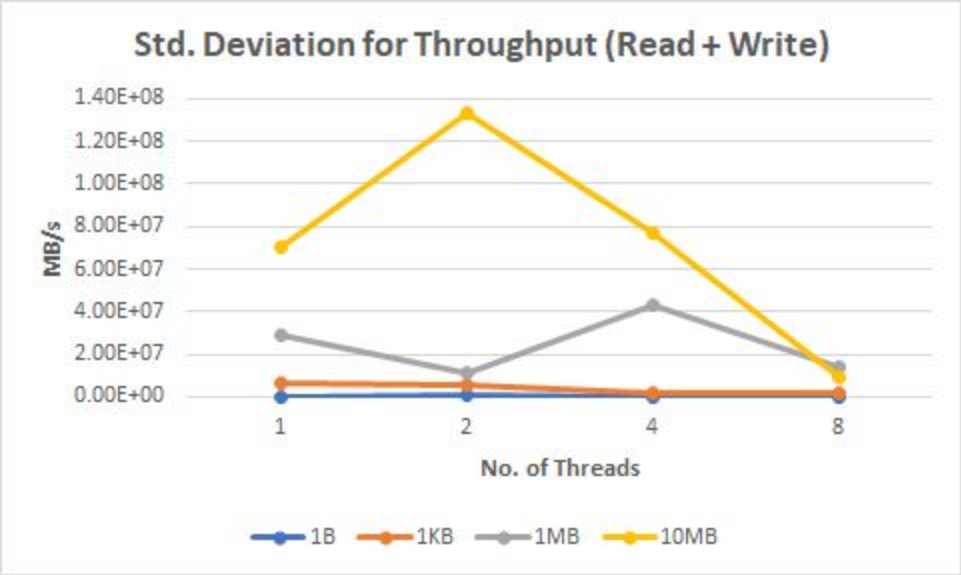
Average and **Standard deviation** for are as follows:

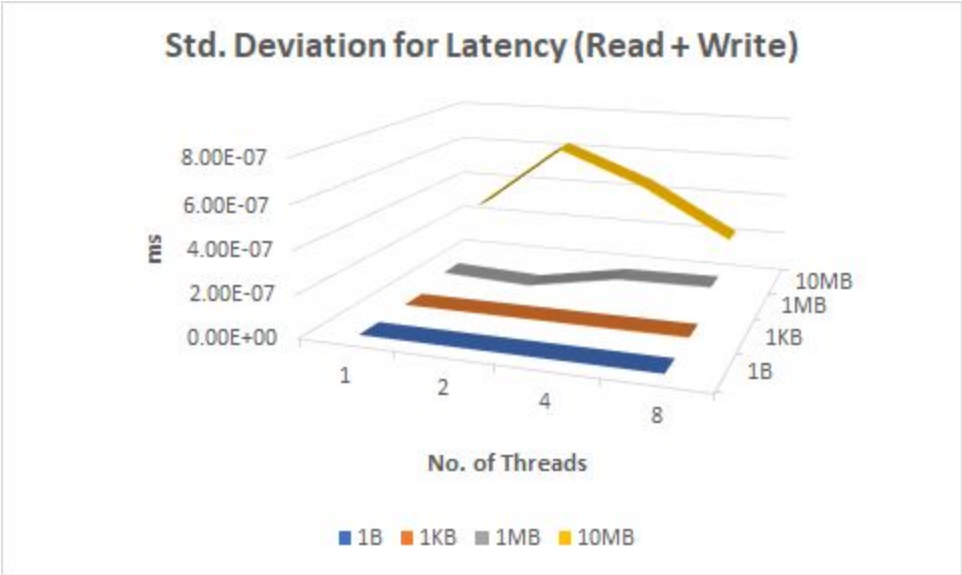
Read + Write (Avg)	No. of Threads	1B	1KB	1MB	10MB
	1	9.33E+06	5.49E+08	9.12E+08	1.55E+09
	2	6.54E+06	9.43E+07	8.51E+08	1.48E+09
	4	1.62E+06	3.63E+07	6.73E+08	1.28E+09
Throughput	8	6.82E+05	2.28E+07	3.66E+08	6.56E+08
	1	1.07E-10	1.82E-09	1.10E-06	6.45E-06
	2	1.56E-10	1.06E-08	1.17E-06	6.79E-06
	4	6.19E-10	2.76E-08	1.49E-06	7.81E-06
Latency	8	1.47E-09	4.41E-08	2.74E-06	1.52E-05





Read + Write (SD)	No. of Threads	1B	1KB	1MB	10MB
	1	2.54E+05	6.72E+06	2.91E+07	7.04E+07
	2	1.11E+06	5.54E+06	1.11E+07	1.33E+08
	4	9.85E+04	2.11E+06	4.35E+07	7.70E+07
Throughput	8	4.00E+04	2.00E+06	1.37E+07	9.51E+06
	1	2.94E-12	2.23E-11	3.55E-08	2.92E-07
	2	2.82E-11	6.14E-10	1.54E-08	6.29E-07
	4	3.64E-11	1.55E-09	9.34E-08	4.62E-07
Latency	8	8.40E-11	3.69E-09	1.01E-07	2.19E-07





From the experiment we have achieved an average of 6.82E+05MB/s for a 1B Read + Write operations running on 8 threads while the average latency is 1.47E-09ms. We have performed experiments by scaling up multiple threads and size of data in memory across multiple operations. Results and graphs are displayed as above for each operation.

STREAM Benchmark results are as below:

According to Linpack results we can achieve an average throughput of 13983.5MB/s. However throughput varies with different operations and with different number of threads. When multiple threads try to write the same file , they have to wait for previous one to complete and hence the throughput differs in these cases.

The result is stored in Memory/stream.txt

```
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 10000000 (elements), Offset = 0 (elements)
Memory per array = 76.3 MiB (= 0.1 GiB).
Total memory required = 228.9 MiB (= 0.2 GiB).
Each kernel will be executed 10 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 11909 microseconds.
   (= 11909 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:           12533.4     0.013056     0.012766     0.013201
Scale:          12477.5     0.013052     0.012823     0.013392
Add:            13983.5     0.017517     0.017163     0.017954
Triad:          13970.5     0.017706     0.017179     0.018359
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
```

