CPU Performance

- **System Configuration**
  - Memory : RAM 4GB
  - CPU core: 2
  - CPU Threads : 2
  - Disk :40 GB
  - m1-medium
  - CentOS7

CPU benchmarking for Integer and Floating point operations

Number of Thread	GIOPS	GFLOPS
1	10.519813	9.990418
2	19.351904	19.189053
4	19.072120	19.005313
8	19.700419	18.593352

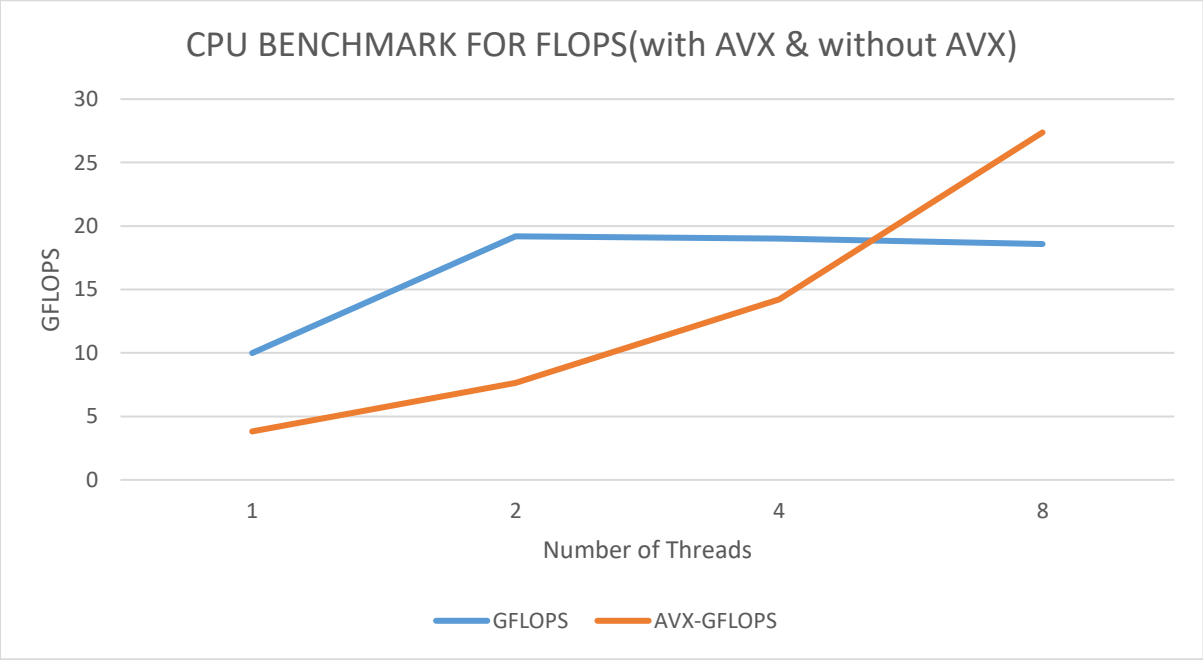
Table 1: Normal Instructions

Number of Thread	GIOPS	GFLOPS
1	3.004218	3.817306
2	5.997491	7.627489
4	11.218634	14.224442
8	19.912195	27.375987

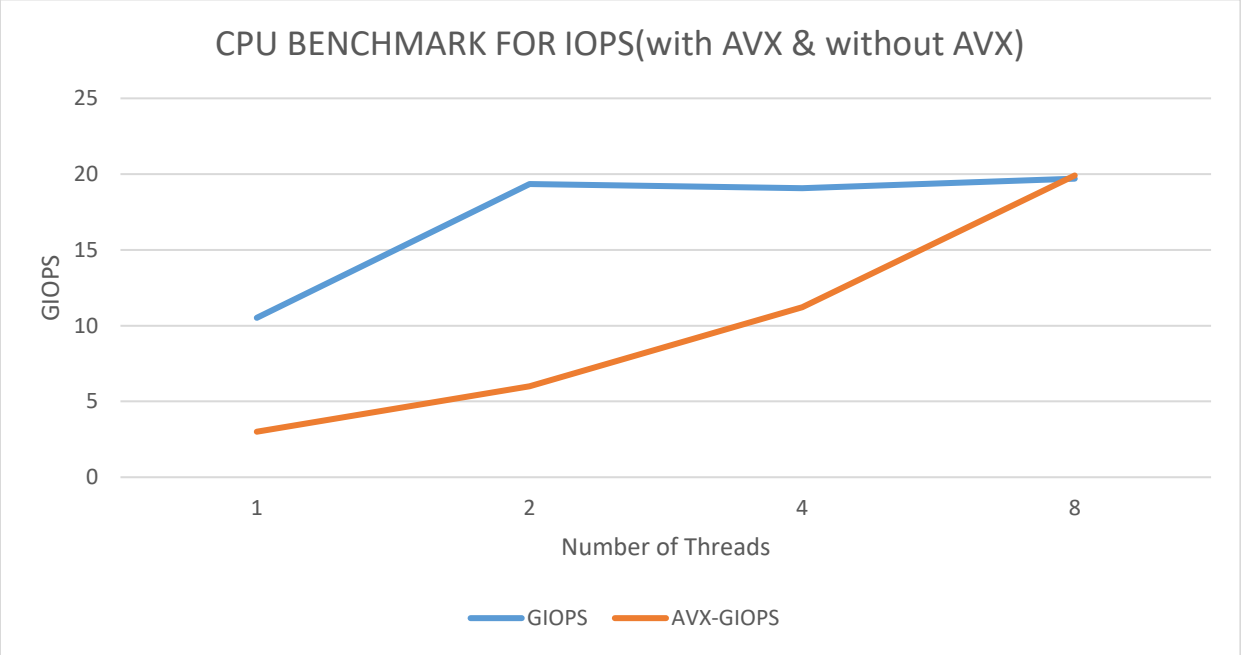
Table 2: AVX Instructions

Analysis:

- The above two tables and the below graph shows the speed of the CPU in terms of Giga Integer Operations(GIOPS) and Giga Floating Point Operations (GFLOPS) for execution of normal instructions and for Advanced Vector Instructions(AVX instructions).
- The experiment is performed with varying number of threads – 1,2,4,8 and the GFLOPS and GIOPS are calculated.
- The Table 1, shows that as the number of threads is increasing for The GIOPS and GFLOPS are also increasing. For the benchmarking with normal instructions, the IOPS and FLOPS increases and then after thread 4, it is constant at 19 GIOPS.



