Performance Report

for

Programming Assignment 1

Manthan Patel
(A-20413535)
mpatel120@hawk.iit.edu

Index

1. CPU Benchmark

The CPU benchmark has been tested on Hyperion cluster with 1, 2 and 4 threads for total 1 trillion arithmetic operations(all add, subtract, division and multiplication) and the output has been measured in GigaOps per sec unit. The benchmark has been run for 3 times in loop and the average value has been saved in output file.

The benchmark has been tested for Quarter Precision (**char datatype)**, Half precision **(sort datatype)**, single precision(**int datatype**), and double precision(**double datatype**) using 1,2 and 4 threads. The output for each precision has been saved in GigaOps per sec unit.

Performance report in Tabular form

- For the testing of performance of CPU I ran MyCPUBenchmark on local as well as on Hyperion and the values for Quarter Precision, Half Precision, Single Precision and Double Precision with concurrency 1,2 and 4 have stored in output file as well as written in below table.
- I have run the HPL benchmark from Linpack suite to check test my application's output for Double precision datatype only.
- I got nearly same output for QP, HP and SP for 1,2 and 4 threads values. However, I received very low values for DP.
- Following are the output taken from Hyperion cluster for MyCPUBenchmark and Linpack suite –

Workload	Concurrency	MyCPUBenchValue	HPL Measured Ops/sec (Giga OPS)	Theoretical Ops/Sec (GigaOPS)	MyCPUBench Efficiency (%)	HPL Efficiency (%)
QP	1	10.600263	N/A	588.8	1.800316	N/A
QP	2	20.665038	N/A	588.8	3.509687	N/A
QP	4	20.807768	N/A	588.8	3.533928	N/A
HP	1	10.435053	N/A	294.4	3.544515	N/A
HP	2	20.735727	N/A	294.4	7.043385	N/A
HP	4	20.493029	N/A	294.4	6.960947	N/A
SP	1	10.875358	N/A	147.2	7.388151	N/A
SP	2	20.906904	N/A	147.2	14.20306	N/A
SP	4	21.451774	N/A	147.2	14.57322	N/A
DP	1	4.656586	36.2241	73.6	3.560922	49.21753
DP	2	2.620838	68.752	73.6	6.326883	93.41304
DP	4	4.69694	66.6462	73.6	6.381712	90.5519

Table 1. Processor Performance

Graphical representation and conclusion

- Based on the values received for QP, HP, SP and DP for 1, 2, and 4 threads I have created various graphs as below.