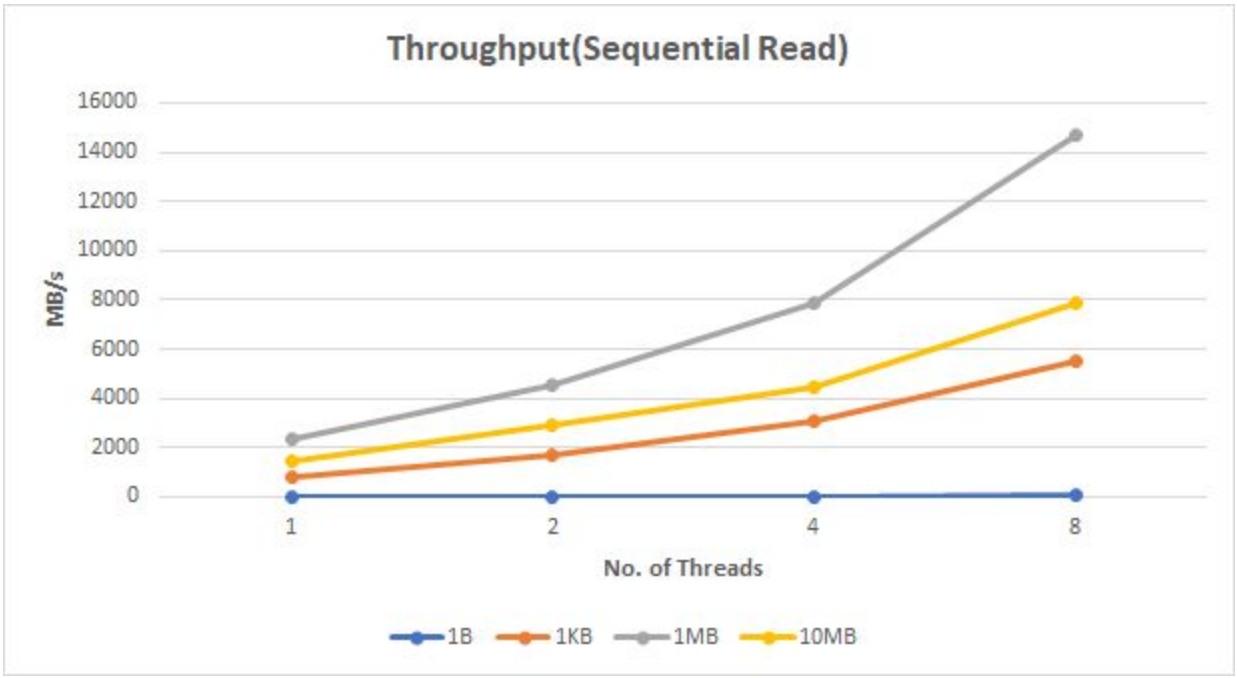
2.4 Disk Benchmark

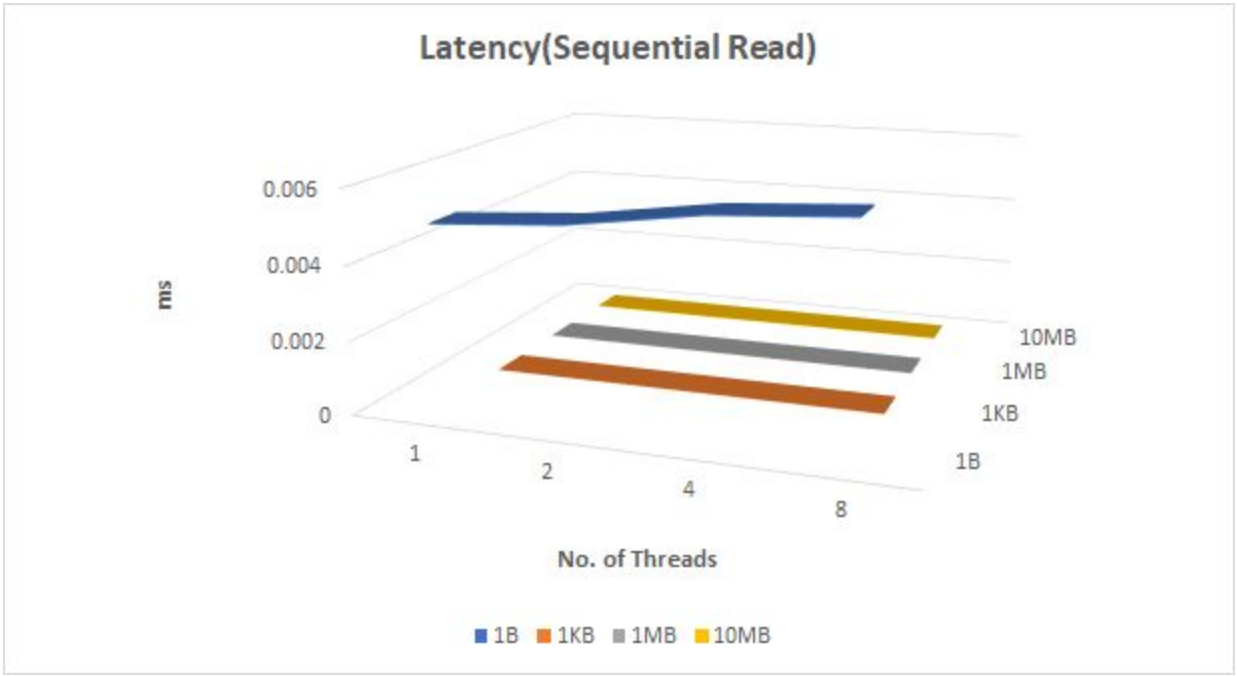
In this section we are measuring the speed of the disk using different parameter space like read+write, sequential read access and random read access. Concurrency is achieved by passing multiple threads with blocks of varying size (1B, 1KB, 1MB, 10MB), using a 20MB file.

```
[cc@pa1-liu-pereira ~]$ lsblk --output NAME,FSTYPE,UUID,RO,RM,MODEL,SIZE,MODE,ROTA,RQ-SIZE,TYPE,DISC-MAX
NAME   FSTYPE UUID                                 RO RM MODEL  SIZE MODE       ROTA RQ-SIZE TYPE DISC-MAX
vda                                                 0  0        160G brw-rw----    1     128 disk       0B
└─vda1 xfs    d613911c-f1e0-4243-903e-535ec22d4891  0  0        160G brw-rw----    1     128 part       0B
[cc@pa1-liu-pereira ~]$
```

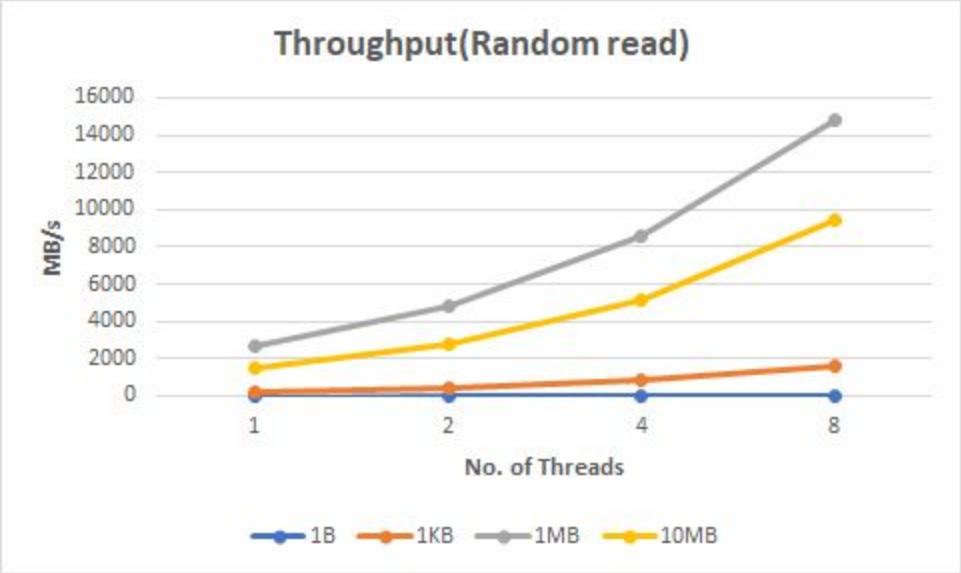
Throughput is measured in MB/sec and Latency in ms for Disk benchmarks.

