CS553 PRORAMMING ASSIGNMENT #1

The benchmark document contains the result of each benchmark with the average and standard deviation. In addition I have run benchmark tools on ec2 t2.micro instance and compare the theoretical and benchmark tool results with output results of benchmark programs.

1) System Environments

I have used following system configuration for all benchmark results.
I performed evaluation on :
OS : Ubuntu 14.04 LTS/Amazon Linux AMI 2015.09.1 (HVM)
Processor : Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz/ Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz

2) Experiments

CPU Benchmark : I
Measure the processor speed, in terms of floating point operations per second (Giga FLOPS) and integer operations per second (Giga IOPS) at concurrency level(1,2 and 4 threads)

Disk Benchmark :
Measured the throughputs (sequential read, sequential write, random read and random write) and latency(random read, random write) for block size(1B/1KB/1MB) and the number of threads(1 and 2).

Network Benchmark :
Measured throughput and latency for both TCP and UDP protocols for packet size(1b/1Kb/64Kb) and the number of threads(1 and 2).

3) Future improvements and extensions

I have evaluated results for specific number of block and packet size for disk and network. Even I have evaluated results for 1,2 and 4 thread. We can extend benchmark program to evaluate more accurate results by passing different and more combinations of arguments.

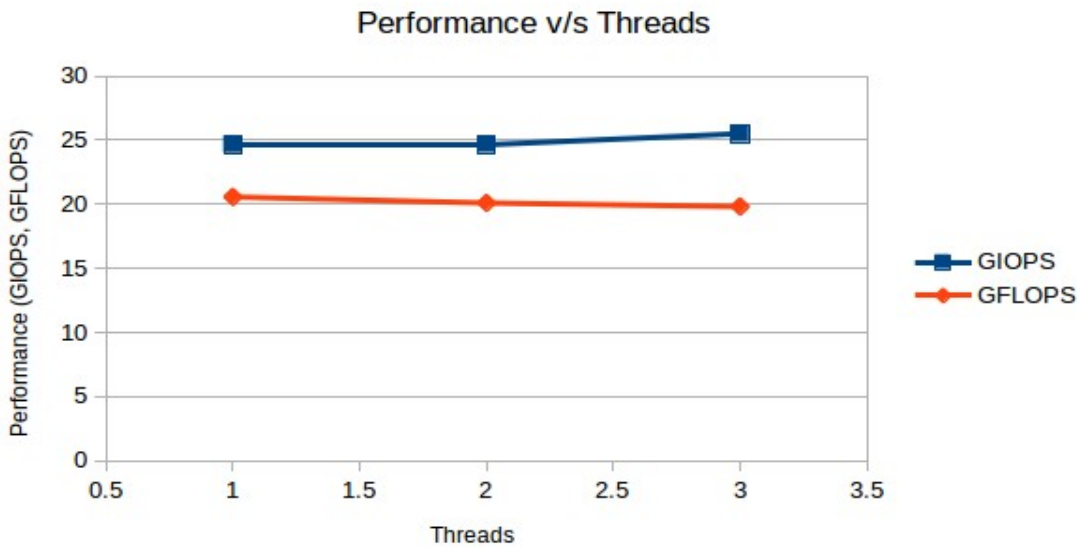
We should extent this utility by applying different algorithms to evaluate benchmark for different components system.

3) Performance Results

1. CPU Benchmark :

a) Processor speed in terms of,

Floating point operation : 20.58 GFLOPS
Integer point operation : 24.54 GIOPS



b) The processor speed at varying level of concurrency

- Speed of processor in terms of integer point operation

	Sample-1	Sample-2	Sample-3	Average	Stdev
Thread-1	25.07	24.89	24.03	24.66	0.56
Thread-2	23.44	25.41	25.20	24.68	1.08
Thread-4	25.37	25.42	25.73	24.50	0.20

- Speed of processor in terms of integer point operation

	Sample-1	Sample-2	Sample-3	Average	Stdev
Thread-1	20.73	20.09	20.92	20.58	0.44
Thread-2	20.65	18.71	20.97	20.11	1.22
Thread-4	20.62	19.57	19.31	20.83	0.69

c) Theoretical peak performance of CPU in terms of flops :