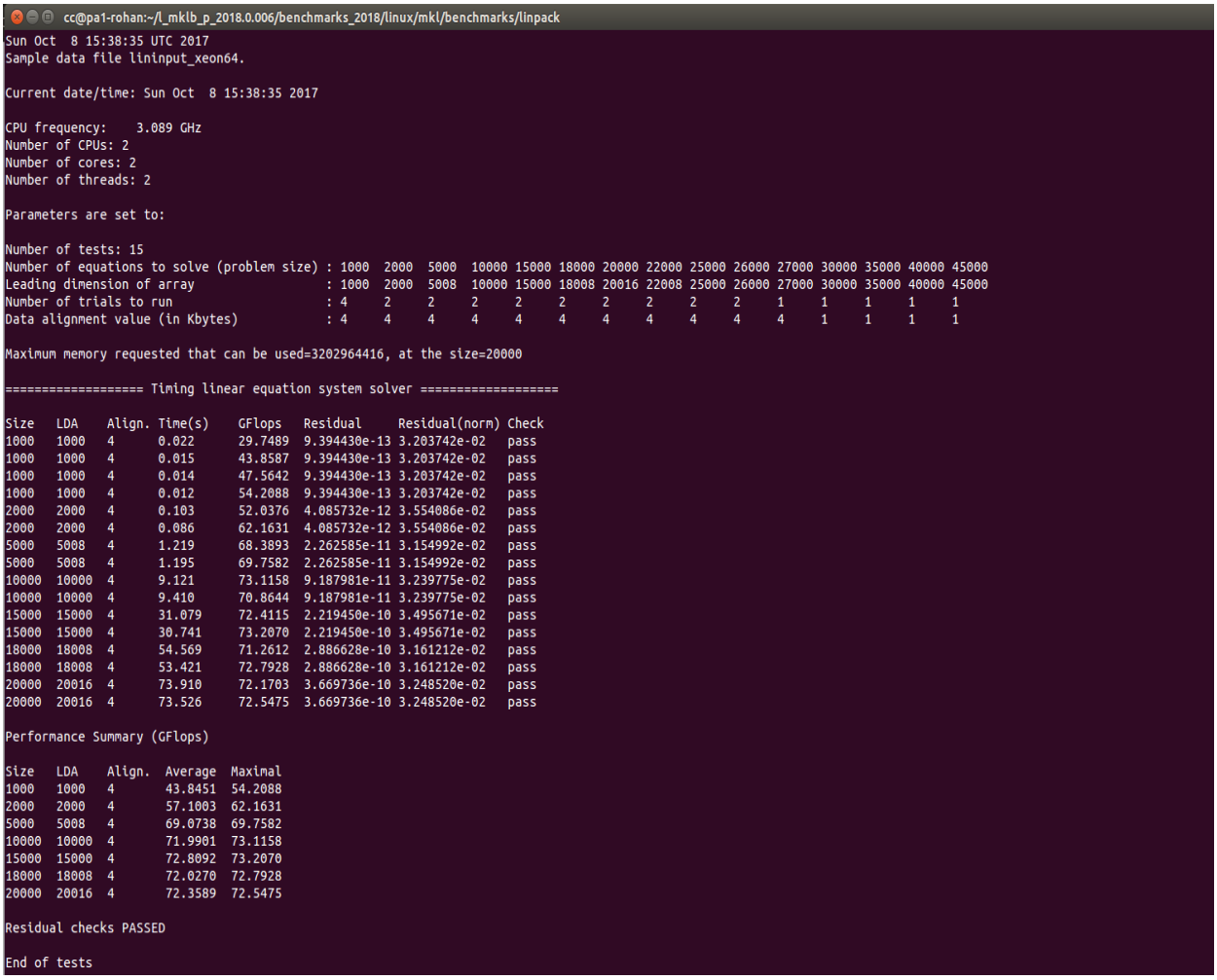
- The Table 2, which is for AVX, we can see the similar results, but the number of GIOPS and GFLOPS are small initially and then rises gradually to 19 as the thread increases.
- The data in both the tables is plotted graphically, GIOPS for normal instructions is compared with GIOPS for AVX Instructions, and similarly for GFLOPS.
- We can see from both the graphs that as the number of thread increases the AVX instructions performs better while normal instructions run at stable speed.



- **Theoretical peak performance**
  = CPU Frequency * Number of Cores * Threads per core * IPC
  = 2.93*2*2*8
  = 93.76 GFLOPS
- Our CPU Benchmark compared to the theoretical performance have an efficiency 20.45% for normal instructions.

LINPACK BENCHMARK

- The linpack benchmark uses basic vector operation and matrix operations.
- The problem size used is 20000. The average performance for this size is 72.35 GFLOPS. The best performance is 72.54 GFLOPS for the problem size of 20000.
- Compared to Theoretical performance, the linpack have an efficiency of 77.16%.
- Compared to our performance we achieved an efficiency of 26.51%, considering GFLOPS for 2 threads as we get best result for 2 threads.
- If we consider the GFLOPS of AVX instructions we achieve an efficiency of 37.75% considering 8 thread GFLOPS.



```
cc@pa1-rohan:~/l_mklb_p_2018.0.006/benchmarks_2018/linux/mkl/benchmarks/linpack
Sun Oct  8 15:38:35 UTC 2017
Sample data file lininput_xeon64.

Current date/time: Sun Oct  8 15:38:35 2017

CPU frequency:    3.089 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     4     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      0.022      29.7489  9.394430e-13 3.203742e-02   pass
1000   1000   4      0.015      43.8587  9.394430e-13 3.203742e-02   pass
1000   1000   4      0.014      47.5642  9.394430e-13 3.203742e-02   pass
1000   1000   4      0.012      54.2088  9.394430e-13 3.203742e-02   pass
2000   2000   4      0.103      52.0376  4.085732e-12 3.554086e-02   pass
2000   2000   4      0.086      62.1631  4.085732e-12 3.554086e-02   pass
5000   5008   4      1.219      68.3893  2.262585e-11 3.154992e-02   pass
5000   5008   4      1.195      69.7582  2.262585e-11 3.154992e-02   pass
10000  10000  4      9.121      73.1158  9.187981e-11 3.239775e-02   pass
10000  10000  4      9.410      70.8644  9.187981e-11 3.239775e-02   pass
15000  15000  4      31.079     72.4115  2.219450e-10 3.495671e-02   pass
15000  15000  4      30.741     73.2070  2.219450e-10 3.495671e-02   pass
18000  18008  4      54.569     71.2612  2.886628e-10 3.161212e-02   pass
18000  18008  4      53.421     72.7928  2.886628e-10 3.161212e-02   pass
20000  20016  4      73.910     72.1703  3.669736e-10 3.248520e-02   pass
20000  20016  4      73.526     72.5475  3.669736e-10 3.248520e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
1000   1000   4       43.8451  54.2088
2000   2000   4       57.1003  62.1631
5000   5008   4       69.0738  69.7582
10000  10000  4       71.9901  73.1158
15000  15000  4       72.8092  73.2070
18000  18008  4       72.0270  72.7928
20000  20016  4       72.3589  72.5475

Residual checks PASSED

End of tests
```

MEMORY PERFORMANCE:

- For performing memory benchmark, we have used strong scaling experiment, where in we have allocated 1GB of memory and the operations are performed.

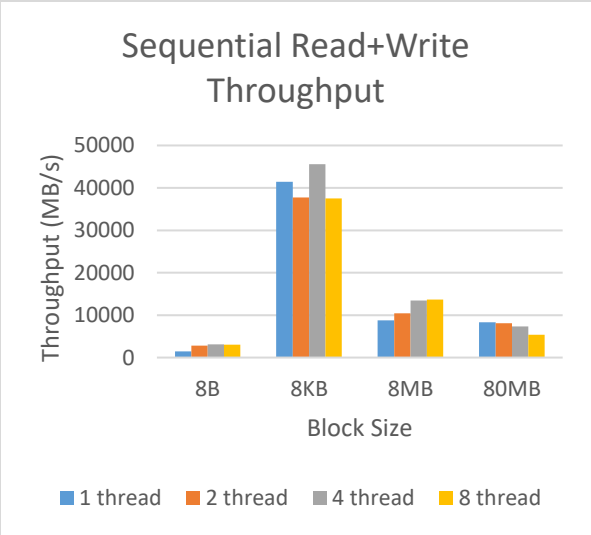
SEQUENTIAL READ+WRITE

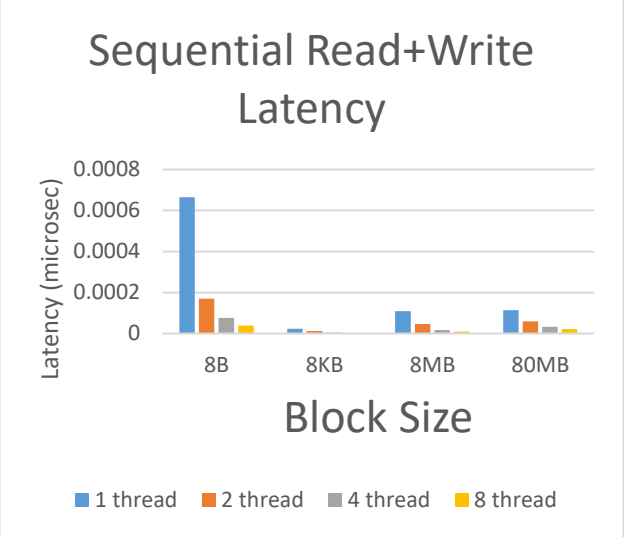
Throughput (MB/s)