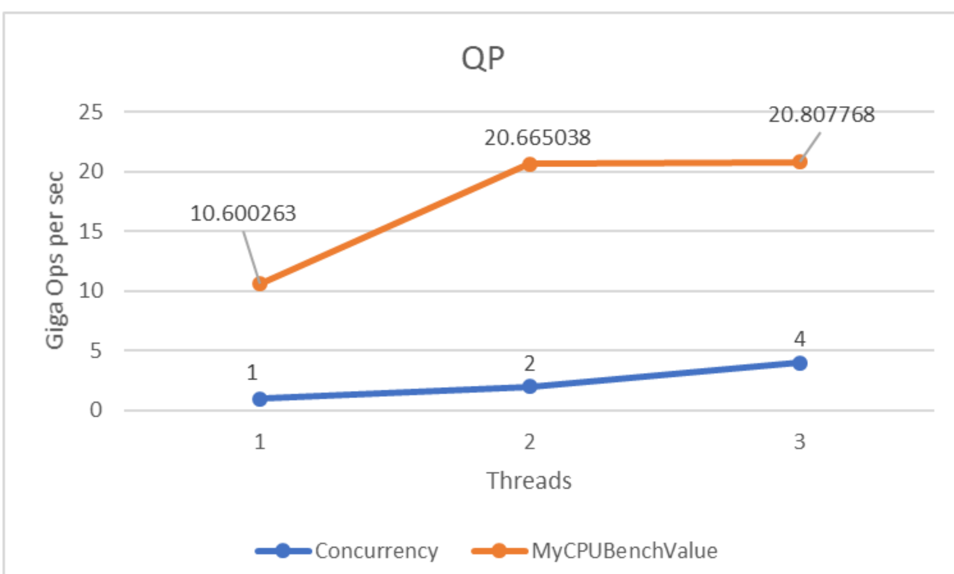

Fig 1 Throughput for QP

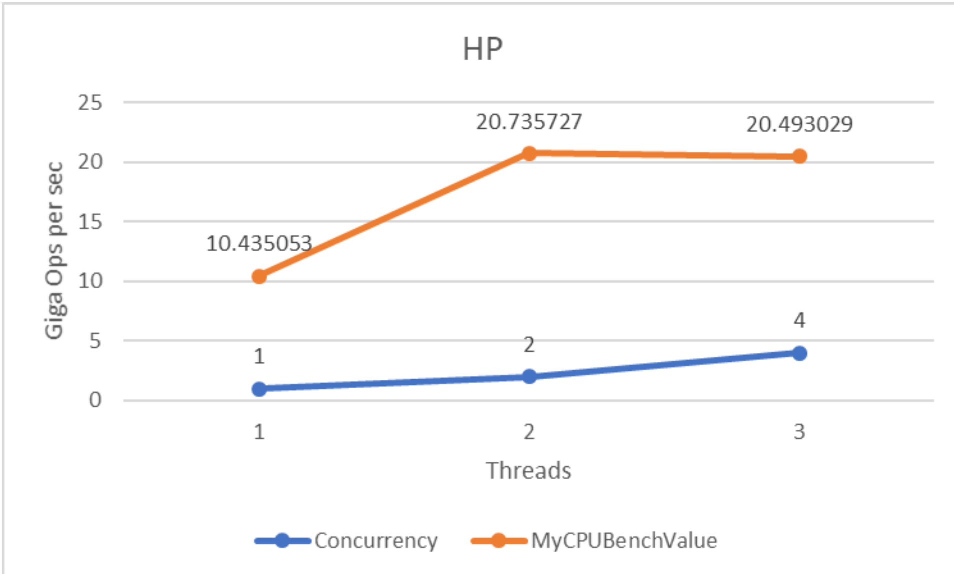

Fig 2 Throughput for HP

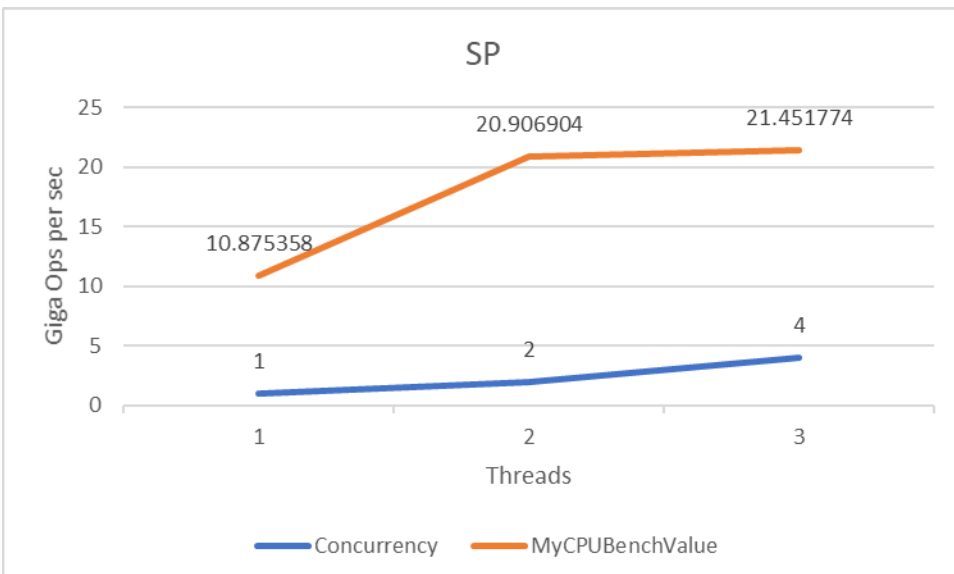

Fig 3 Throughput for SP

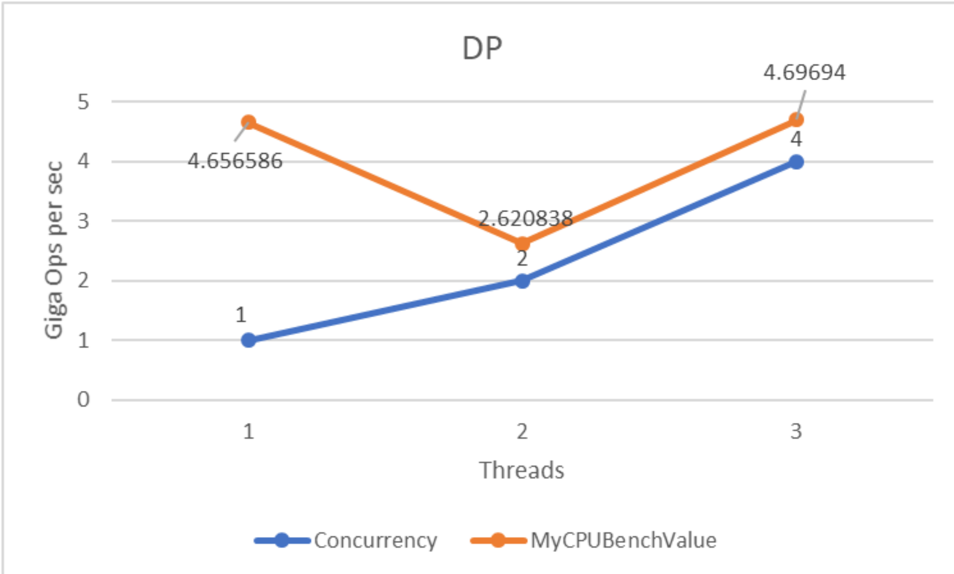

Fig 4 Throughput for DP

From the graph 1,2 and 3 we can say the performance of the CPU is almost same as the arithmetic operations have computed on RISC architecture of CPU. Hence, each cycle has used full length of memory size and evaluated all the operations in same manner. However, Double precision have shown more low values which might be because of double precision required more instruction cylcle then Single precision. Hence the behavior CPU archicature looks like RISC type from the above values and graphs.

Based on the output received I can concluded that MyCPUBenchmark has received **maximum efficiency** when the arithmetic operations have been calculated for **single precision with 4 threads.**

Formula for Theoretical value calculation

Each node in Hyperion cluster have used 2 CPUs having 2.3GHz with 2 cores and 16 Instruction per cycles for Double Precision. Hence the formula for Theoretical Throughput can be written as –

Theoretical Throughput for DP = 2 x 2.3 x 16 = 73.6 GigaOps per sec
Theoretical Throughput for SP = 2 x 2.3 x 32 = 147.2 GigaOps per sec
Theoretical Throughput for HP = 2 x 2.3 x 64 = 294.4 GigaOps per sec
Theoretical Throughput for QP = 2 x 2.3 x 128 = 588.8 GigaOps per sec

Efficiency(%) = MyCPUBenchValue x 100 / Theoretical Throughput

Although my benchmark's output is showing that the Instructions per cycle should not be doubled for HP like RISC architecture, I have taken the values based on the behavior of CISC architecture.

2. Disk Benchmark

The Disk Benchmark has been tested on Prometheus cluster for the operations Read Sequentially, Write Sequentially, Read Randomly and Write Randomly with multiple threads and different block size. The Throughput has been saved in MB/s for 10GB data, miliseconds for latency check and IOPS for the test of I/O operation randomly.

Performance report in Tabular form

- For reading the file, I have created a file of 10GB size with putting '\0' at the end of 10GB-th location so that the random read/write function can reach till end of the file and the core dump error do not occur.
- Below are the throughput and latency values for MyDiskBenchmark and IOZone benchmark taken from Prometheus cluster

Workload	Con-currency	Block Size (MB)	MyDiskBench Measured Throughput (MB/sec)	IOZone Measured Throughput (MB/sec)	Theoretical Throughput (MB/sec)	MyDiskBench Efficiency (%)	IOZone Efficiency (%)
RS	1	1	294.1974606	250	372	79.08533888	67.20430108
RS	1	10	227.3458921	249	372	61.11448713	66.93548387
RS	1	100	217.6434639	249	372	58.50630749	66.93548387
RS	2	1	350.3676405	261	744	47.0924248	35.08064516
RS	2	10	245.8096305	270	744	33.03892883	36.29032258
RS	2	100	242.6152664	294	744	32.60957881	39.51612903
RS	4	1	257.14462	270	1488	17.28122446	18.14516129
RS	4	10	228.6735748	288	1488	15.36784777	19.35483871
RS	4	100	226.0162905	300	1488	15.18926683	20.16129032
RR	1	1	206.5663359	255	540	38.25302516	47.22222222
RR	1	10	240.3030041	250	540	44.50055632	46.2962963
RR	1	100	151.3512515	227	540	28.02800954	42.03703704
RR	2	1	425.9853951	303	1080	39.44309214	28.05555556
RR	2	10	332.2539954	305	1080	30.76425883	28.24074074
RR	2	100	318.1616958	364	1080	29.45941627	33.7037037
RR	4	1	565.5200071	350	2160	26.18148181	16.2037037
RR	4	10	348.7558849	373	2160	16.14610578	17.26851852
RR	4	100	335.0615311	349	2160	15.51210792	16.15740741
WS	1	1	159.358142	130	172	92.65008256	75.58139535
WS	1	10	302.238716	154	172	175.7201837	89.53488372
WS	1	100	97.205245	150	172	56.51467733	87.20930233

WS	2	1	167.670485	157	344	48.74142006	45.63953488
WS	2	10	306.74787	172	344	89.17089244	50
WS	2	100	90.077825	173	344	26.18541424	50.29069767
WS	4	1	303.527202	161	688	44.11732587	23.40116279
WS	4	10	281.079331	162	688	40.85455392	23.54651163
WS	4	100	76.979359	162	688	11.18886032	23.54651163
WR	1	1	126.333386	234	410	30.81302098	57.07317073
WR	1	10	336.990435	237	410	82.19278902	57.80487805
WR	1	100	214.24373	255	410	52.25456829	62.19512195
WR	2	1	118.189093	278	820	14.41330402	33.90243902
WR	2	10	319.199605	292	820	38.9267811	35.6097561
WR	2	100	333.703643	305	820	40.69556622	37.19512195
WR	4	1	318.90532	202	1640	19.44544634	12.31707317
WR	4	10	164.275875	207	1640	10.01682165	12.62195122
WR	4	100	310.64206	207	1640	18.94158902	12.62195122