- Performance in GFlops
  = (CPU speed in GHz) x (number of CPU cores) x (CPU instruction per cycle) x (number of CPUs per node)
  = 2.40 * 1 * 16 * 1 = 38.4 GFlops

d) The efficiency what I achieved compared to theoretical performance

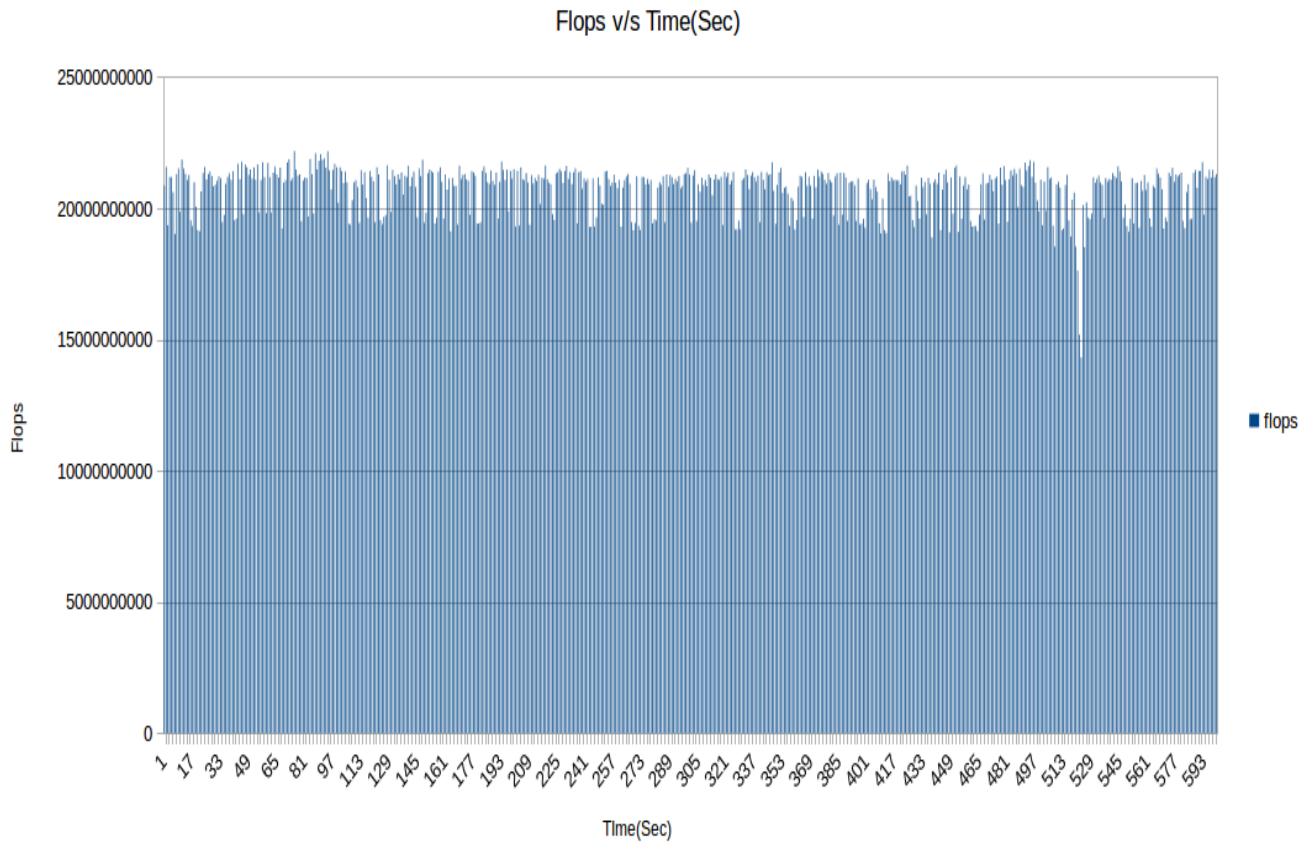
- Efficiency =(CPU benchmark program performance/Theoretical performance) * 100
  = (20.58/38.4) *100
  = 53.59 %

I have achieved 53.59 % efficiency compared to theoretical performance.