Number of Thread	8B	8KB	8MB	80MB
1	1434.84	41433.8184	8785.41	8348.7
2	2802.42	37720.2572	10400	8128.31
4	3110.26	45547.6837	13485.6	7315.88
8	3067.17	37534.937	13653.2	5394.38

Latency(microsec)

Number of Thread	8B	8KB	8MB	80MB
1	0.00066	2.3017E-05	0.00011	0.00011
2	0.00017	1.2641E-05	4.6E-05	5.9E-05
4	7.7E-05	5.2345E-06	1.8E-05	3.3E-05
8	3.9E-05	3.176E-06	8.7E-06	2.2E-05





Analysis

- The throughput is increasing when we run our benchmark from 8B block size to 8KB block but decreases further when we increase the block.
- As we can see that the throughput is increasing and latency is decreasing as the number of thread increases from 1 to 8.
- The throughput spikes up for 8kb block.
- The optimal number of thread we get is 4 as we have best performance for 8B and 8KB block and a marginal less throughput in 8MB block.

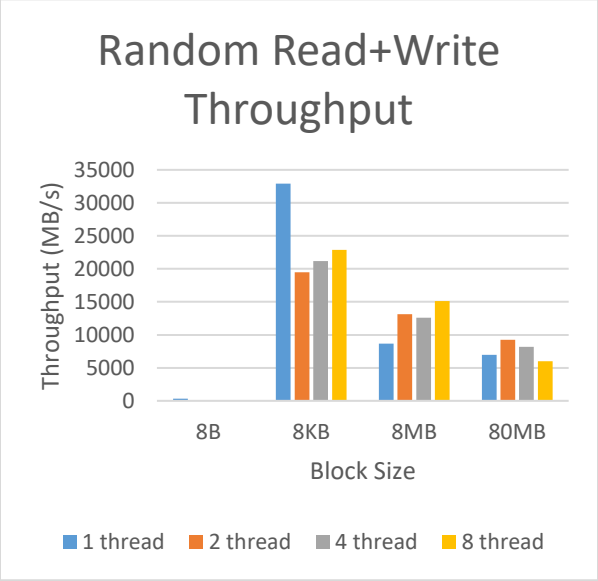
RANDOM READ+WRITE MEMORY

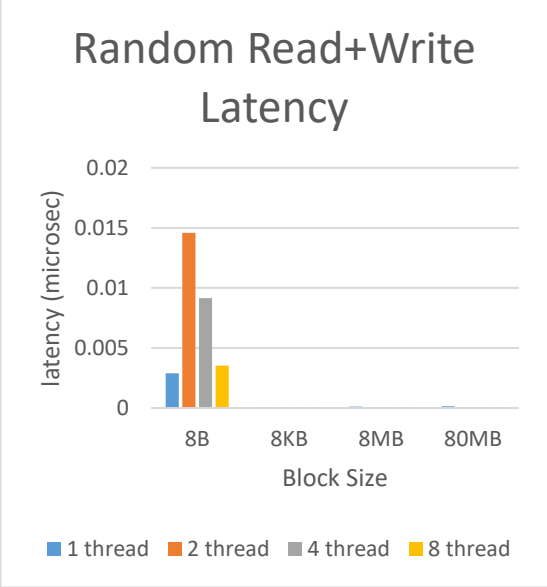
Throughput (MB/s)

Number of Thread	8B	8KB	8MB	80MB
1	330.79	32888.2477	8664.77	6980.13
2	32.6984	19499.5932	13133.9	9254.29
4	26.0631	21173.4085	12587.8	8205.8
8	33.8029	22851.8876	15114.7	5989.49

Latency (micro sec)

Number of Thread	8B	8KB	8MB	80MB
1	0.00288	2.9E-05	0.00011006	0.00014
2	0.01458	2.4E-05	3.6306E-05	5.2E-05
4	0.00915	1.1E-05	1.894E-05	2.9E-05
8	0.00353	5.2E-06	7.887E-06	2E-05





Analysis

- The random read + write operation has a very less throughput and high latency for the block of 8B irrespective of the number of threads.
- The throughput spikes for 8KB block and then decreases for 8 MB and further for 80MB.
- The latency is low for 8KB, 8MB, 80MB as the block size is more and not dependent on the operation.
- The optimal number of thread is difficult to know as for 8KB block 1 thread performs better, for 8MB – number of thread = 8 performs better and for 80 MB thread 2 gives maximum throughput.

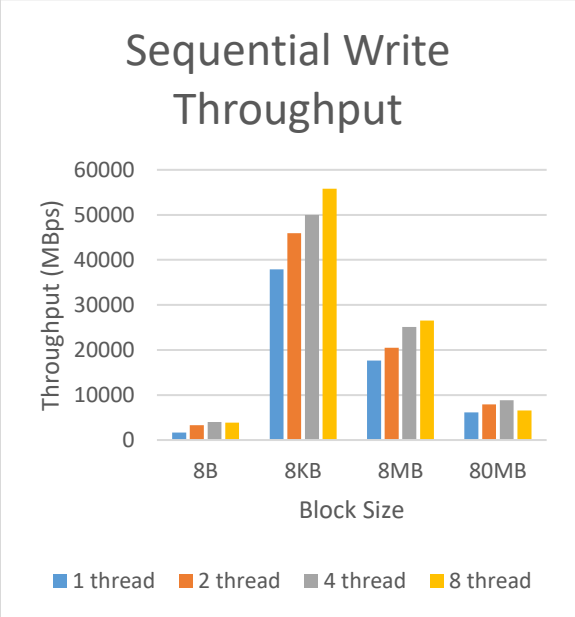
Sequential READ+WRITE MEMORY

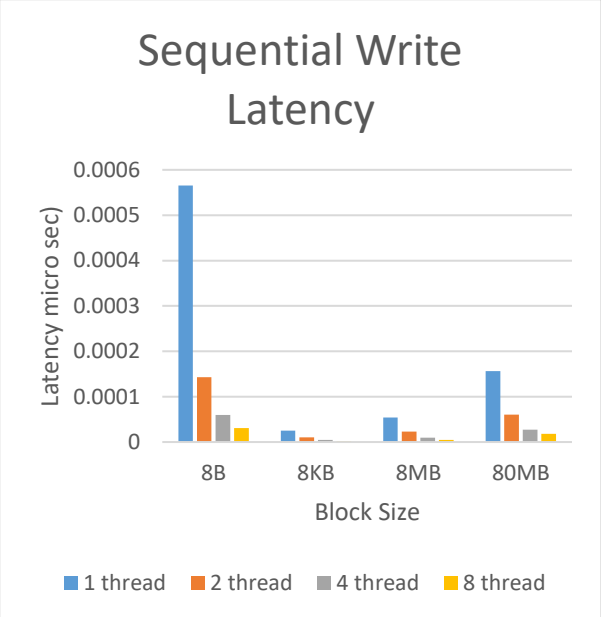
Throughput (MBps)

Number of Thread	8B	8KB	8MB	80MB
1	1687.07	37906	17618.9064	6106.05
2	3329.7	45955.9	20511.5485	7881.59
4	4004.97	50002.4	25121.1586	8817.42
8	3834.49	55800.5	26516.1874	6547.21

Latency(micro sec)

Number of Thread	8B	8KB	8MB	80MB
1	0.00057	2.5E-05	5.4128E-05	0.00016
2	0.00014	1E-05	2.3247E-05	6.1E-05
4	6E-05	4.8E-06	9.4907E-06	2.7E-05
8	3.1E-05	2.1E-06	4.4957E-06	1.8E-05