Table 2. Disk Throughput

Below is the latency output table –

Workload	Con-currency	Block Size (KB)	MyDiskBench Measured Latency (ms)	IOZone Measured Latency (ms)	Theoretical Latency (ms)	MyDiskBench Efficiency (%)	IOZone Efficiency (%)
WR	1	1	0.008408	1.3	0.5	98.3184	-160
WR	2	1	0.007431	1.3	0.5	98.5138	-160
WR	4	1	0.004414	2.5	0.5	99.1172	-400
WR	8	1	0.004441	1.3	0.5	99.1118	-160
WR	16	1	0.00423	0.7	0.5	99.154	-40
WR	32	1	0.005187	0.9	0.5	98.9626	-80
WR	64	1	0.005988	0.9	0.5	98.8024	-80
WR	128	1	0.004342	0.57	0.5	99.1316	-14
RR	1	1	0.001473	1.5	0.5	99.7054	-200
RR	2	1	0.001024	1.5	0.5	99.7952	-200
RR	4	1	0.000975	3.2	0.5	99.805	-540
RR	8	1	0.001081	0.77	0.5	99.7838	-54
RR	16	1	0.001018	0.8	0.5	99.7964	-60
RR	32	1	0.001111	0.45	0.5	99.7778	10
RR	64	1	0.000952	0.55	0.5	99.8096	-10
RR	128	1	0.001071	0.7	0.5	99.7858	-40

Table 3. Disk Latency (Measured in msec)

Following table shows the disk latency measured in IOPS –

Workload	Con-currency	Block Size (KB)	MyDiskBench Measured IOPS	IOZone Measured IOPS	Theoretical IOPS	MyDiskBench Efficiency (%)	IOZone Efficiency (%)
WR	1	1	118934.3482	504234	363281.25	32.73891736	138.799897
WR	2	1	134571.3901	345241	726562.5	18.52165369	47.5170409
WR	4	1	226551.8804	234252	1453125	15.59066704	16.1205677
WR	8	1	225174.5102	381250	2906250	7.747940137	13.1182796
WR	16	1	236406.6194	845762	5812500	4.067210656	14.5507441
WR	32	1	192789.6665	515432	11625000	1.658405733	4.43382366
WR	64	1	167000.668	924521	23250000	0.718282443	3.97643441
WR	128	1	230308.6135	205842	46500000	0.495287341	0.44267097
RR	1	1	678886.6259	216542	167968.75	404.1743633	128.918028
RR	2	1	976562.5	236412	335937.5	290.6976744	70.3738047
RR	4	1	1025641.026	135495	671875	152.653548	20.1666977
RR	8	1	925069.3802	510582	1343750	68.84237248	37.9968
RR	16	1	982318.2711	710562	2687500	36.55137753	26.4395163
RR	32	1	900090.009	875461	5375000	16.74586063	16.2876465
RR	64	1	1050420.168	895682	10750000	9.771350401	8.33192558
RR	128	1	933706.8161	686456	21500000	4.3428224	3.1928186

Table 4. Disk latency (measured in IOPS)

Graphical representation and conclusion

I have created various graphs based on the 3 parameters (Throughput, No of threads and block size) for Disk throughput values representation (taken from table 1) and latency(measured in msec as well as IOPS).