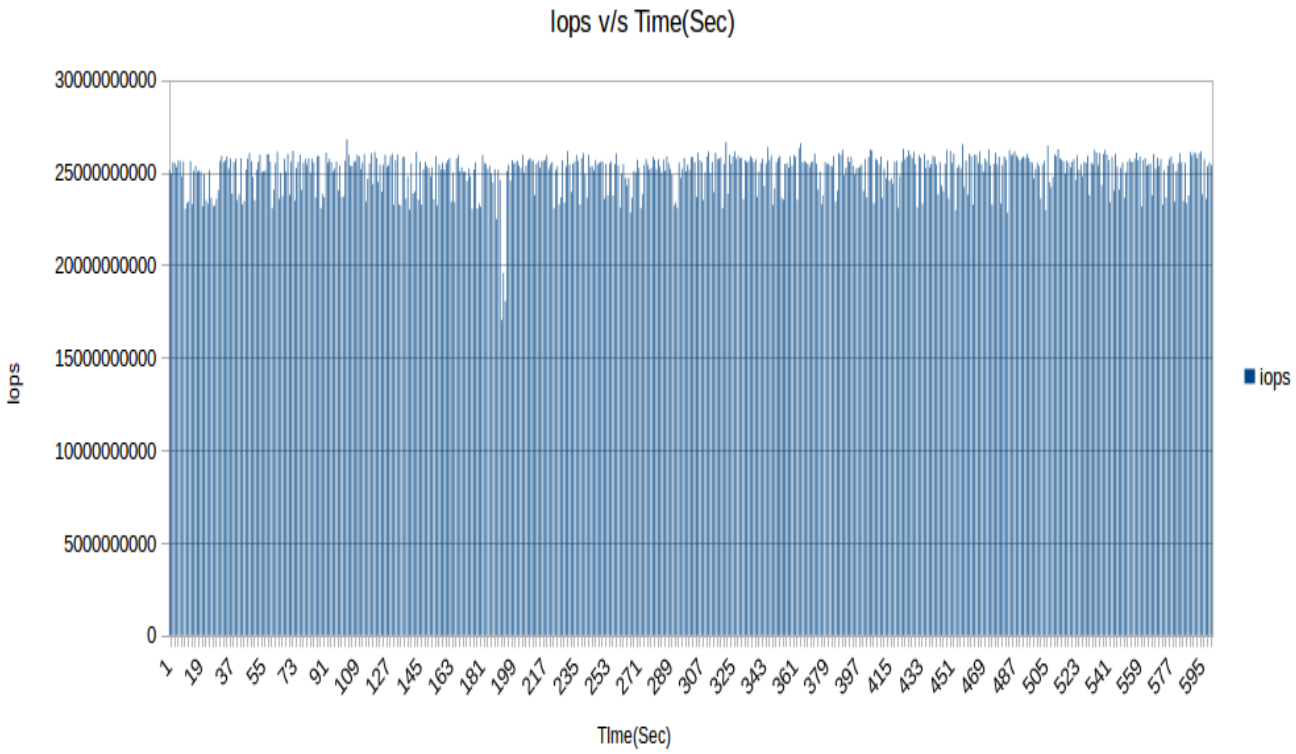
e) Benchmark on

I have performed 600 samples for CPU benchmark results and plotted data for floating point and integer point operation. The plotted data is as below :

- Floating point instruction for 4 thread for 10 minutes



- Integer point instruction for 4 thread with 10 minutes



f) Linpack performance :

- Efficiency = (Linpack performance/Theoretical performance)*100
  = (30.47/38.4) **= 79** %

ec2-user@ip-10-0-0-59:~/l_mklb_p_11.3.1.002/benchmarks_11.3.1/linux/mkl/benchmarks/linpack ec2-user@ip-10-0-1-216:~