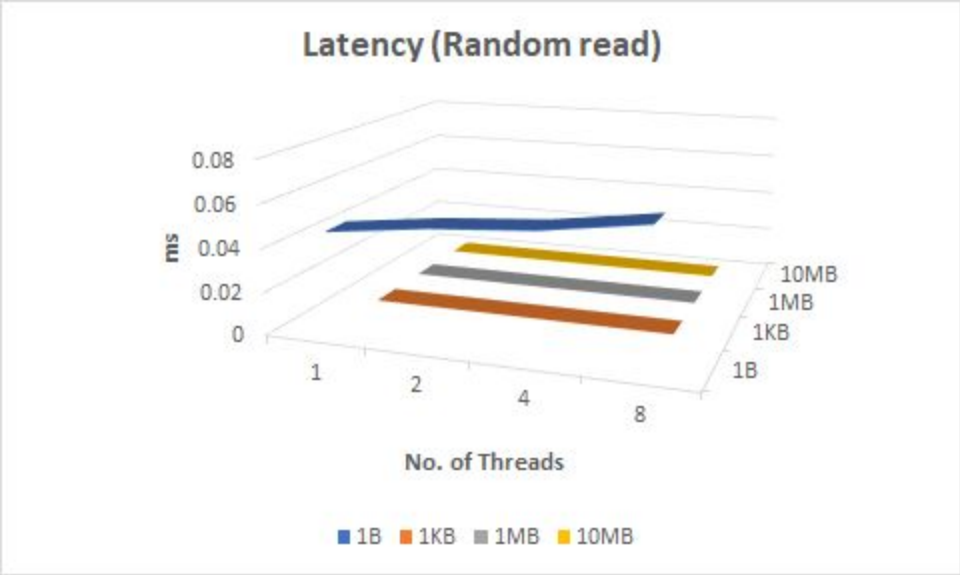
Sequential Read	No. of Threads	1B	1KB	1MB	10MB
	1	3.984067734	811.6934212	2333.734316	1450.112018
	2	7.612959007	1656.305569	4573.442373	2886.601401
	4	13.87236956	3090.466594	7864.811551	4474.282209
Throughput	8	26.73232203	5514.558154	14658.9917	7831.677812
	1	0.005019995	2.41E-08	8.17E-12	1.32E-12
	2	0.005254199	2.36E-08	8.34E-12	1.32E-12
	4	0.005766859	2.53E-08	9.70E-12	1.71E-12
Latency	8	0.005985264	2.83E-08	1.04E-11	1.95E-12


Throughput(Sequential Read)
16000
14000
12000
10000
8000
6000
4000
2000
0
MB/s
1
2
4
8
No. of Threads
1B
1KB
1MB
10MB

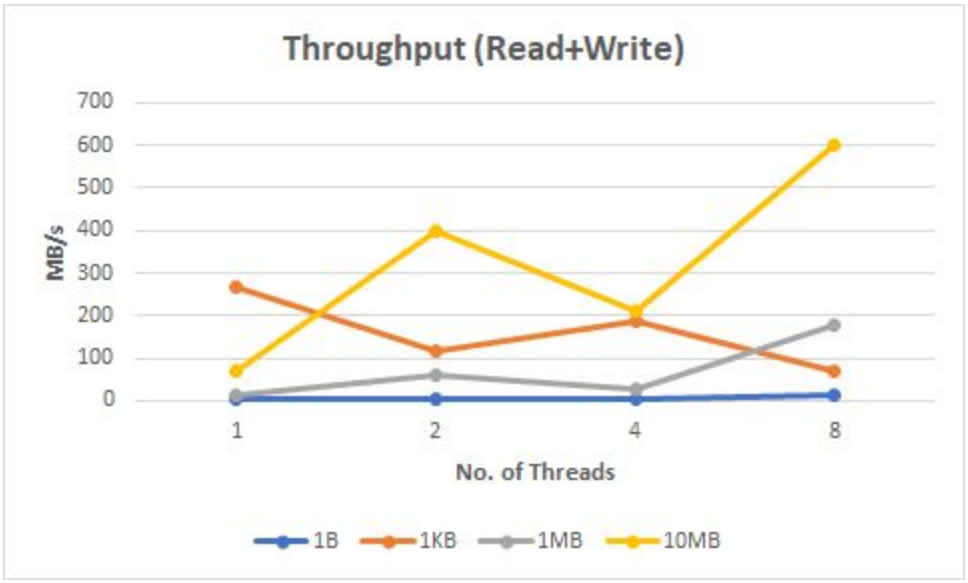

Latency(Sequential Read)
0.006
0.004
0.002
0
ms
1
2
4
8
No. of Threads
10MB
1MB
1KB
1B
1B
1KB
1MB
10MB

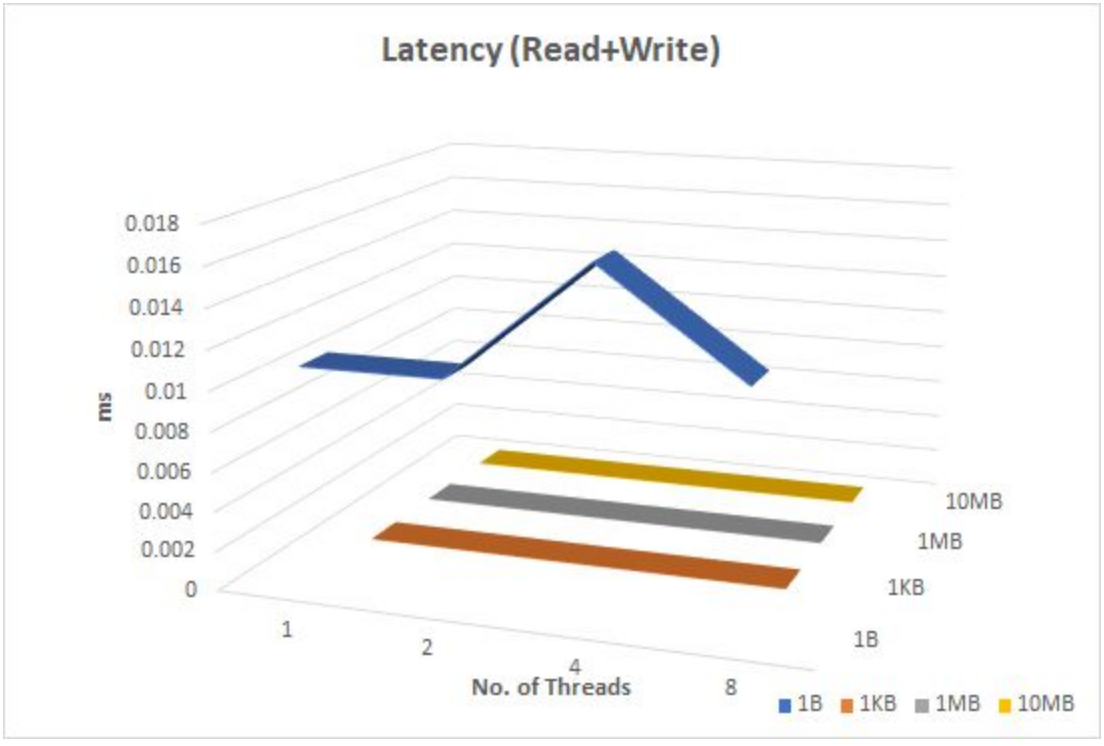
Random Read	No. of Threads	1B	1KB	1MB	10MB
	1	0.429725497	201.205225	2628.092359	1461.862921
	2	0.767411181	422.645677	4833.81814	2804.054018
	4	1.441065593	807.5267435	8545.198767	5161.346849
Throughput	8	2.565619035	1557.465774	14772.57727	9473.568424
	1	0.046541339	9.71E-08	7.26E-12	1.30E-12
	2	0.052123296	9.24E-08	7.89E-12	1.36E-12
	4	0.055514475	9.67E-08	8.93E-12	1.48E-12
Latency	8	0.062363117	1.00E-07	1.03E-11	1.61E-12