The Graphs for the Read/write sequential and random are as follow –

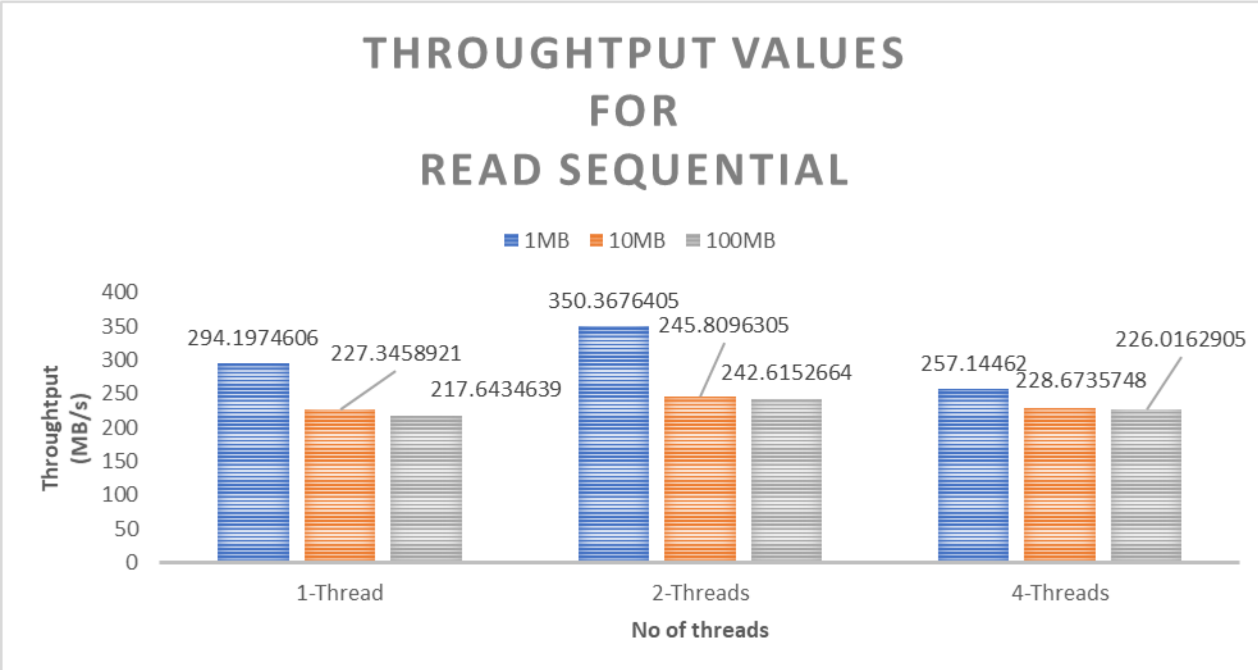

Fig 5. Throughput for RS

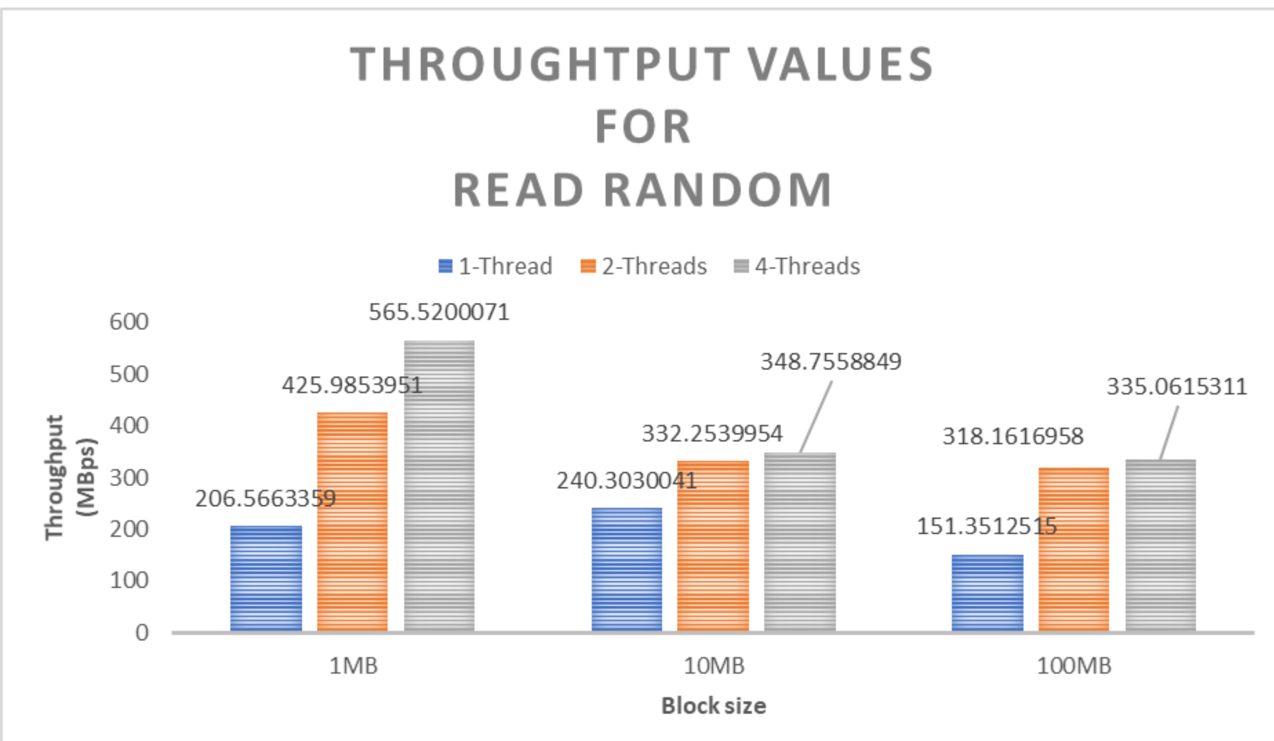

Fig 6. Throughput for RR

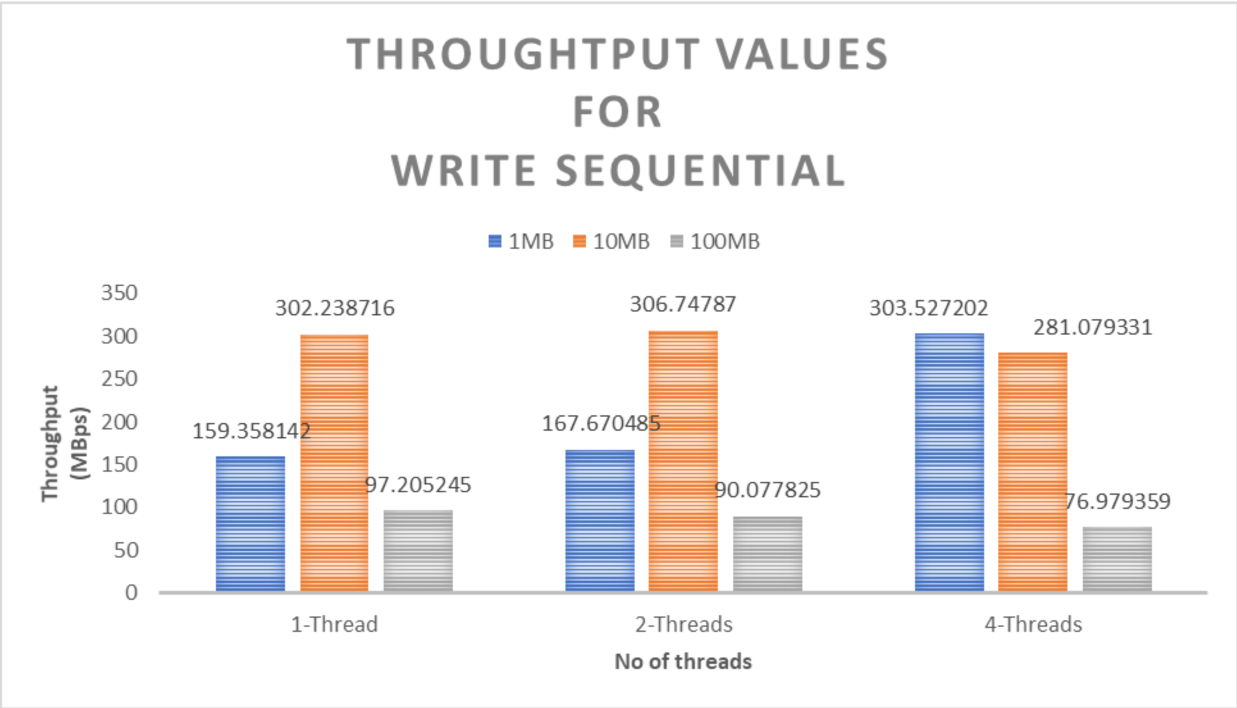

Fig 7. Throughput for WS

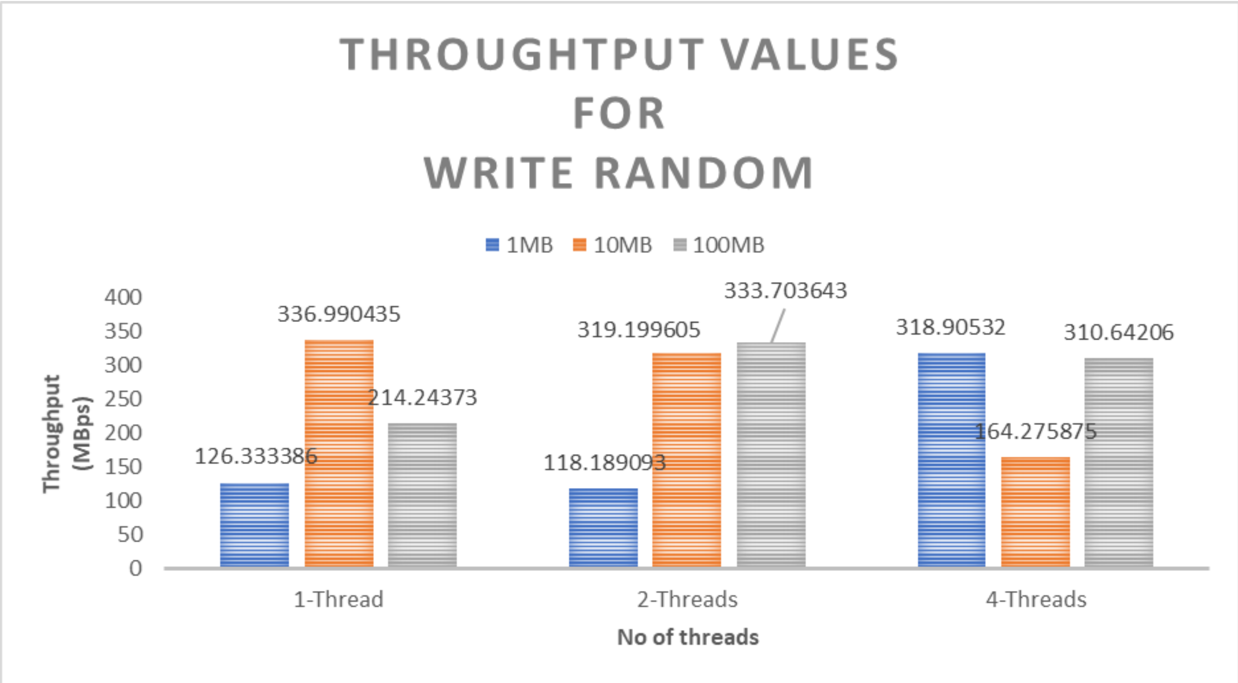

Fig 8. Throughput for WR

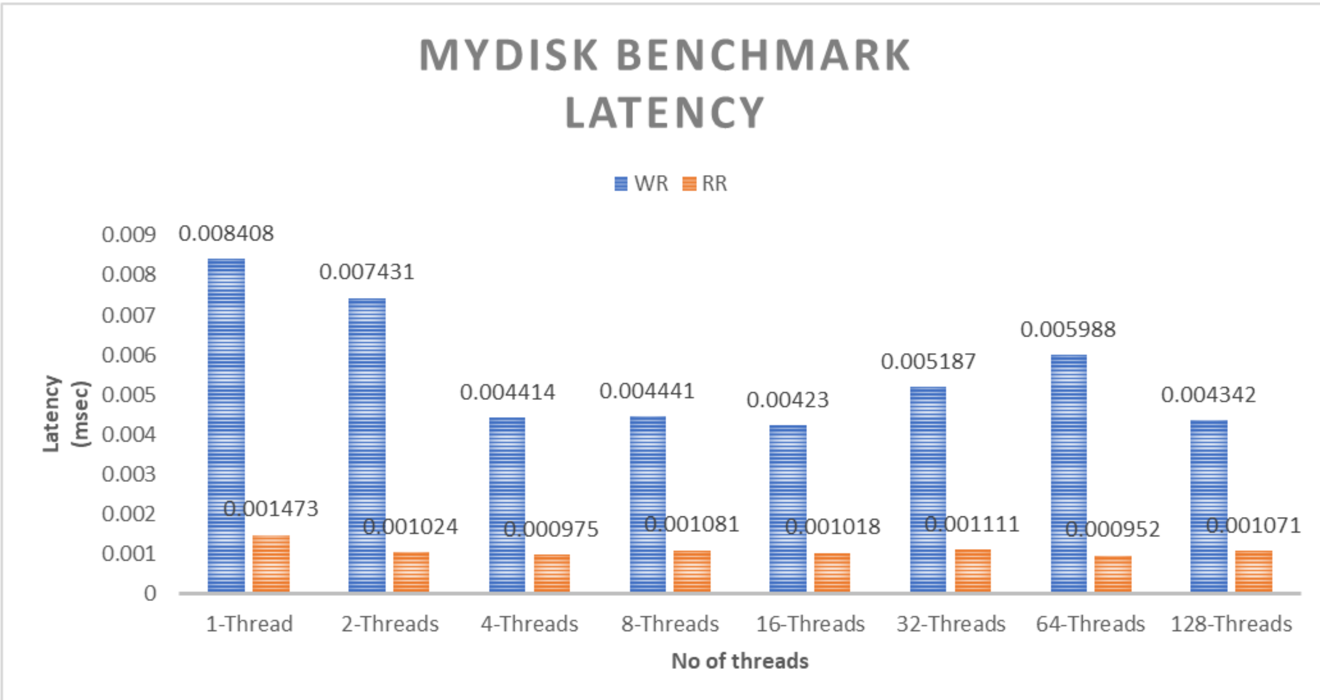

Fig 9. Latency in msec

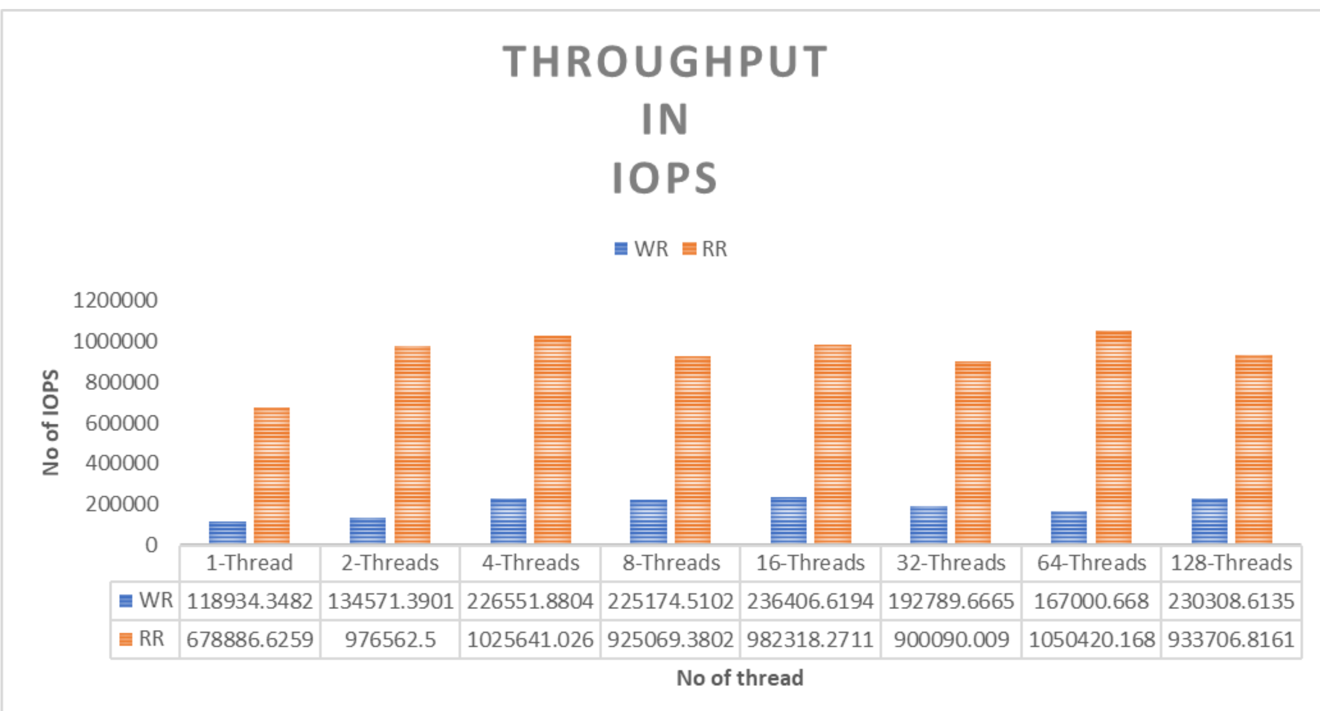

Fig 10. Latency in IOPS

- From Fig 5, as the threads increases the reading speed in sequential way gets decreases for the 1MB data block. However, the speed remains nearly same for all 1,2 and 4 threads for 10MB and 100MB data blocks. The highest efficient reading sequential way is 1 thread having 1MB data block(from table).

- From Fig 6, as the threads increases the reading speed gets increases for al 1MB, 10MB and 100MB data blocks. The Highest speed got at 1MB block with 4 threads.

- From Fig 7 and 8, the better way of wirtting to disk is write randomly as the speed for write randomly has highest value amongs them. The highest speed got for 10MB data block having 1 thread in write randomly.

- From fig 9 and 10, the latency for WR has very low values compared to RR in IOPS and has very high values compared to RR in mili seconds.

In conclusion, the best way to read the data from disk is read randomly with 4 threads and 1MB Data block size. And for write the data to disk is write randomly with 10MB data block having 1 thread. Hence, the random way of read and write gives more better output than sequential way.

Formula for Theoretical value calculation

Total workload = 10 GB
Throughput (in Mb/s) = Total workload / (Time taken x 1000000)
Theoretical Throughput RS = 372Mbps
Theoretical Throughput RR = 172Mbps
Theoretical Throughput WS = 540Mbps
Theoretical Throughput WR = 410Mbps
Throughput Efficiency = (Throughput x100)/ Theoretical Throughput
Latency (in milliseconds) = (execution time x 1000) / 1000000
Theoretical Latency = 0.5 milliseconds
Latency Efficiency = 100 - (Latency x 100 / Theoretical Latency)
Latency (in IOPS) = 1000000 / Time taken

3. Memory Benchmark

The Memory Benchmark has been tested on Hyperion cluster for testing the memory read-write operation speed (in MB/s) and the latency (in µs) with changes of threads.