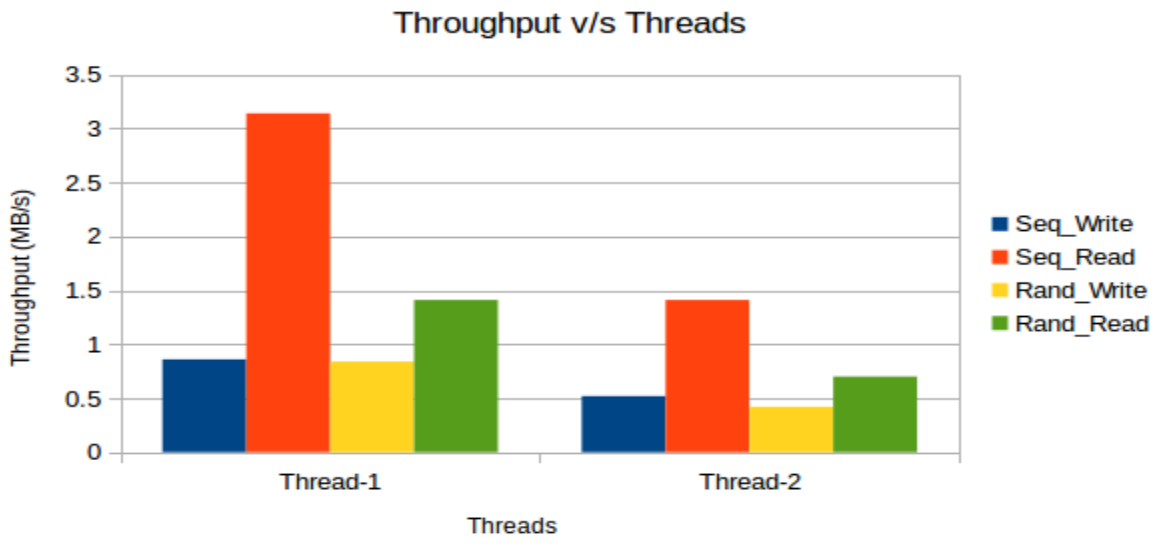
```
-rwxr-xr-x 1 ec2-user ec2-user 1.4M Feb  9 05:09 xlinpack_mic
-rw-r--r-- 1 ec2-user ec2-user  592 Feb  9 05:09 lininput_xeon64_ao
-rw-r--r-- 1 ec2-user ec2-user  486 Feb  9 05:09 lininput_xeon32
-rw-r--r-- 1 ec2-user ec2-user  13K Feb  9 05:09 xhelp.lpk
-rw-rw-r-- 1 ec2-user ec2-user 2.1K Feb  9 17:35 lin_xeon64.txt
[ec2-user@ip-10-0-0-59 linpack]$ ./runme_xeon64
This is a SAMPLE run script for SMP LINPACK. Change it to reflect
the correct number of CPUs/threads, problem input files, etc..
Frl Feb 12 02:13:22 UTC 2016
Intel(R) Optimized LINPACK Benchmark data

Current date/time: Frl Feb 12 02:13:22 2016

CPU frequency:    2.692 GHz
Number of CPUs: 1
Number of cores: 1
Number of threads: 1

Parameters are set to:

Number of tests: 15
Number of equations to solve (problem size) : 1000  2000  5000  10000 15000 18000 20000 22000 25000 26000 27000 30000 35000 40000 45000
Leading dimension of array                  : 1000  2000  5008  10000 15000 18008 20016 22008 25000 26000 27000 30000 35000 40000 45000
Number of trials to run                     : 4     2     2     2     2     2     2     2     2     2     1     1     1     1     1
Data alignment value (in Kbytes)            : 4     4     4     4     4     4     4     4     4     4     1     1     1     1     1

Maximum memory requested that can be used=800204096, at the size=10000

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

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
1000   1000   4      0.040      16.7900  9.632295e-13 3.284860e-02   pass
1000   1000   4      0.026      26.0798  9.632295e-13 3.284860e-02   pass
1000   1000   4      0.025      26.3456  9.632295e-13 3.284860e-02   pass
1000   1000   4      0.025      26.4005  9.632295e-13 3.284860e-02   pass
2000   2000   4      0.191      27.9430  4.746648e-12 4.129002e-02   pass
2000   2000   4      0.192      27.7587  4.746648e-12 4.129002e-02   pass
5000   5008   4      2.502      33.3222  2.651185e-11 3.696863e-02   pass
5000   5008   4      2.479      33.6322  2.651185e-11 3.696863e-02   pass
10000  10000  4      18.460     36.1253  9.014595e-11 3.178637e-02   pass
10000  10000  4      18.817     35.4404  9.014595e-11 3.178637e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
1000   1000   4       23.9040  26.4005
2000   2000   4       27.8509  27.9430
5000   5008   4       33.4772  33.6322
10000  10000  4       35.7828  36.1253

Residual checks PASSED

End of tests

Done: Frl Feb 12 02:14:12 UTC 2016
[ec2-user@ip-10-0-0-59 linpack]$
```