Analysis

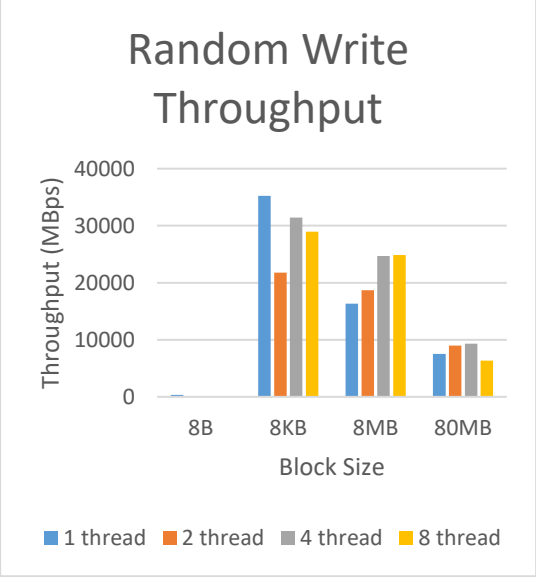
- The sequential write operation performs similar to sequential read + write operation.
- The throughput spikes up for 8KB block and the latency is low for this block.
- The number of thread when 4 and 8 gives good throughput performance
- Also as the number of threads are increasing the throughput is increasing as well.

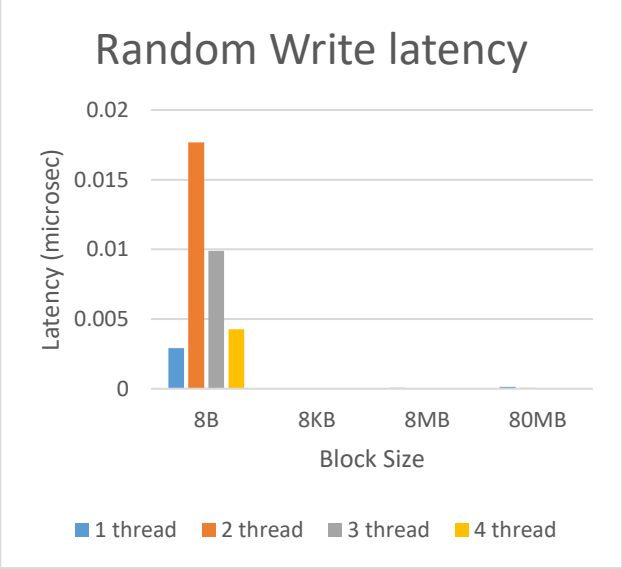
Random Write
Throughput (MBps)

Number of Thread	8B	8KB	8MB	80MB
1	327.779	35221.1	16321.9638	7547.74
2	26.96	21750.5	18678.7085	9009.89
4	24.085	31408.1	24715.8473	9319.14
8	28.01	28925.8	24870.02	6325.59

Latency(microsec)

Number of Thread	8B	8KB	8MB	80MB
1	0.00291	2.9E-05	5.8429E-05	0.00013
2	0.01769	2.2E-05	2.5528E-05	5.3E-05
4	0.0099	7.6E-06	9.6464E-06	2.6E-05
8	0.00426	4.1E-06	4.7933E-06	1.9E-05





Analysis

- The random write operation results are similar to random read+write operation.
- The throughput spikes for 8KB block and then decreases for 8 MB and further for 80MB.
- The latency is low for 8KB, 8MB, 80MB as the block size is more and not dependent on the operation.
- The optimal number of thread is difficult to know as for 8KB block 1 thread performs better, for 8MB and 80MB – number of thread = 4 gives better results.

THEORETICAL BENCHMARK

= Clock Frequency * Number of data transfer per clock * Memory bus interface width * number of interfaces

= 2.93GHz * 2 *64bits *2

= 75.08GHz

STREAM BENCHMARK

```
cc@pa1-rohan:~
[cc@pa1-rohan ~]$ gcc stream.c
[cc@pa1-rohan ~]$ ./a.out
-------------------------------------------------------------
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 29601 microseconds.
   (= 29601 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:           5016.1     0.032914     0.031897     0.034762
Scale:          5001.9     0.032903     0.031988     0.034167
Add:            6565.7     0.037518     0.036554     0.039572
Triad:          6318.0     0.038945     0.037987     0.040959
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
[cc@pa1-rohan ~]$
```

The best rate achieived for performing memory copy function is 5016.1MB/s.

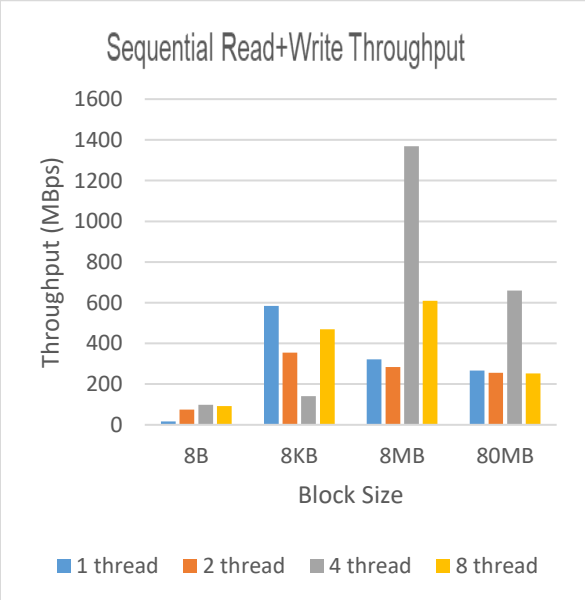
DISK BENCHMARK

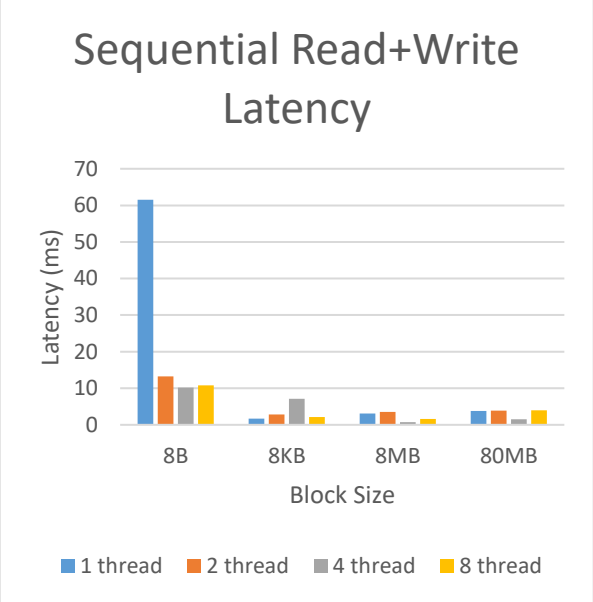
Sequential Read Write Throughput(MBps)

Number of Thread	8B	8KB	8MB	80MB
1	16.241194	584.147248	321.853274	266.554358
2	75.327317	354.749373	283.295392	255.383030
4	98.338451	141.230566	1367.958705	660.344544
8	92.918141	469.685636	609.796716	251.735583

Latency(ms)

Number of Thread	8B	8KB	8MB	80MB
1	61.571826219559	1.711897134781	3.107005834579	3.751579999924
2	13.275396466255	2.818891525269	3.529884457588	3.915686964989
4	10.168962240219	7.080620229244	0.731016218662	1.514360964298
8	10.762161105871	2.129083633423	1.639890760183	3.972422122955





Analysis

- We performed disk operations read+write on the 10GB data.

- We can see that with low block size we are having less throughput and as the block size increases the throughput increases as well.
- As the number of thread increases the throughput also increases irrespective of block size.
- The optimal number of thread is 4 for sequential read+write operation.