Performance report in Tabular form

- I have used random generator function for generating random number in the range of 0 to (10^9/block_size) and the received value has been used to place the whole block to the given position.
- Based on the performance of the algorithm the output of threoughput, latency and efficiency for Read-write-sequential and Read-write-random with different threads and different block sizes have been given as below –

Workload	Con-currency	Block Size	MyRAMBench Measured Throughput (GB/sec)	Theoretical Throughput (GB/sec)	MyRAMBench Efficiency (%)	pmbw Measured Throughput (GB/sec)	pmbw Efficiency (%)
RWS	1	1000	3.62854	68.256	5.316074777	16.88	24.73042663
RWS	1	1000000	3.876088	68.256	5.678750586	22.1	32.37810595
RWS	1	10000000	2.968031	68.256	4.348381095	18.45	27.03059072
RWS	2	1000	6.571116	68.256	9.627162447	23.46	34.37060478
RWS	2	1000000	6.872976	68.256	10.06940928	15.32	22.44491327
RWS	2	10000000	5.407683	68.256	7.922648558	34.88	51.10173465
RWS	4	1000	6.463568	68.256	9.469596812	23.55	34.50246132
RWS	4	1000000	6.739439	68.256	9.873767874	37.46	54.88162213
RWS	4	10000000	5.243572	68.256	7.682214018	31.67	46.39885138
RWR	1	1000	2.052113	68.256	3.006494667	3.82	5.59657759
RWR	1	1000000	3.101437	68.256	4.543830579	1.04	1.523675574
RWR	1	10000000	4.292492	68.256	6.288812705	0.88	1.289263947
RWR	2	1000	2.25328	68.256	3.30121894	8.42	12.33591186
RWR	2	1000000	5.896757	68.256	8.639177508	1.55	2.270862635
RWR	2	10000000	7.721459	68.256	11.31249853	1.3	1.904594468
RWR	4	1000	2.549801	68.256	3.735643753	18.77	27.49941397
RWR	4	1000000	5.923526	68.256	8.678396038	2.89	4.234060009
RWR	4	10000000	7.664487	68.256	11.22903041	1.12	1.640881388

Table 5 Memory Throughput

Workload	Con-currency	Block Size	MyRAMBench Measured Throughput (GB/sec)	Theoretical Throughput (GB/sec)	MyRAMBench Efficiency (%)	pmbw Measured Throughput (GB/sec)	pmbw Efficiency (%)
RWS	1	1	0.004578	0.015	69.48	0.035	-133.33333
RWS	2	1	0.002637	0.015	82.42	0.0395	-163.33333
RWS	4	1	0.003765	0.015	74.9	0.042	-180
RWR	1	1	0.044067	0.015	-193.78	0.21	-1300
RWR	2	1	0.144257	0.015	-861.7133333	0.381	-2440
RWR	4	1	0.263541	0.015	-1656.94	0.516	-3340

Table 6 Memory Latency

Graphical representation and conclusion

Below are the graphs for the throughput and latency values based on 3 parameters –