2. DISK BENCHMARK

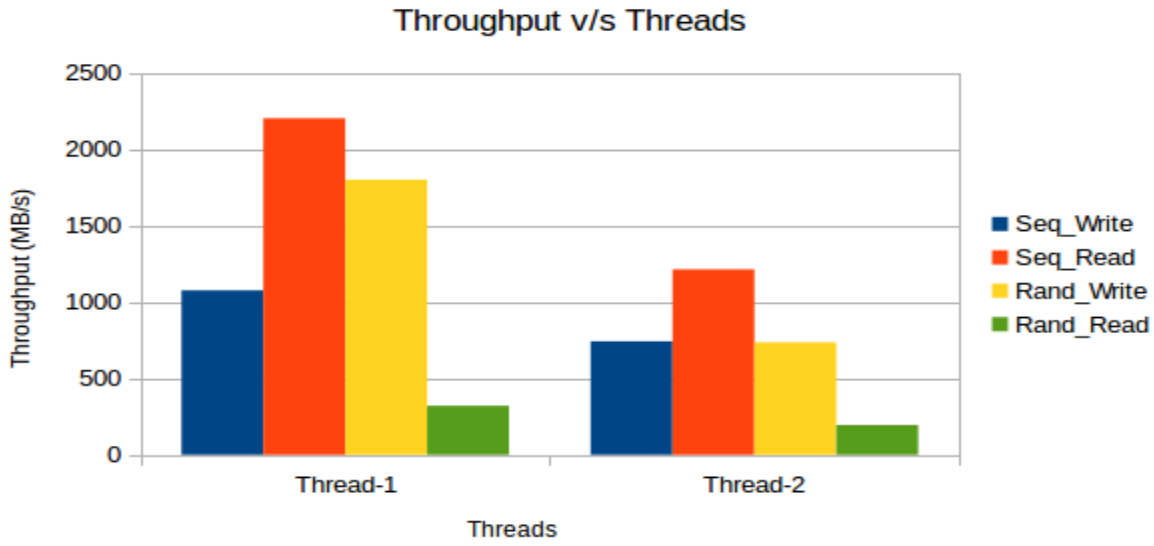
For single and multi threads, I have computed benchmark for different block size (1 Byte, 1 KB, 1 MB) using different file (Sequential Write, Sequential Read, Random Write, Random Read) access methods.

I have performed 1000 iteration for 1Byte and 1KB block size and 100 iterations for 1MB block to evaluate disk benchmarks.