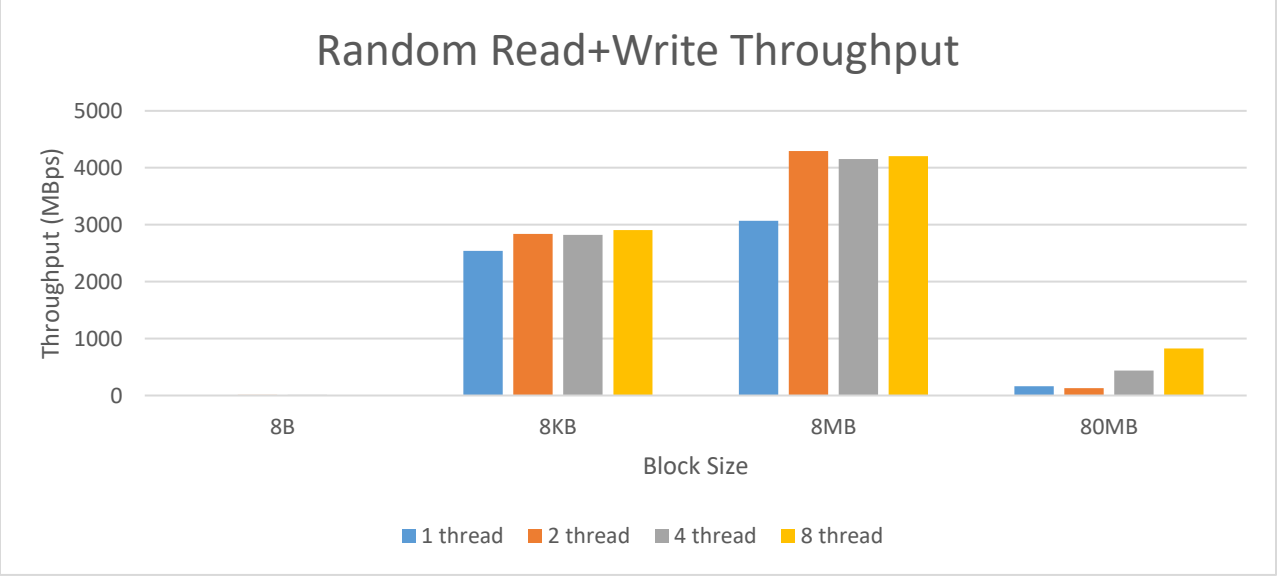
Random read Write

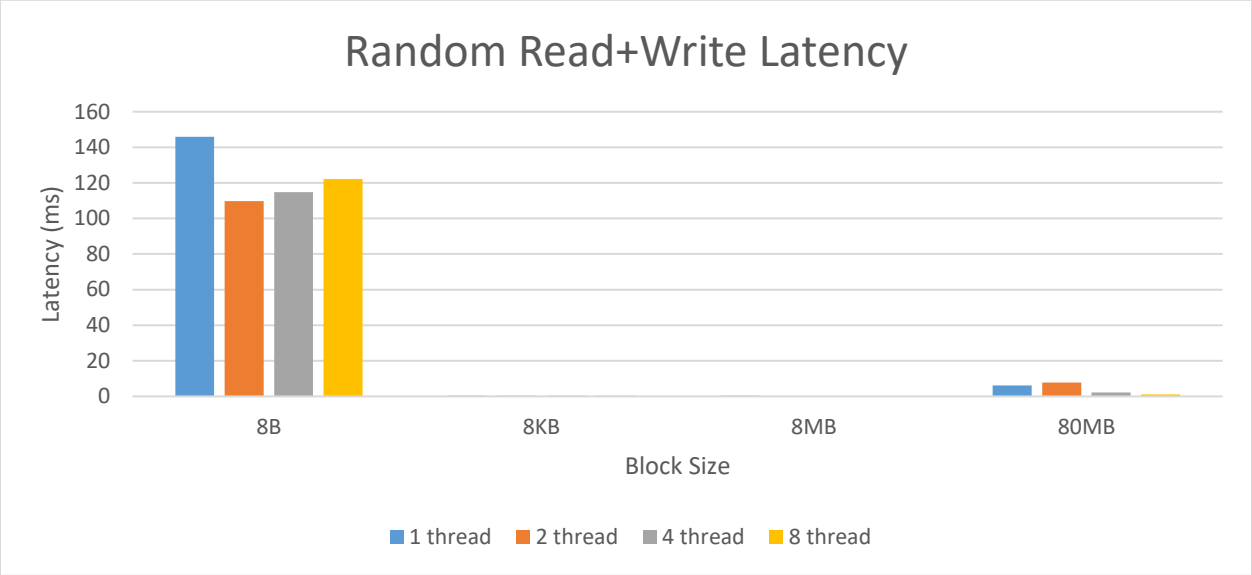
Throughput (MBps)

Number of Thread	8B	8KB	8MB	80MB
1	6.853093	2543.196956	3071.499082	160.652898
2	9.105492	2840.872547	4291.521993	129.973646
4	8.715221	2819.311624	4153.177163	439.406512
8	8.186703	2905.573539	4202.278348	827.351493

Latency(ms)

Number of Thread	8B	8KB	8MB	80MB
1	145.919523954391	0.393205881119	0.325573921204	6.224599838257
2	109.823823451996	0.352004528046	0.233017563820	7.693867444992
4	114.741778731346	0.354696512222	0.240779519081	2.275796949863
8	122.149295389652	0.344166129827	0.237966150045	1.208676129580





Analysis

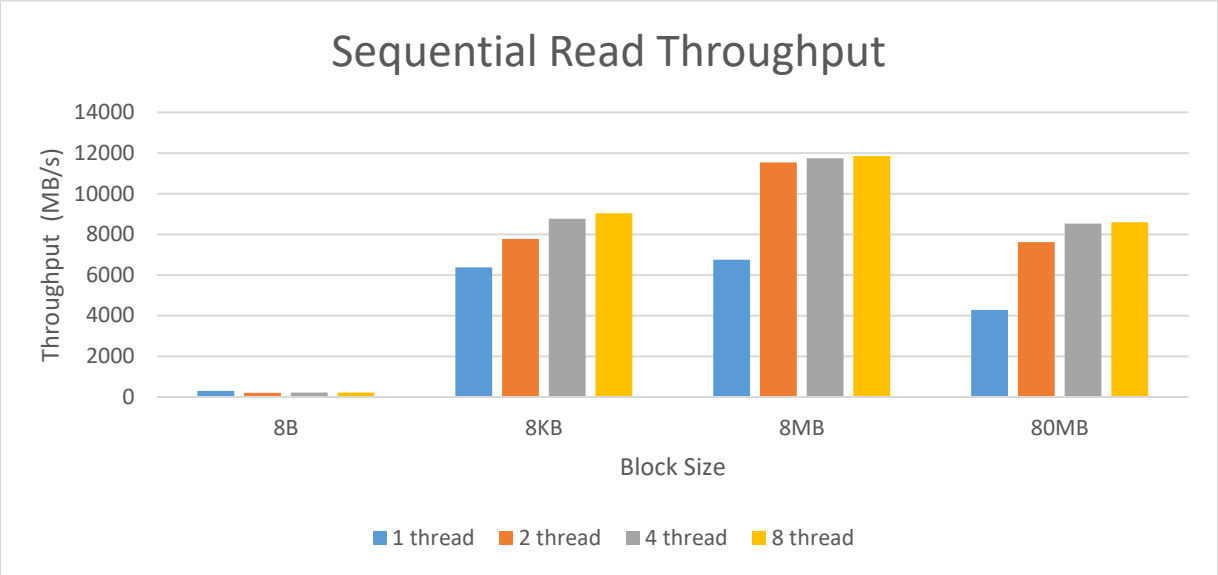
- The random read +write operation results are similar to sequential read + write operations.
- The throughput increases with increase in blocks size and also with increase in thread.
- The latency is high in 8B block access.
- The optimal number of thread is 8 for random read+write operation.