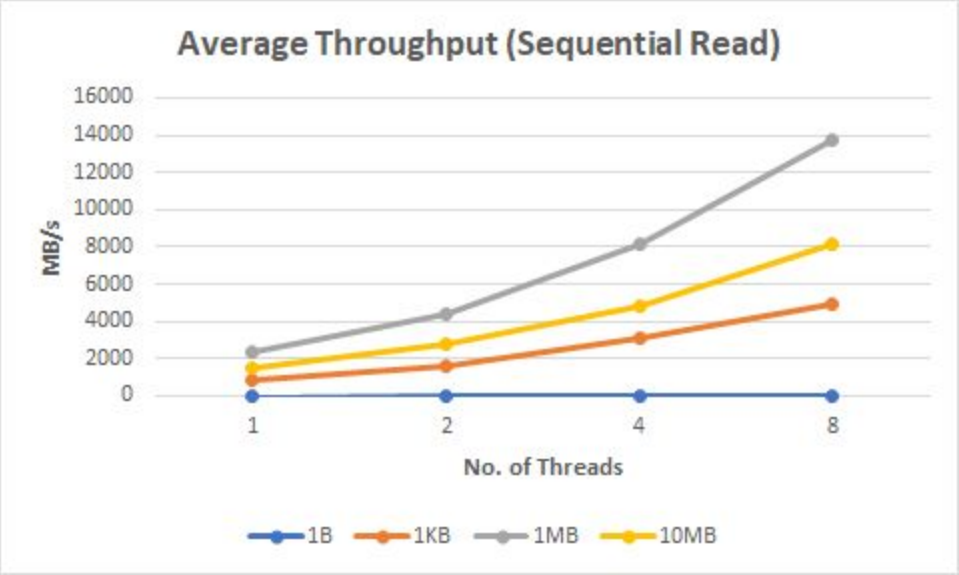
Read and Write	No. of Threads	1B	1KB	1MB	10MB
	1	1.816034767	268.8458223	12.73613664	68.99337669
	2	3.622946185	115.8537101	61.14805387	400.5408916
	4	4.697484638	188.803873	26.99696612	210.244754
Throughput	8	13.34959315	69.91418464	175.7621767	602.5346569
	1	0.011013005	7.26E-08	1.50E-09	2.76E-11
	2	0.011040738	3.37E-07	6.24E-10	9.52E-12
	4	0.017030391	4.14E-07	2.83E-09	3.63E-11
Latency	8	0.011985384	2.23E-06	8.68E-10	2.53E-11

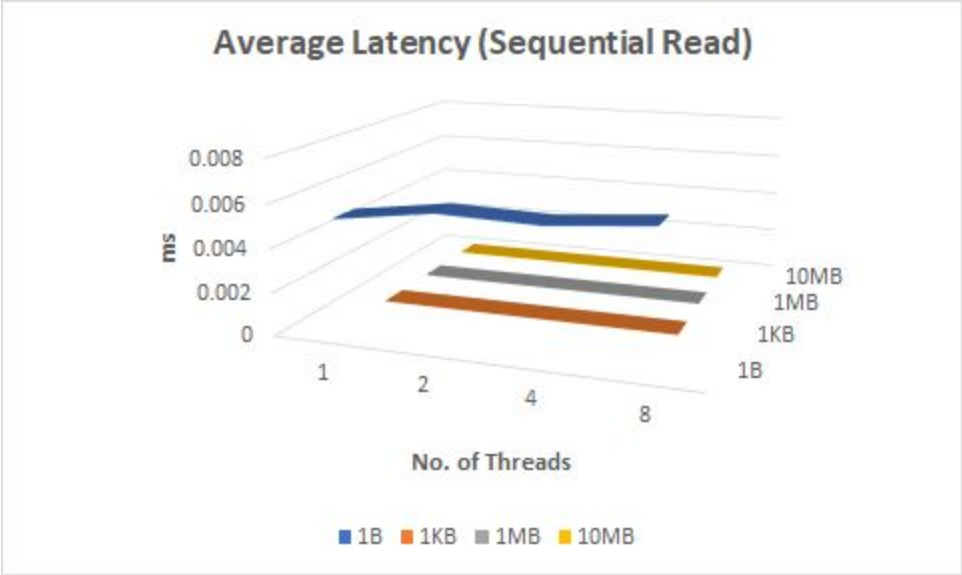

Throughput (Read+Write)
700
600
500
400
300
200
100
0
MB/s
1
2
4
8
No. of Threads
1B
1KB
1MB
10MB


Latency (Read+Write)
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0
ms
1
2
4
8
No. of Threads
10MB
1MB
1KB
1B
1B
1KB
1MB
10MB

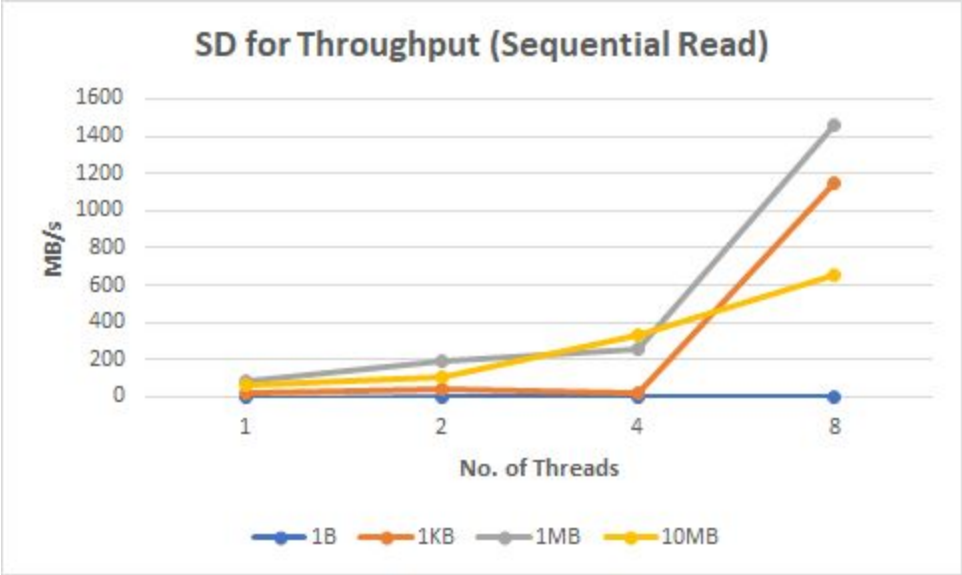
Average and **Standard deviation** for are as follows:

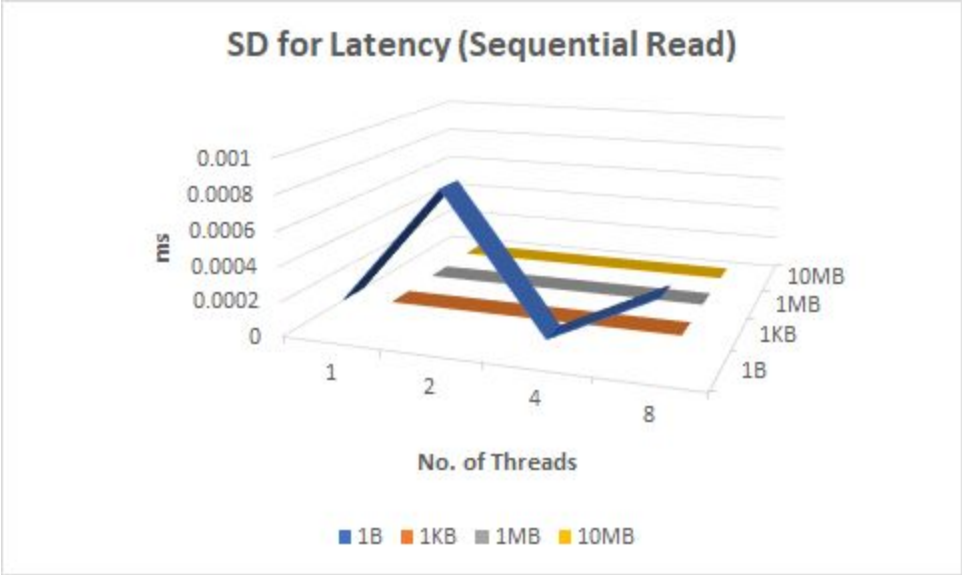
Sequential Read (Avg)	NO. of Threads	1B	1KB	1MB	10MB
	1	3.823210718	829.4546963	2308.53	1524.04
	2	6.903690689	1614.989205	4376.038	2819.4395
	4	13.91614839	3104.835034	8157.283	4861.1999
Throughput	8	26.10821944	4912.340177	13701.9	8112.3075
	1	0.005236138	2.36E-08	8.27E-12	1.25E-12
	2	0.005872229	2.42E-08	8.73E-12	1.35E-12
	4	0.005749992	2.52E-08	9.36E-12	1.58E-12
Latency	8	0.006144557	3.32E-08	1.12E-11	1.89E-12





Sequential Read (SD)	NO. of Threads	1B	1KB	1MB	10MB
	1	0.144756525	17.27838132	83.94254	66.610892
	2	0.941942917	37.44901186	186.5776	101.1686
	4	0.254330658	24.46893778	253.6396	336.41064
Throughput	8	1.617278947	1150.050556	1464.028	653.17958
	1	0.000195435	4.92E-10	3.05E-13	5.57E-14
	2	0.000860673	5.57E-10	3.69E-13	4.96E-14
	4	0.000104622	1.97E-10	2.96E-13	1.13E-13
Latency	8	0.000392808	8.98E-09	1.28E-12	1.46E-13





We have achieved an average throughput of 13701.9MB/s for a 1MB Sequential Read operation with an average latency of 1.28E-12ms with multithreading of 8 threads. From the results we also can deduce that higher throughput is achieved for higher number of threads while sequentially reading a block of data. We also notice that the latency reduces as multiple threads are spawned for a particular operation.

Though it seems to be performing faster, average SSD speed might be faster than this for the given configurations. We can thus say that HDD is being used here.

Iozone Benchmark results are as below:

Below results shows the output for the Iozone benchmark. We see that a average throughput of 1306.3MB/s is achieved in this case. Also the throughput decreases if multiple processes are trying to write to the same file because latency increases in this case. Threads will have to wait for other threads to finish the process to start writing or any other data change operation to the file.

Full Result please visit file at Disk/iozone_output.xls

./iozone -a -b output.xls

The top row is records sizes, the left column is file sizes

Writer Report

	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384
64	703068	877797	889431	969761	955947								
128	800337	978253	1152564	955616	1155043	1102844							
256	937924	1219045	1108314	1248819	1272498	1260547	1230218						
512	977103	1190529	1237168	1147892	1218910	1344020	1234323	1340664					
1024	1017811	1194770	1306332	1416636	1439908	1354535	1254808	1317957	1438461				
2048	1060642	1272054	1376193	1411470	1481072	1473198	1334295	2306143	2334979	3003677			
4096	1509149	2183288	2858164	2957047	2950952	3105653	2981164	2856263	1476592	1463386	1407516		
8192	937949	1411726	1681761	1850102	1884087	1405604	1486996	1684234	1986019	1365551	1612617	1229689	
16384	1068270	1095256	1159767	1545375	1614808	1500366	1485543	1592837	1595166	1490117	1757000	1457903	1338131
32768	0	0	0	0	1387658	1472600	1502363	2315782	1473310	1471858	1472505	1417118	1404720
65536	0	0	0	0	1520773	1582880	1474707	1567731	1633331	2108496	1667373	1698929	2004216
131072	0	0	0	0	1586808	2055547	1821023	1885878	1934011	1858150	2019974	1874094	1793423
262144	0	0	0	0	2063606	2229587	2114897	1984152	1637970	2234798	1833776	1852318	1529570
524288	0	0	0	0	1852856	2192234	2163322	2140021	2146197	2178469	2171199	2155417	1972402