Throughput for 1-Byte

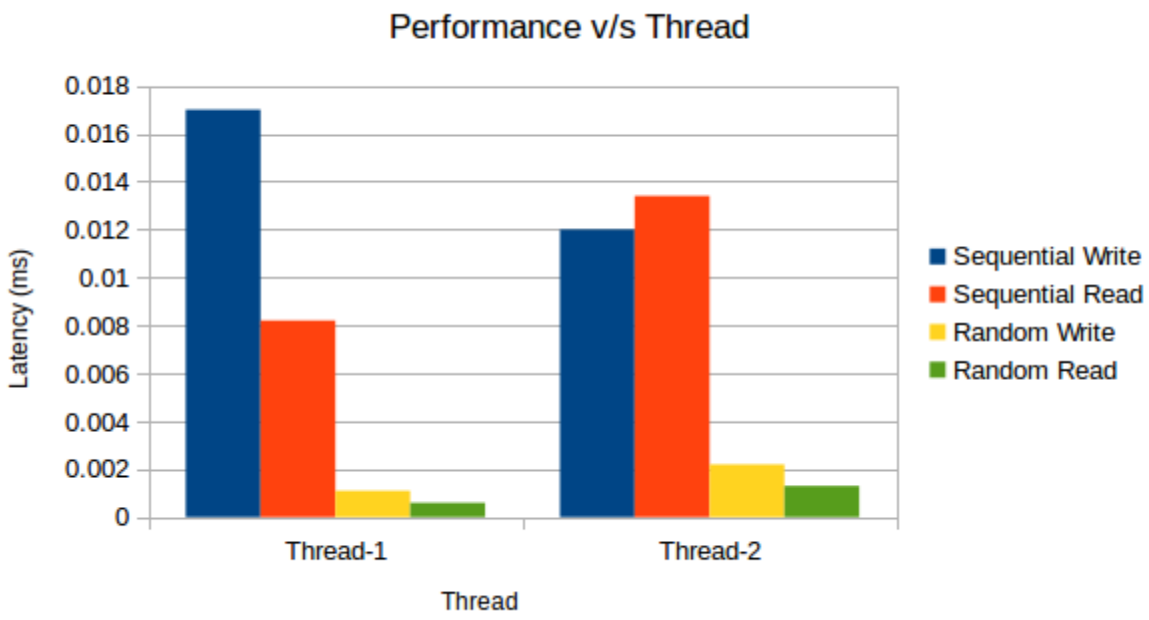


Throughput for 1-MB

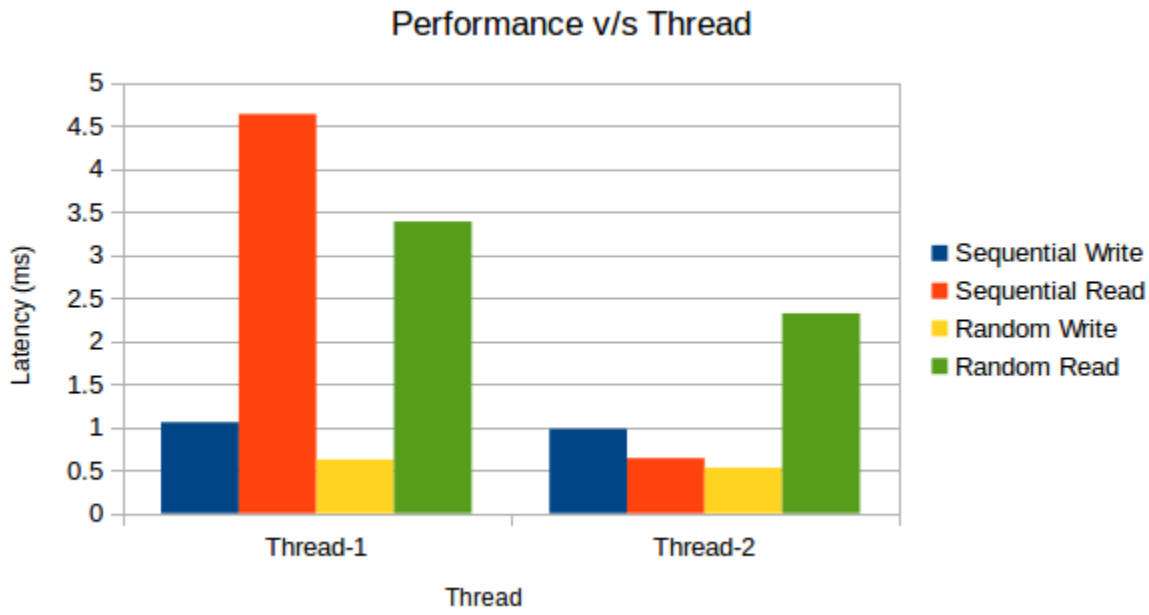


- The above graph represents the relation between no of threads and performance(throughput) in (Mbps).
- For ec2-t2.micro instance, we have thread per core is equals to 1. Hence, when number of threads are being increased, throughput of block will decrease.

Latency for 1-Byte



Latency for 1-MB



The above graph represents the relation between no of threads and performance(latency) in (ms). It shows when

number of threads are being increased, latency of blocks will increase in case of random file read and write.

Disk benchmark evaluated results for 1,2 Thread and 1Byte, 1KB and 1 MB are as below:

Performance in throughput (MB/s) for 1-Thread, 1 Byte

	Average Throughput	Stdev Throughput	Average Latency	Stdev Latency
Seq_Write	0.86	0.007	0.0314	0.0022
Seq_Read	3.14	0.014	0.0075	0.0004
Rand_Write	0.84	0.003	0.0011	0.0000004
Rand_Read	1.41	0.010	0.0006	0.00000001

Performance in throughput (MB/s) for 1-Thread, 1 KB

	Average Throughput	Stdev Throughput	Average Latency	Stdev Latency
Seq_Write	516.667	13.793	0.032	0.0067
Seq_Read	1387.892	59.472	0.006	0.0005
Rand_Write	58.203	0.879	0.016	0.0002
Rand_Read	50.910	0.373	0.019	0.0001