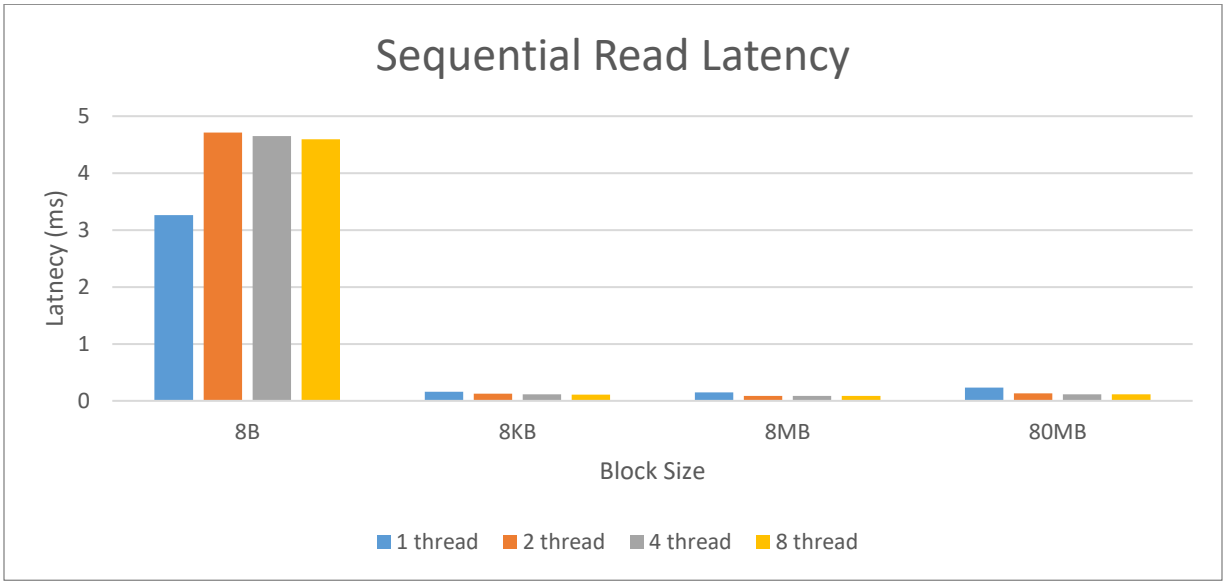
Sequential Read
Throughput (MB/s)

Number of Thread	8B	8KB	8MB	80MB
1	306.777814	6368.853492	6752.840447	4285.849751
2	212.102434	7781.466832	11538.551898	7622.791841
4	214.969523	8766.624864	11740.644065	8523.184827
8	217.693971	9028.695081	11856.773546	8598.664782

Latency (ms)

Number of Thread	8B	8KB	8MB	80MB
1	3.259688138962	0.157014131546	0.148085832596	0.233325958252
2	4.714703083038	0.128510475159	0.086665987968	0.131185531616
4	4.651822209358	0.114068984985	0.085174202919	0.117327034473
8	4.593604475260	0.110757976770	0.084339976311	0.116297125816





Analysis

- The throughput increases and latency decreases with the increase in block size and also the with the increase in thread.
- Only for 80MB block the throughput and latency starts decreasing.

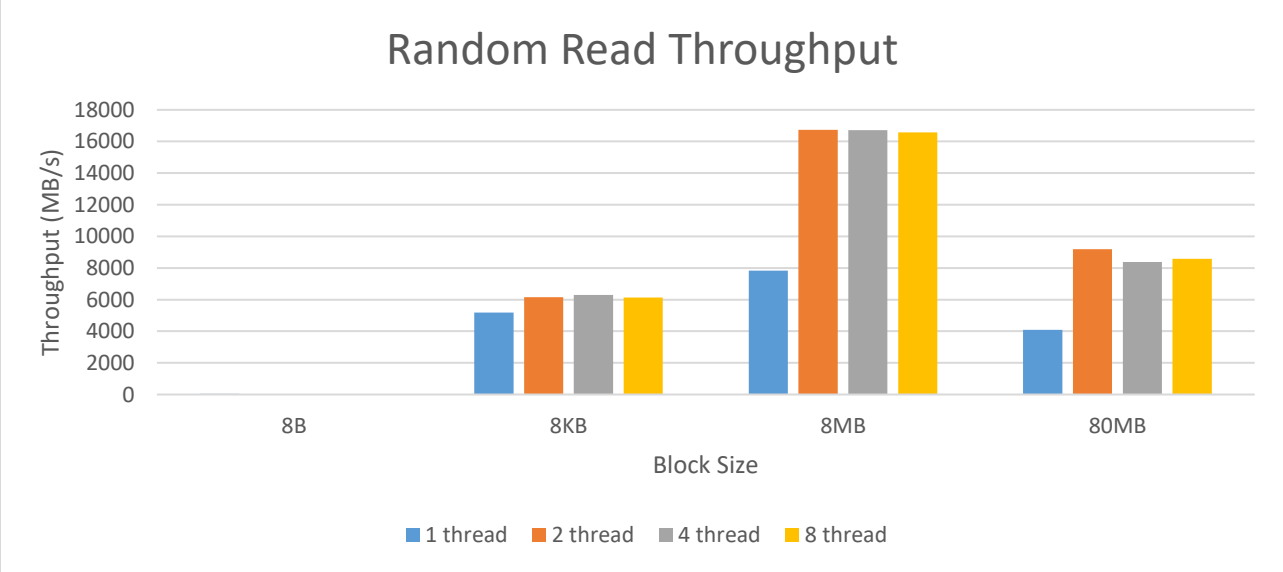
- The optimal number of thread can be 4 or 8 as the sequential read operation have almost equal throughput for all the blocks.

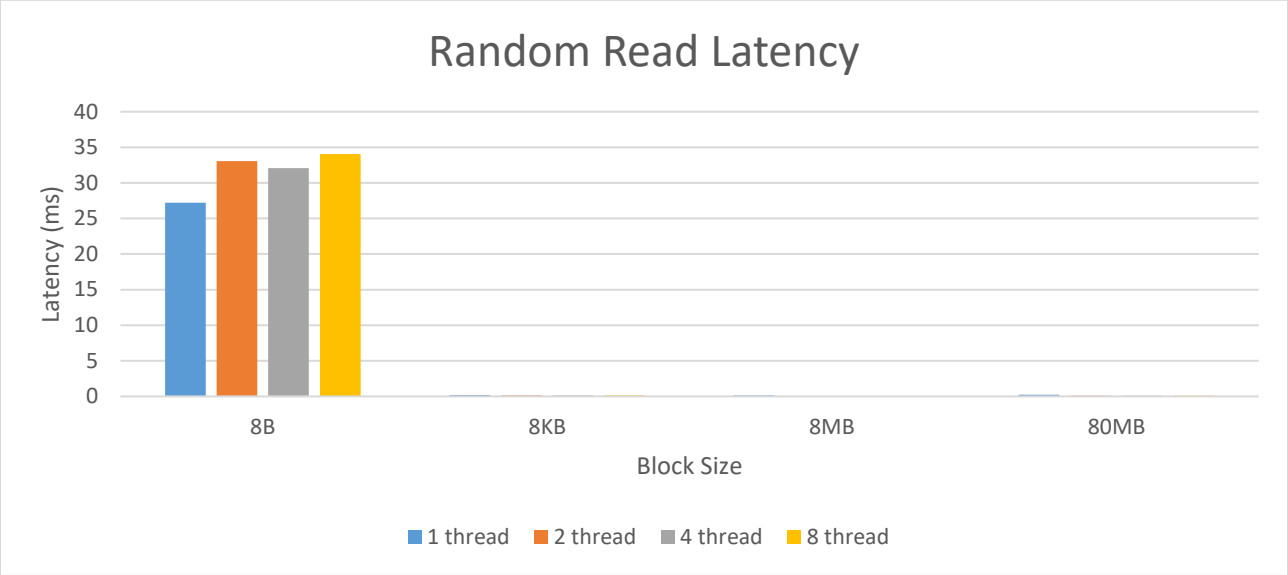
Random Read
Throughput (MB/s)