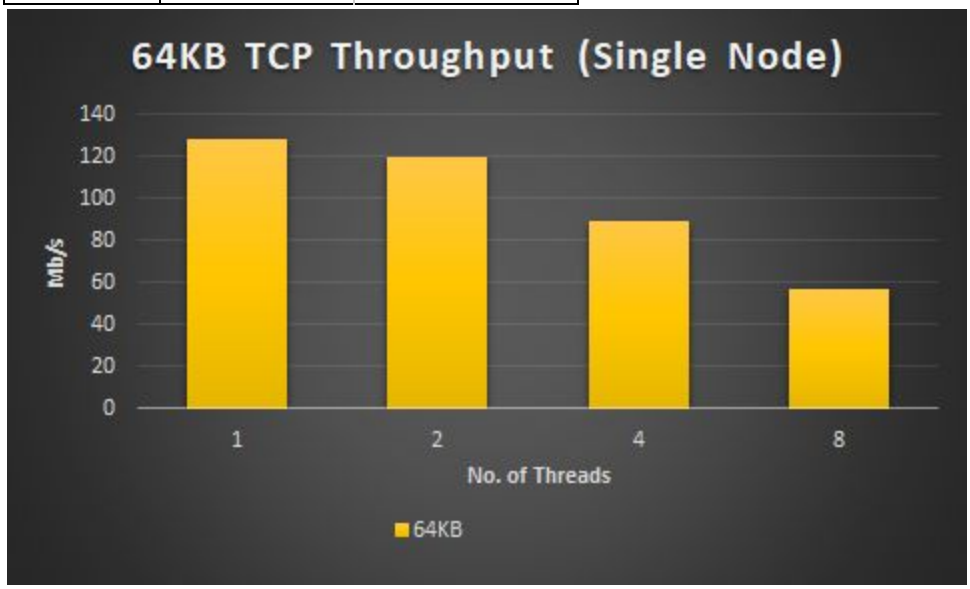
Re-writer Report

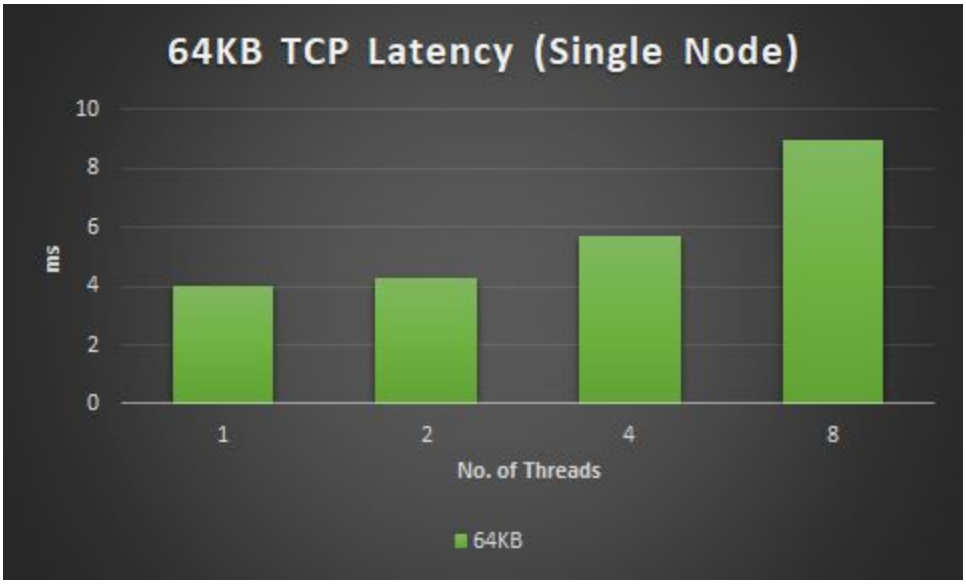
	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384
64	1933893	1599680	1828508	2067979	2133730								
128	1294270	1858934	2326073	2286447	2558895	2608629							
256	1826695	2345409	2533571	2692393	2607399	2539563	2754556						
512	1909204	1970519	2572435	2449212	2860238	2926501	2796910	2845081					
1024	1822368	2311849	2348509	2716948	2553783	2604895	2744728	2625597	2625597				
2048	1859773	2255875	2656292	2798201	2779189	2921939	2727131	4751580	5238385	6132138			
4096	3767675	5051705	3761077	5704175	6043279	6195843	5978090	1970212	2892815	2957556	2919360		
8192	2084454	3030813	3123099	3953573	1205232	2842277	2966955	3402019	4037664	3161607	2925775	2080793	
16384	1822903	1680930	1872016	2677993	3034132	2819514	3196878	2929368	2728933	2928868	2099470	2629524	2278436
32768	0	0	0	0	2309905	2437906	2840011	2146752	2385396	3229313	2250706	2222751	2241456
65536	0	0	0	0	2291626	2335812	2321352	2359876	2432659	2398652	2342340	3422037	2112839
131072	0	0	0	0	3257661	2456089	2500324	2452736	3005627	2545486	2544649	2159802	2496940
262144	0	0	0	0	2720150	2929641	2766395	2579573	2826531	2838190	2845962	2822482	2388073
524288	0	0	0	0	3103546	3554403	3246568	3328092	3186949	3173982	3275427	3297678	3014397

2.5 Network Benchmark

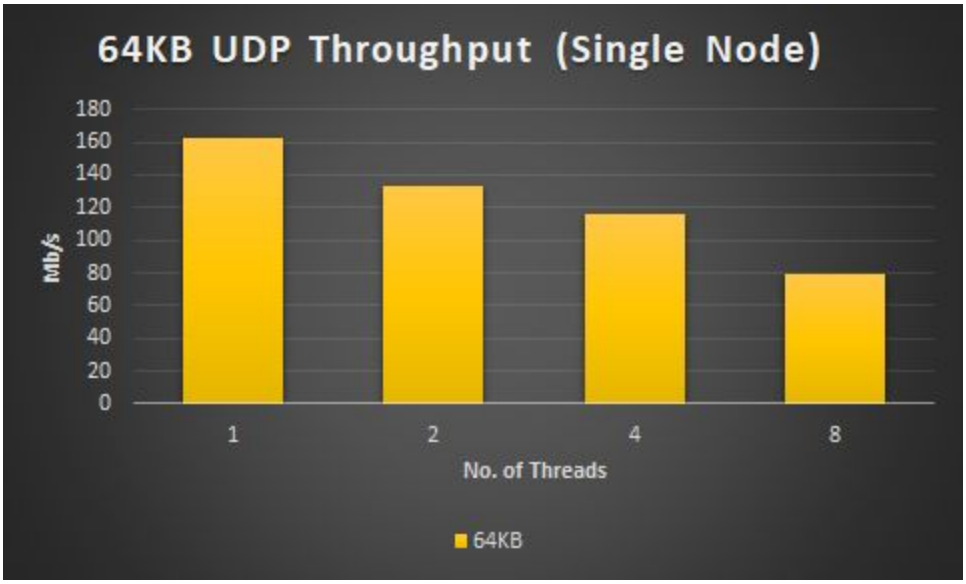
A measurement of network speed over the loopback interface card is done for 1 node, between 2 processes on the same node and the network speed between two nodes. The parameter space here includes TCP protocol stack, UDP with fixed packet/buffer size of 64KB. Multiple threads are parallelly run achieve concurrency.Throughput is measured in Mb/sec and Latency in ms for Network benchmarks.