Performance in throughput (MB/s) for 1-Thread, 1 MB

	Average Throughput	Stdev Throughput	Average Latency	Stdev Latency
Seq_Write	1074.882	278.172	0.999	0.037
Seq_Read	2201.695	46.450	0.656	0.027
Rand_Write	1799.277	204.942	0.560	0.066
Rand_Read	321.004	4.771	3.115	0.046

Performance in throughput (MB/s) for 2-Thread, 1 Byte

	Average Throughput	Stdev Throughput	Average Latency	Stdev Latency
Seq_Write	0.518	0.0008	0.019	0.004
Seq_Read	1.413	0.1434	0.005	0.0004
Rand_Write	0.423	0.0074	0.002	0.00003
Rand_Read	0.698	0.0313	0.001	0.00005

Performance in throughput (MB/s) for 2-Thread, 1 KB

	Average Throughput	Stdev Throughput	Average Latency	Stdev Latency
Seq_Write	274.392	3.977	0.018	0.001
Seq_Read	485.421	17.491	0.004	0.0001
Rand_Write	39.866	0.336	0.024	0.0001
Rand_Read	20.475	3.525	0.048	0.009

Performance in throughput (MB/s) for 2-Thread, 1 KB

	Average Throughput	Stdev Throughput	Average Latency	Stdev Latency
Seq_Write	742.089	47.015	0.847	0.041
Seq_Read	1212.519	30.947	0.573	0.004
Rand_Write	735.130	124.896	1.445	0.303
Rand_Read	194.185	17.513	5.230	0.446

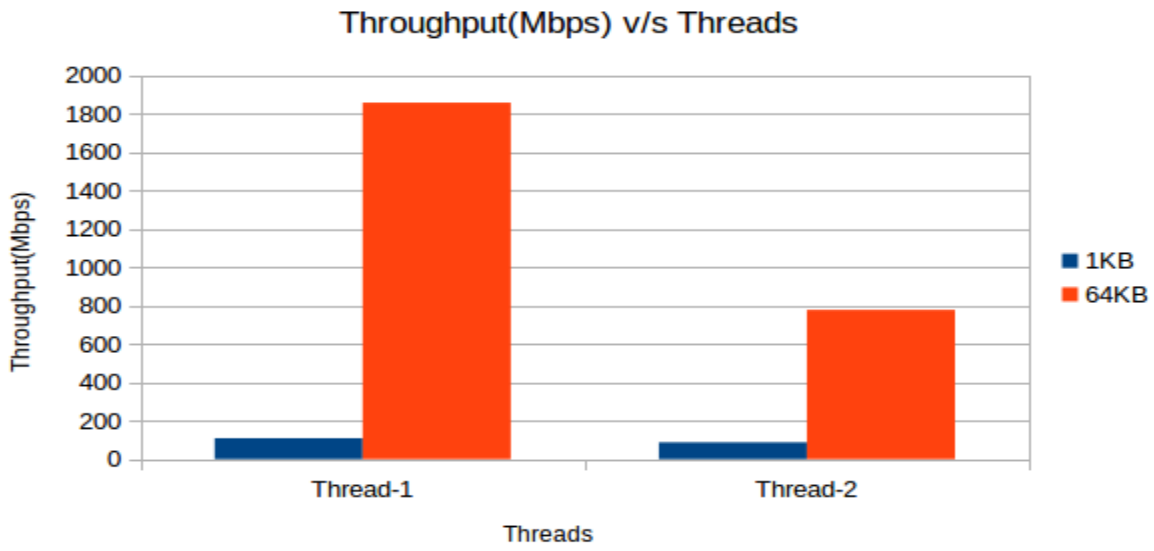
After performing disk benchmark on ec2 t2.micro instances, I can conclude that the speed of disk will be decreased as threads are increased and most of the time the latency of disk will be increase as threads are increased.

3. NETWORK BENCHMARK

For single and multi thread, I have computed benchmark for different block size (1 Byte, 1 KB, 64 KB) using different packet transfer protocol (TCP and UDP).