Number of Thread	8B	8KB	8MB	80MB
1	36.715080	5177.213518	7841.110313	4086.537909
2	30.257610	6154.373485	16726.467896	9182.700381
4	31.194412	6299.125185	16725.634144	8386.079597
8	29.357849	6133.182330	16571.931202	8590.365622

Latency (ms)

Number of Thread	8B	8KB	8MB	80MB
1	27.236765146255	0.193154096603	0.127532958984	0.244705915451
2	33.049536466599	0.162486076355	0.059785485268	0.108900427818
4	32.057023227215	0.158752202988	0.059788465500	0.119245231152
8	34.062440991402	0.163047492504	0.060342997313	0.116409480572





Analysis

- The throughput increases and latency decreases with the increase in block size and also the with the increase in thread.
- Only for 80MB block the throughput and latency starts decreasing.
- The optimal number of thread can be 2 or 4 as the random read operation have almost equal throughput for all the blocks.

- The optimal number of concurrency to obtain best performance is 4 or 8 number of threads.
- Our chameleon instances has SSD disk.

IOZONE BENCHMARK FOR DISK

```
        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: Sat Oct  7 05:14:37 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.
                                                        random  random    bkwd   record   stride
              KB  reclen   write rewrite    read    reread    read   write    read  rewrite     read   fwrite frewrite   fread  freread
              64       4  581273 1310683  2561267  4018152 2772930 2662899 3791156  3541098  4274062  2662899  3057153  587635  9006179
              64       8 1484662 3541098  5389653 15972885 9318832 3203069 5283570  4564786  2892445  1553394  3738358 3057153  4564786
              64      16  735831 3738358  6421025 15972885 9318832 3363612 5389653  1991276  7100397  3541098  4564786 7100397 12310336
              64      32  771797 4018152 15972885  642102512902017 3541098 5860307  4564786  2923952  3541098  2278628 3057153  4564786
              64      64 1599680 4018152  6421025  6421025 4564786 1879725 5735102  1879725  6421025  1330167  1933893 3022727 12902017
             128       4 1508887 1852520  4135958 10779307 8036304 3895854 5847904  4596273  6114306  3785961  3657016 7582312  3468030
             128       8 1622919 1852520  5603747 1420079410567140 3895854 7476717  2511022  3867787  4407601  4557257 4012317  5122535
             128      16 1997245 4717434 16365173 1588107812542043 3199360 8036304  2608629  9287508  4889281  2098744 4135958  5325799
             128      32 1526043 4889281  5545860 1588107812842051 5603747 8036304  5784891  9129573  2326073  5603747 9129573  5325799
             128      64 1727352 2166499 15881078  640613814200794 4934216 3359523  6114306  9977956  1967960  2905056 4135958  5325799
             128     128 1997245 2202044  6114306 15881078 6114306 3895854 7476717  3657016  9129573  5545860  5847904 9977956  5545860
             256       4  886067 2849590  4281170  4350555 2588541 2639446 6936061  5455847  7120034  1694118  4197489 8534922  3879045
             256       8 1312954 4929815 13454450 1416439511091721 5455847 9114515  6936061  4054829  2114473  5022044 4477549 12812277
             256      16 1610276 5455847  8815202  6107548 5217259 2392442 9114515  7314033  7735574  5320671  2242541 4819184 14164395
             256      32  856386 2170027 14953435 1495343513454450 639872010245071  7571923 11695808  5938650  610754811569783 14164395
             256      64 1003679 2539563 16072608 1607260814164395 693606110651598  7791708  3810220  3368013  6761355 4553502  5687020
             256     128 1829808 2533571 14164395  5971678 8534922 7120034 4197489  5320671  4197489  4264168  35577251091721  4652146
```

```
           65536    1024 2144069 3490610  5973554  6249241 6306447 3754352 6284675  8738265  6744509  2982687  2944096 6042892  6352209
           65536    2048 2147738 3530647  6227861  6398639 6463030 3764274 6250520  8625296  6924782  2665927  2769918 6080050  6318624
           65536    4096 2445276 3776635  7683775  7788938 7581624 4067035 7855044  8635865 12656799  3274039  3084917 7619664  7658727
           65536    8192 2490810 3777517  8103758  8209524 7967982 4071311 7743082  8489177 12208219  3503467  3320992 7822408  7865608
           65536   16384 2354781 3514532  6151897  6252226 6175115 3841716 6116171  4696615  6972737  3837425  3742136 6050740  6338147
          131072      64 2471514 3738517  8135032  8180670 8165481 3628612 7929802 10693806  8250280  3727238  3686647 8067464  8068411
          131072     128 2481532 3684596  7733366  7865136 7839676 4073206 7652417 10478963  7770092  3559999  3558248 7934151  7965997
          131072     256 2472403 3758248  7155036  7317899 7346258 4006125 7162027  9435793  7318776  3479757  3474699 7438802  7438399
          131072     512 2491113 3740323  7492841  7565536 7532366 3996805 7639550  8511099  7552440  3502526  3474391 7419426  7432064
          131072    1024 2495851 3803701  7835989  7596165 7528549 4048311 7508704  8531704  7877421  3285757  3264625 7831747  7745242
          131072    2048 2508504 3745343  7673137  7793112 7712861 4066878 7905968  8582584  7759892  3031065  3056426 7851993  7903127
          131072    4096 2504721 3784924  7725108  7849191 7772289 4055837 7859065  8593720  7704753  2987595  3016479 7849527  7867951
          131072    8192 2501473 3778369  7764385  7938046 7773278 4076015 7838782  8599366 12215491  3217847  3134265 7925458  7870767
          131072   16384 2338691 3466679  6152435  6199055 6207244 3773908 6216087  4829460  6806115  3202104  3160482 5964338  6008076
          262144      64 2458537 3549871  8332107  8118203 7934598 3596233 8087331 10678830  8119162  3672201  3680659 8142853  8166501
          262144     128 2478460 3766007  7833633  7928590 7673770 4035840 8023769 10092187  8062957  3526575  3571839 8063252  8121441
          262144     256 2461972 3669738  7279566  7392678 7454073 4020419 7311467  9006572  7133348  3424567  3425997 7221997  7194635
          262144     512 2481201 3734731  7659870  7695199 7651128 3998168 7389300  8657312  7580130  3474499  3541284 7595315  7635984
          262144    1024 2502707 3751027  7560272  7694552 7627985 4039503 7534420  8818682  7816759  3202689  3220683 7747687  7805660
          262144    2048 2517468 3765840  7628250  7790285 7704203 4077206 7416714  8608105  7660350  3020895  2992000 7627562  7575795
          262144    4096 2503665 3844879  7606507  7701505 7544346 4084961 7885715  8510032  7968121  2942665  2967516 7701073  7899993
          262144    8192 2520371 3669175  7586302  7804940 7749544 4094530 7636462  8700952  7763278  3043472  2981178 7658536  7598150
          262144   16384 2334024 3483868  6080400  5994476 5968671 3734185 6194612  5119894  6868693  3000287  2948979 6006002  6272571
          524288      64 2411429 3729143  7990352  8132518 8135045 3572680 8071178 10825917  8117568  3699077  3683142 8321223  8387398
          524288     128 2509421 3855880  8009181  8098733 7903764 4121990 7733106 10509734  7776560  3526354  3537734 7947727  7967654
          524288     256 2461597 3743280  7315086  7054373 7178097 3935831 7061486  8937454  6970431  3425644  3425980 7243641  7246362
          524288     512 2465195 3724135  7213535  7438309 7540564 3956715 7332501  8641791  7507202  3557053  3477421 7632322  7659651
          524288    1024 2503787 3800344  7782092  7545247
```