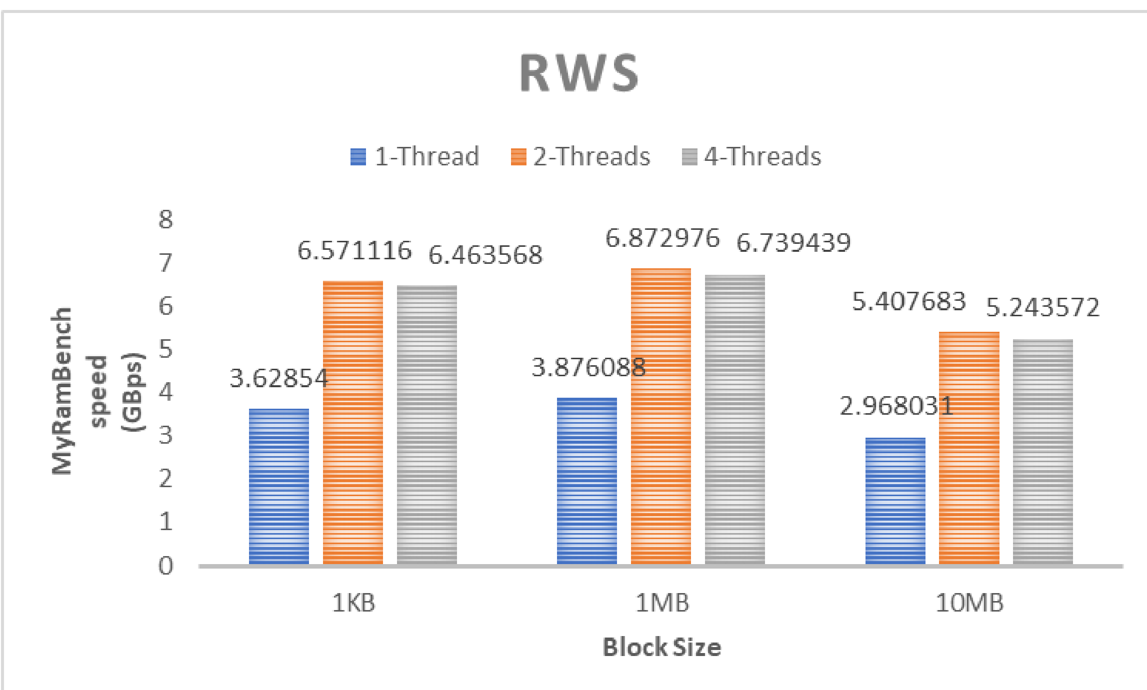

Fig 11. Throughput for RWS

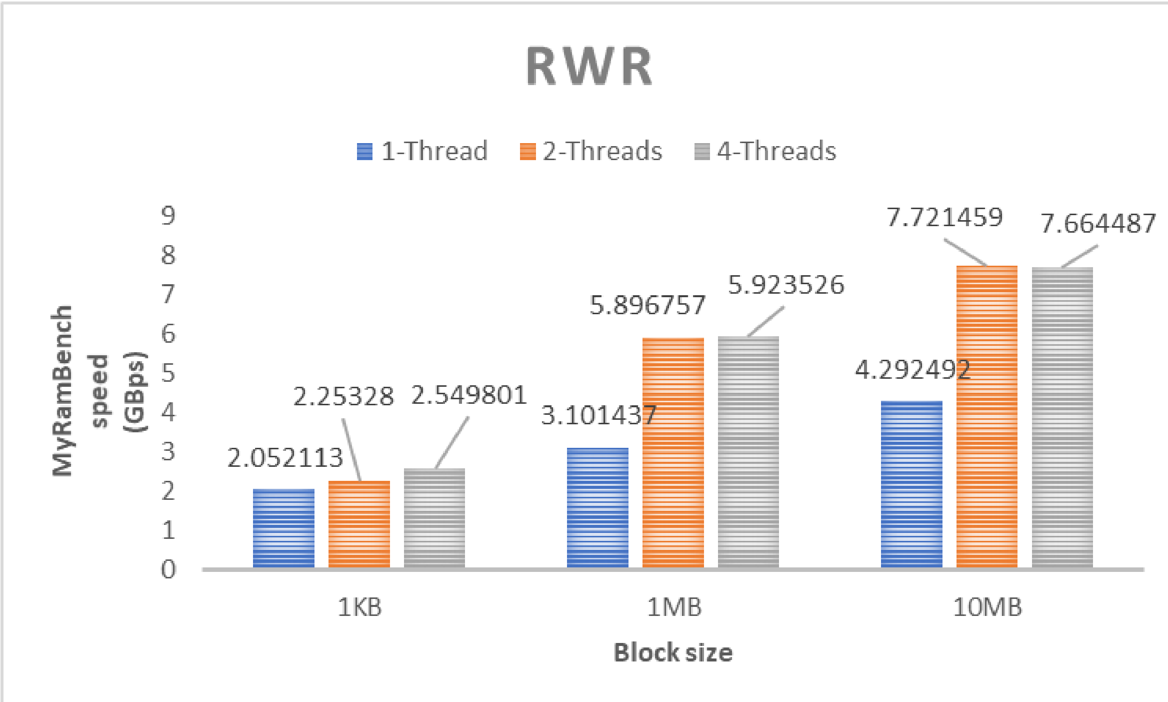

Fig 12. Throughput for RWR

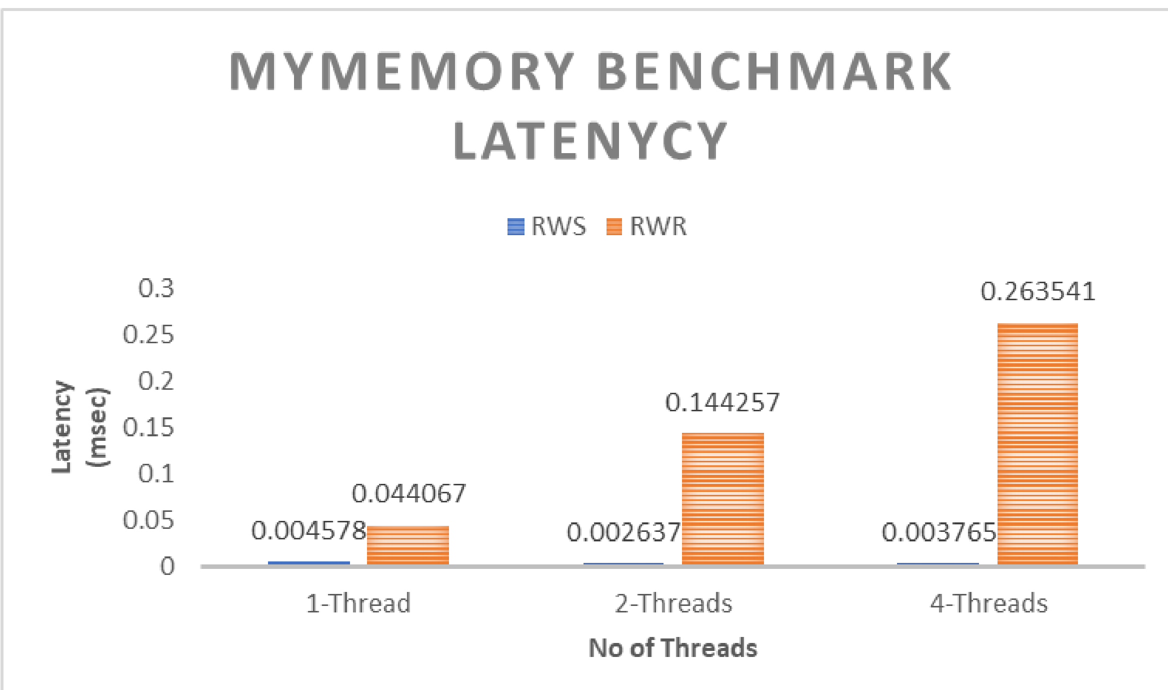

Fig 13. Latency

- From Fig 11 and 12, The highest speed for read-write operation got when we do operation randomly with 2 or 4 threads as both the values are nearly equal. However, the speed in sequential read + write operations are almost same for 2 and threads for all three block sizes but overall speed compared based on threads and block sizes then read + write randomly gives better performance than sequential way.
- From Fig 13, The RWS operations have very low latency which shows a better performance then RWR.

Formula for Theoretical value calculation

Workload = 1 GB
Throughput (in GB/s) = (100 x Workload) / (Time taken x 1000000000)
Theoretical Throughput (in GB/s) = (2133 MHz x 32Bytes)/1000
Latency (in µs) = (Time taken * 1000000) / 100 x 1000000

4. Network Benchmark

This benchmark has been developed for testing the network speed for the TCP and UDP protocol with single and multiple sender and receiver connections. The output has been taken in Mb/sec as Throughput and in mili seconds as Latency for ping-pong.

Performance report in Tabular form

- I have used the sockets for the development of this benchmark.
- For TCP Network Benchmark, I have used IPv4 and TCP protocol while creating the socket and pointed to the localhost while binding the connection. At the time of client call, we need to pass the address of the server where client can connect.
- For UDP Network Benchmark, I have used IPv4 and UDP protocol at the time of socket creation and assigned address as localhost. When client wants to send data, client uses the address given at the run time.
- Based on the performance of the algorithm, the output of my benchmark and iperf is as below -

Protocol	Con-currency	Message Size (KB)	MyNETBench Measured Throughput (Mb/sec)	Theoretical Throughput (Mb/sec)	MyNETBench Efficiency (%)	iperf Measured Throughput (Mb/sec)	iperf efficiency
TCP	1	1	1445.958794	10000	14.459588	2004.17	20.0417
TCP	1	32	2150.925819	10000	21.509258	2103.36	21.0336
TCP	2	1	3508.155233	10000	35.081552	3492.21	34.9221
TCP	2	32	4135.599013	10000	41.35599	4020.35	40.2035
TCP	4	1	2631.016187	10000	26.310162	4138.23	41.3823
TCP	4	32	9161.777103	10000	91.617771	4699.22	46.9922
TCP	8	1	3552.918122	10000	35.529181	4277.28	42.7728
TCP	8	32	6921.155238	10000	69.211552	5926.83	59.2683
UDP	1	1	2210.00397	10000	22.1000397	1172.35	11.7235
UDP	1	32	4361.856092	10000	43.618561	5837.14	58.3714
UDP	2	1	2804.770775	10000	28.04770775	1326.55	13.2655
UDP	2	32	4950.981324	10000	49.509813	9427.22	94.2722
UDP	4	1	7516.813797	10000	75.16813797	3299.6	32.996
UDP	4	32	7680.988574	10000	76.80988574	7927.89	79.2789
UDP	8	1	7428.536366	10000	74.28536366	1638.92	16.3892
UDP	8	32	8358.353721	10000	83.58353721	9217.62	92.1762

Table 7 Throughput for TCP and UDP network

Protocol	Con-currency	Message Size (B)	MyNETBench Measured Latency (msec)	Theoretical Latency (msec)	MyNETBench Efficiency (%)	iperf Measured Latency (msec)	iperf efficiency
TCP	1	1	0.197233	0.0007	-28076.1317	0.00431	-515.71429
TCP	2	1	0.089715	0.0007	-12716.44196	0.0031	-342.85714
TCP	4	1	0.050378	0.0007	-7096.825893	0.0029	-314.28571
TCP	8	1	0.030351	0.0007	-4235.868862	0.0055	-685.71429
UDP	1	1	0.222486	0.0007	-31683.67857	0.0041	-485.71429
UDP	2	1	0.338424	0.0007	-48246.22321	0.0026	-271.42857
UDP	4	1	0.222443	0.0007	-31677.625	0.00092	-31.428571
UDP	8	1	0.372961	0.0007	-53180.15179	0.00088	-25.714286

Table 8 Ping pong Latency for TCP and UDP

Graphical representation and conclusion

Following are the various graphs based on 3 parameters (Throughput or latency, no of Threads and message package size or protocol type) –