The throughput and latency performance for 1,2 thread and 1KB and 64KB blocks are as below :

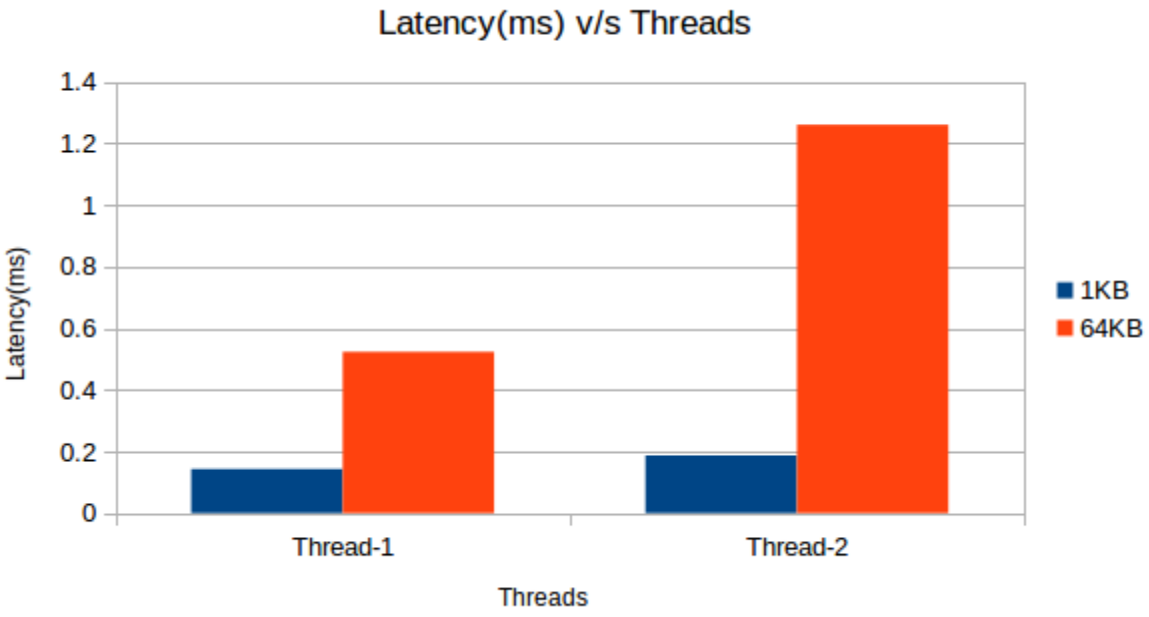
UDP Throughput (Thread-1,2, 1KB and 1MB)



The above graph represents relationship between threads and throughput performance. As we have only one thread per CPU, the performance will be decreased as threads are increased in both case of packet transfer protocol.

As UDP is connectionless protocol, the performance(throughput) for TCP is better than UDP and the latency is less than UDP.

UDP Latency (Thread-1,2 1KB and 1MB)



The above graph represents relationship between threads and latency performance. As we have only one thread per CPU, the latency will be decreased as threads are increased in both case of packet transfer protocol.

UDP benchmark results (1- Thread) :

Block Size	Average Throughput	Stddev	Average Latency	Stdev
1 B	0.095	0.034	0.163	0.048
1 KB	109.261	8.441	0.143	0.010
64 KB	1856.312	59.832	0.523	0.015

UDP benchmark results (2- Thread) :

Block Size	Average Throughput	Stddev	Average Latency	Stdev
1 B	0.056	0.009	0.123	0.084
1 KB	87.282	4.993	0.187	0.012
64 KB	777.403	83.297	1.267	0.136

TCP benchmark results (1- Thread) :

Block Size	Average Throughput	Stddev	Average Latency	Stdev
1 B	5.123	0.109	0.002	0.00005
1 KB	750.464	30.45	0.006	0.0002
64 KB	2234.546	95.23	0.133	0.084

TCP benchmark results (2- Thread) :

Block Size	Average Throughput	Stddev	Average Latency	Stdev
1 B	3.79	0.110	0.003	0.0001
1 KB	674.948	33.432	0.012	0.0006
64 KB	2081.233	59.234	0.167	0.043

d) Extra credit

I have run IPerf on ec2 and the best performance results that I have achieved are as below :

TCP Performance : 831 Mbits/sec

UDP Performance : 98.8 Mbits/sec

After performing network experiments on ec-2 t2.micro instance, I got some interesting results which are closed to Iperf evaluation results.

As per programming benchmark, I have evaluated results for 1 KB packet using TCP protocol which is 750.464 Mbps throughput.

As per programming benchmark, I have evaluated results for 1 KB packet using UDP protocol which is 109.36 Mbps throughput.

Efficiency compare to Iperf(TCP) = (750.46/831) % 100
= 90 %