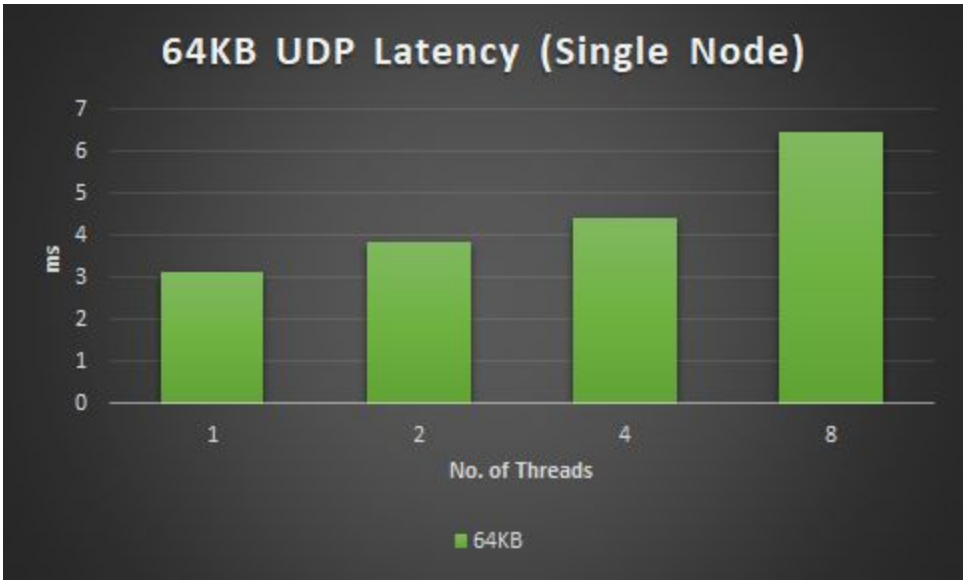
TCP	No. of Threads	64KB
	1	128.2922306
	2	119.6569704
	4	89.6503151
Throughput	8	57.09115108
	1	3.990888596
	2	4.278898239
	4	5.711078644
Latency	8	8.968114853





UDP	No. of Threads	64KB
	1	162.897948
	2	133.682996
	4	115.549295
Throughput	8	79.05041773
	1	3.143072128
	2	3.829956055
	4	4.431009293
Latency	8	6.47687912



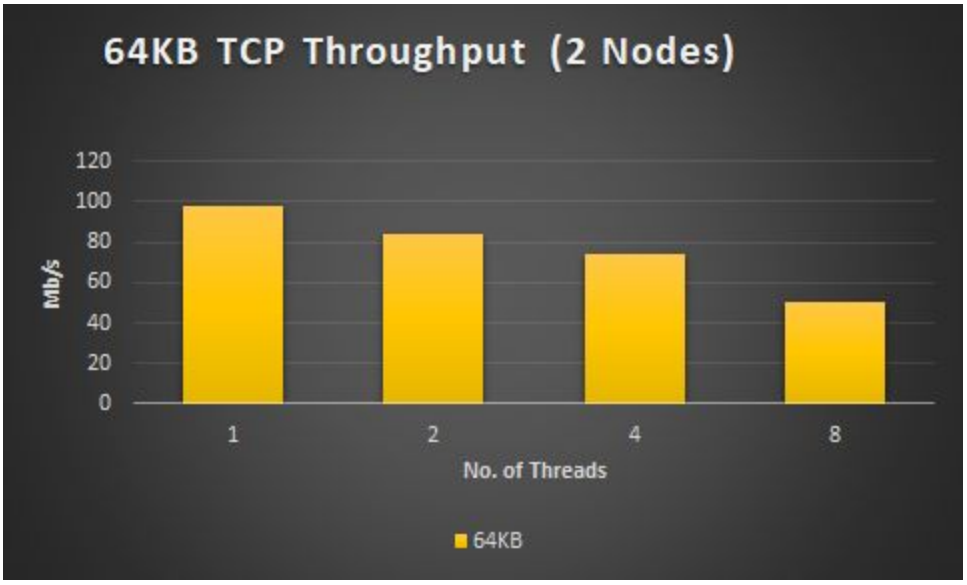


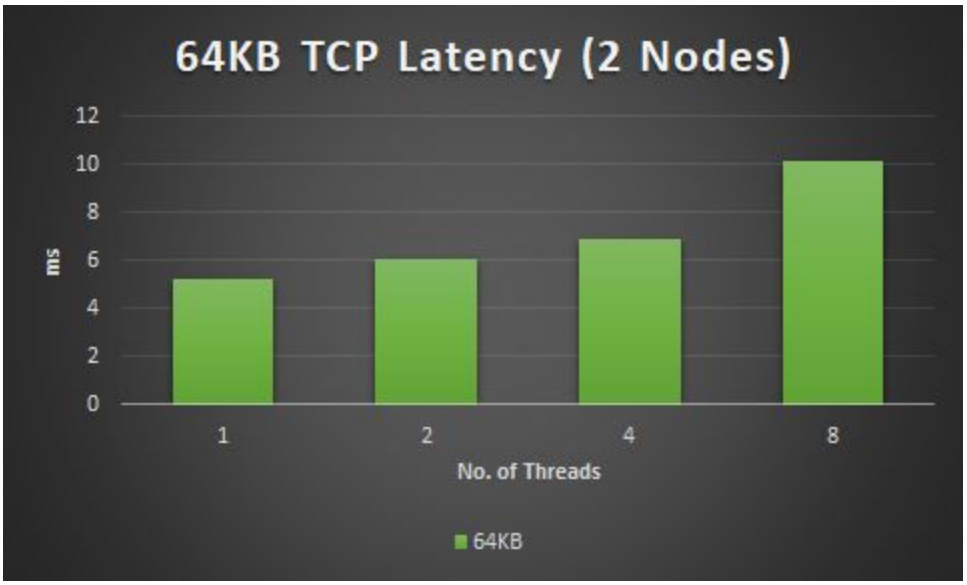
Multi Nodes (2 nodes used for testing):

Throughput is measured in Mb/sec and Latency in ms for Network benchmarks.

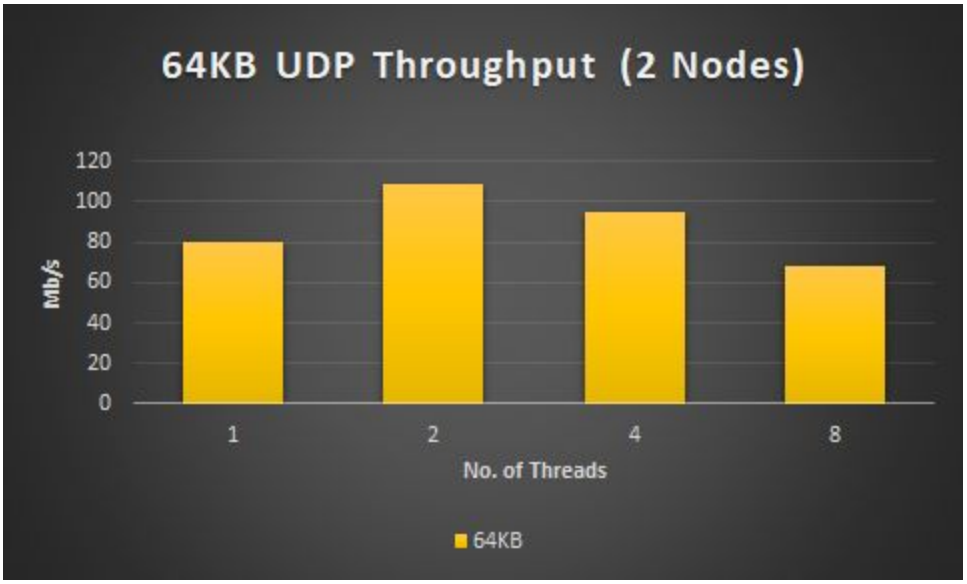
TCP (2 nodes)	No. of Threads	64KB
	1	98.19761525
	2	83.68993172
	4	74.25600443
Throughput	8	50.52308312