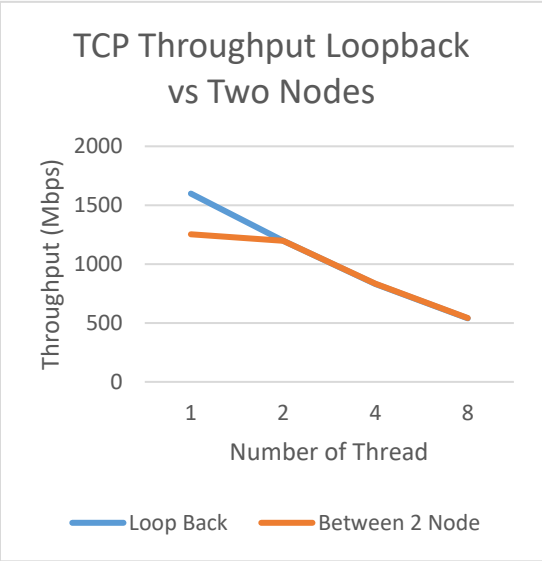
NETWORK BENCHMARK

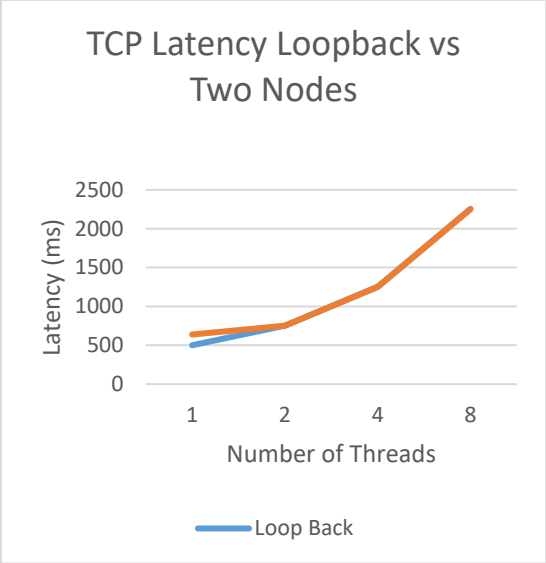
TCP - Loopback

Number of Thread	Throughput(Mbps)	Latency(ms)
1	1598.401598	500.499999
2	1196.54189	751.492123
4	830.57617	1252.8129
8	539.56229	2255.5287

TCP – Two Node

Number of Thread	Throughput(Mbps)	Latency(ms)
1	1252.936	638.5
2	1196.5793	751.597
4	834.259	1250.25
8	542.125	2251.25





Analysis

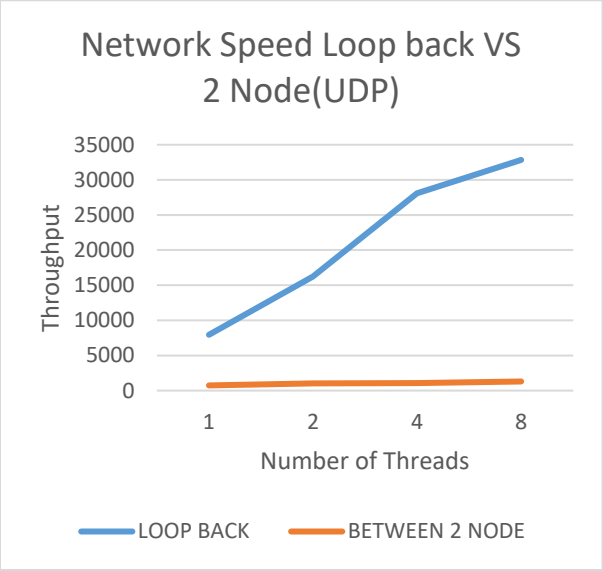
- For TCP we are sending 100 MB file from client to server and back to the client.
- As we can see that as the number of threads are increasing the throughput is decreasing for both loopback and two nodes.

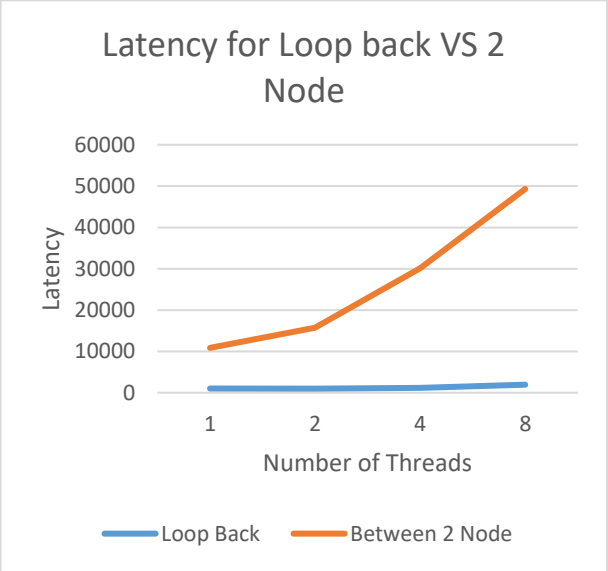
UDP - Loop Back Interface

Number of Thread	Throughput(Mbps)	Latency(ms)
1	7944.38927507448	1006.9999
2	16235.413495687468	985.5
4	28070.175438596492	1140.0
8	32837.35248845562	1949.0

UDP – Two Nodes (192.168.0.195) and (192.168.0.216)

Number of Thread	Throughput(Mbps)	Latency(ms)
1	736.750011511719	10858.5
2	1016.8091258619046	15735.5
4	1061.7296239154598	30139.5
8	1297.9243350672791	49309.5



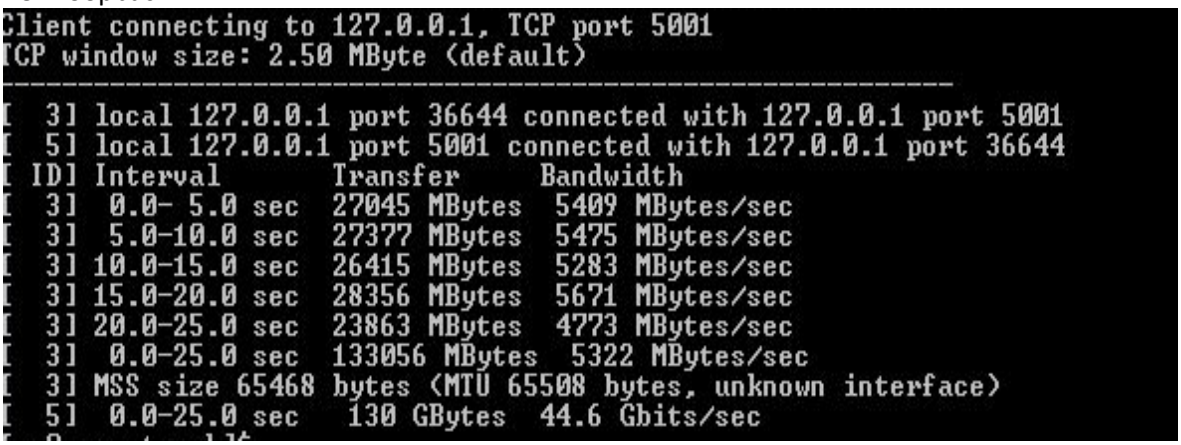


Analysis

- We performed UDP for 1GB data transfer from client to server and back to client with block size of 64KB.
- For loopback, as the number of threads are increasing the network speed is also increasing.
- While for two nodes the network speed is almost stable over all the threads.
- Same thing we can observe for latency. As the throughput is increasing in loopback, the latency is almost stable across all the threads.
- And as the throughput is stable in two node, the latency increases across all the threads.

IPERF BENCHMARK

TCP Loopback



UDP Loopback