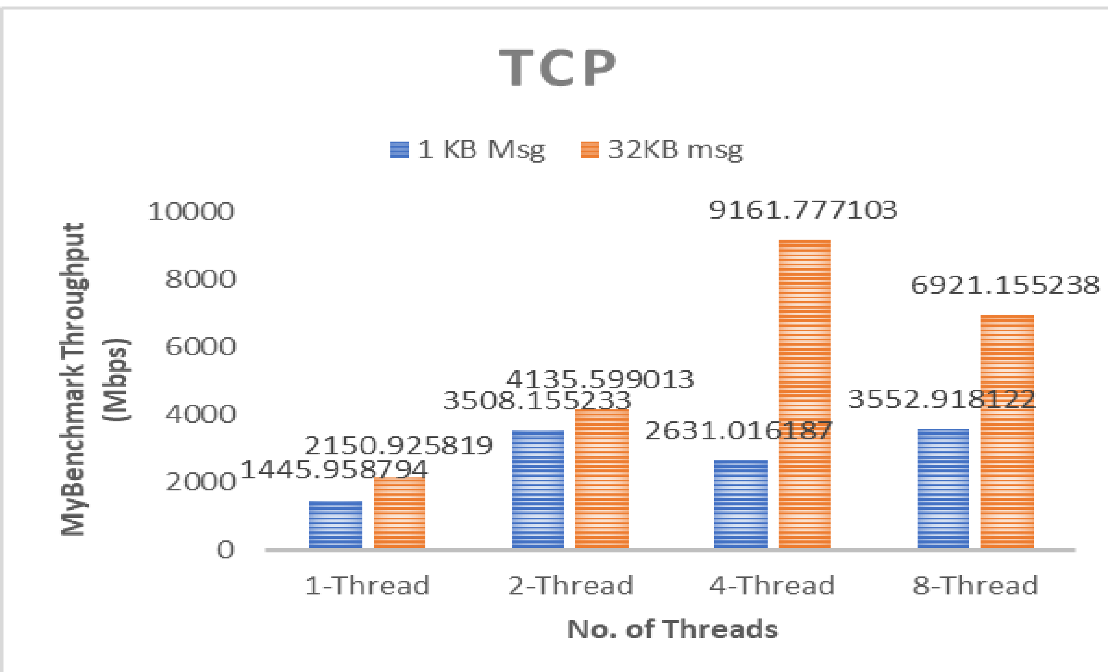

Fig 14. TCP Protocol throughput

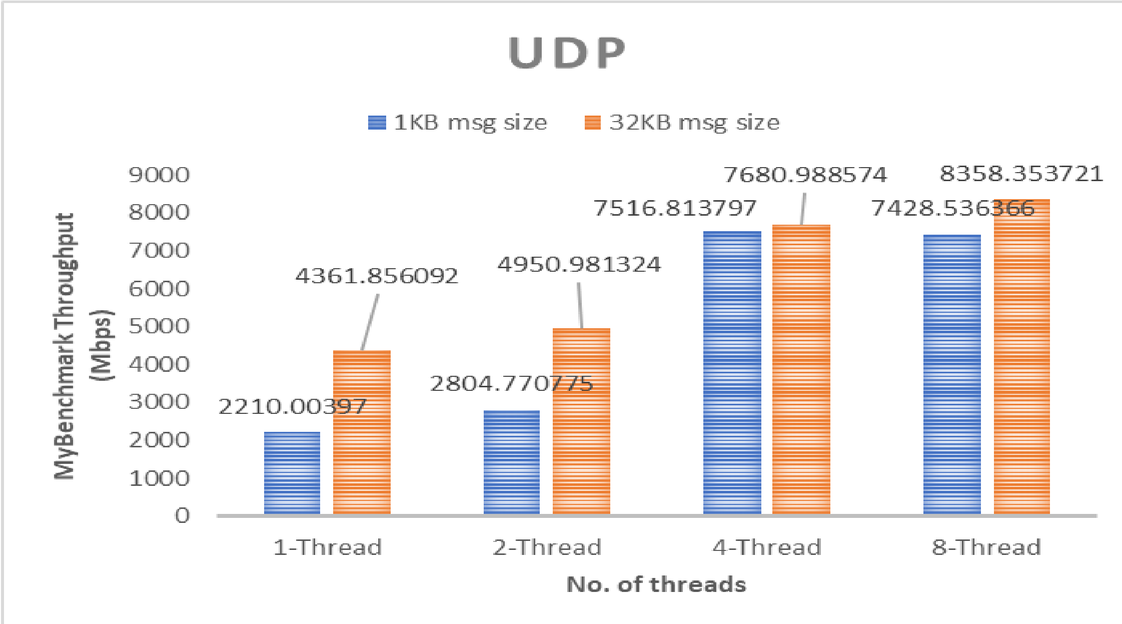

Fig 15. UDP Protocol throughput

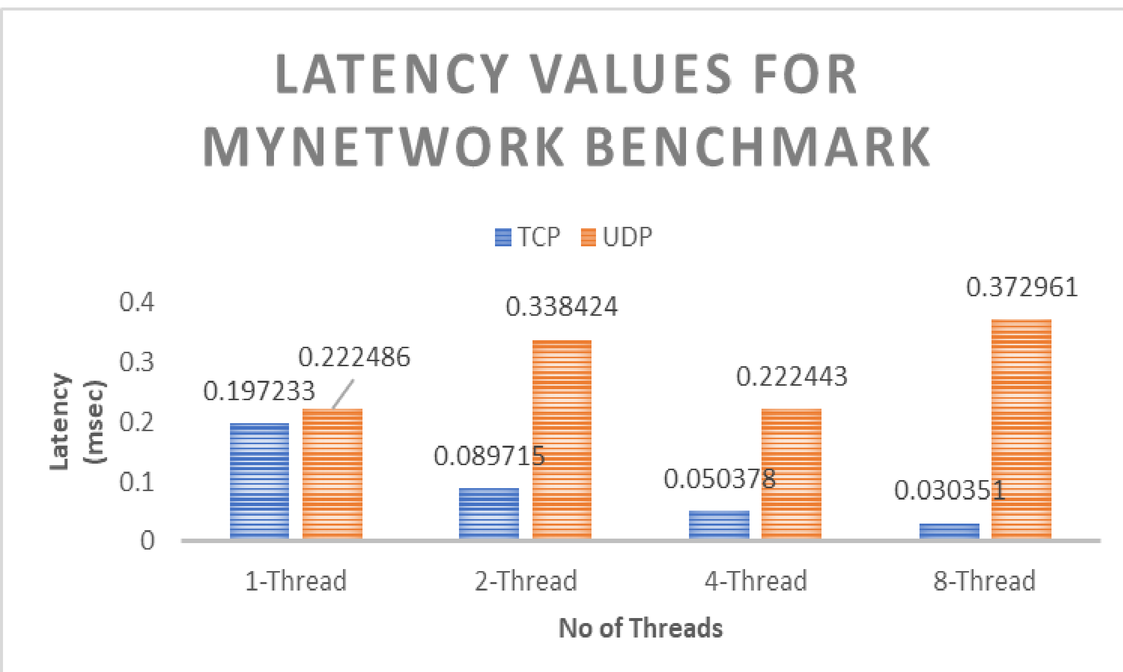

Fig 16. Latency comparison for TCP and UDP protocol

- From fig 14, TCP protocol works better with 4 threads having 32KB of package size. The maximum efficiency has been achieved when the connection occurred by 4 threads at each end and message have been passed in 32KB size.
- From Fig 15, UDP shows increasing behavior for speed as the threads get increased for both 1KB and 32KB. However, there are very high chance of package loss at receiver side and the high waiting time for receiver. The highest speed has been received at 8 threads and 32 KB package passing.
- From Fig 16, the fig shows the latency comparison for TCP and UDP protocol. For TCP, the latency value gets decreases as the threads get increases. Hence, the performance of the network shows better result as the threads get increases. However, there is no such good response for UDP when the threads get increases. Instead the performance might get decreases. One of the reason for this that UDP is connection less protocol hence it has to manage everything in the whole round trip.

Formula for Theoretical value calculation

For Testing the benchmark, following formulas and the values have been used –

Total workload = 1 GB
Total Iterations = 100 time
Throughput (in Mb/s) = (Total Iterations x Total workload x 8) / (Time taken x 1MB)
Theoretical Throughput (for TCP) = 10000 Mb/s
Theoretical Throughput (for UDP) = 10000 Mb/s
Efficiency (%) = (Throughput x 100) / Theoretical Throughput
Latency (in msec) = (Time taken x 1000) / Total workload (here 1 MB workload passed)
Theoretical Latency (in msec) = 0.0007
Efficiency for Latency (%) = 100 – (Latency x 100 / Theoretical Latency)