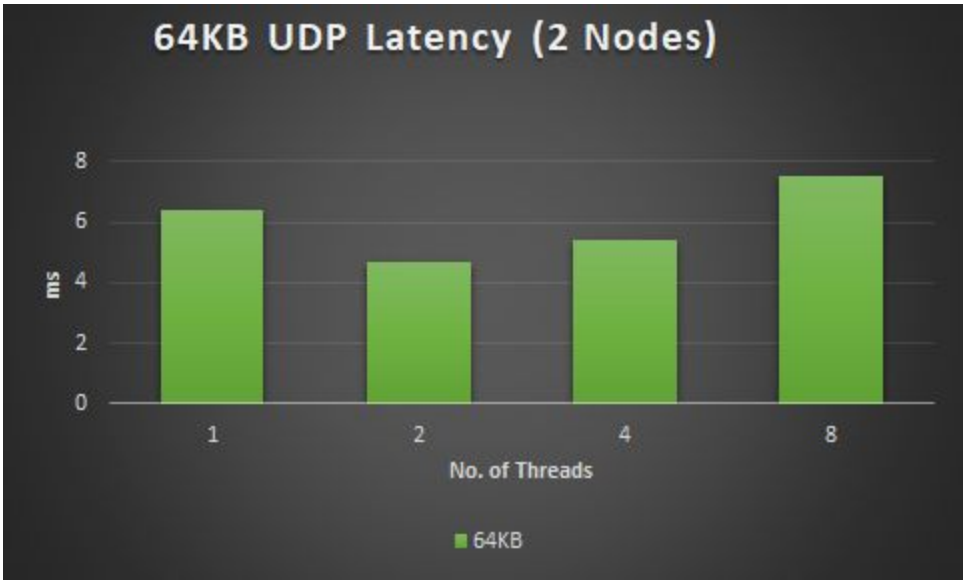
	1	5.213975906
	2	6.11782074
	4	6.895065308
Latency	8	10.1339817





UDP (2 nodes)	No. of Threads	64KB
Throughput	1	80.21079625
	2	109.2866996
	4	94.79908392
	8	68.13947354
Latency	1	6.383180618
	2	4.684925079
	4	5.400896072
	8	7.513999939





Comparing to theoretical memory performance, the UDP is different because there is size limitation for packet, no more than 65536 bytes per packet.

For loopback, the entire process is usually CPU bound so its performance is therefore directly linked to CPU speed plus stack efficiency. The traffic never hits the physical network. However, for two nodes, we have to count the physical network in.

We observe that TCP throughput is lesser than UDPs. It is because that UDP is faster than TCP, and the reason is because it’s nonexistent acknowledge packet (ACK)(connectionless protocol), instead of TCP that acknowledges a set of packets and has to create a handshake first.

And for two nodes, both of our throughout are lesser than the theoretical network speed. It might be caused by the unknow configuration of Chameleon's Hubs/routers and the latency between two machines(nodes).

In the IPerf results, it has given bandwidth and packets which are transferred across two sides. However, for UDP, it is not feasible to send more than 65536 bytes at the same time due to the limitation of UDP. On the other hand, TCP can send more bytes. So we have to consider this when we look into the results.

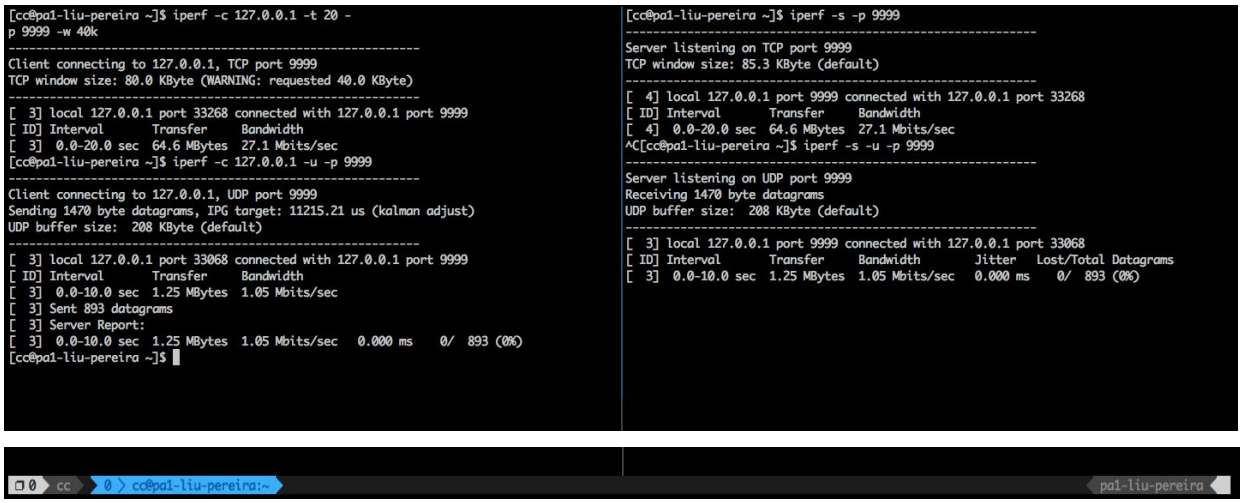
There is difference between the theoretical latency. Our conclusion is that even we already keep only one process(our code for testing) running but for CentOS itself there are still some

background processes running. This will be one of the reasons that affect our results of both IPerf and our code.

IPerf Benchmark results are as below:

Results are stored in Network/IPERF_loopback.png and Network/IPERF_2nodes.txt

LOOPBACK:



2 nodes:

```
TCP:
Server:
[cc@pa1-lp-t ~]$ iperf -s -p 9999
------------------------------------------------------------
Server listening on TCP port 9999
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local 192.168.0.186 port 9999 connected with 192.168.0.178 port 40738
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-20.0 sec  2.33 GBytes  1.00 Gbits/sec

Client:
[cc@pa1-liu-pereira ~]$ iperf -c 192.168.0.186 -t 20 -p 9999 -w 40k
------------------------------------------------------------
Client connecting to 192.168.0.186, TCP port 9999
TCP window size: 80.0 KByte (WARNING: requested 40.0 KByte)
------------------------------------------------------------
[  3] local 192.168.0.178 port 40738 connected with 192.168.0.186 port 9999
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-20.0 sec  2.33 GBytes  1.00 Gbits/sec
```

UDP:

Server:

```
[cc@pa1-lp-t ~]$ iperf -s -u -p 9999
------------------------------------------------------------
Server listening on UDP port 9999
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.186 port 9999 connected with 192.168.0.178 port 58691
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec   0.019 ms    0/  893 (0%)
```

Client:

```
[cc@pa1-liu-pereira ~]$ iperf -c 192.168.0.186 -u -p 9999
------------------------------------------------------------
Client connecting to 192.168.0.186, UDP port 9999
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.0.178 port 58691 connected with 192.168.0.186 port 9999
[ 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   0.019 ms    0/  893 (0%)
```