Performance

CPU Benchmark:

Hardware information:

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                2
On-line CPU(s) list:   0,1
Thread(s) per core:    1
Core(s) per socket:    1
Socket(s):             2
NUMA node(s):          1
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 42
Model name:            Intel Xeon E312xx (Sandy Bridge)
Stepping:              1
CPU MHz:               2299.996
BogoMIPS:              4599.99
Hypervisor vendor:     KVM
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              4096K
NUMA node0 CPU(s):     0,1
```

Computation:

Theoretical performance in GigaOPS = (CPU speed in GHz) * (number of CPU cores) * (number of CPUs per node) * (CPU instruction per cycle) = 2.30 * 2 * 2 * 8 = 73.6 GigaOPS

My program performance in GigaOPS = number of arithmetic operations/execution time/1,000,000,000 = 1,000/execution time

The results of my program and linpack are shown on Table 1.

Workload	Concurrency	MyCPUBench Measured Ops/Sec (GigaOPS)	HPL Measured Ops/Sec (GigaOPS)	Theoretical Ops/Sec (GigaOPS)	MyCPUBench Efficiency(%)	HPL Efficiency(%)
QP	1	2.8818	N/A	73.6	3.9	N/A
QP	2	5.0761	N/A	73.6	6.9	N/A
QP	4	5.1282	N/A	73.6	7.0	N/A
HP	1	2.6954	N/A	73.6	3.7	N/A
HP	2	5.0000	N/A	73.6	6.8	N/A
HP	4	5.0761	N/A	73.6	6.9	N/A
SP	1	2.7248	N/A	73.6	3.7	N/A
SP	2	4.8544	N/A	73.6	6.6	N/A
SP	4	5.6818	N/A	73.6	7.7	N/A
DP	1	2.7933	35.5044	73.6	3.8	48.2
DP	2	5.1546	67.2236	73.6	7.0	91.3
DP	4	5.2910	67.0702	73.6	7.2	91.1

Table 1

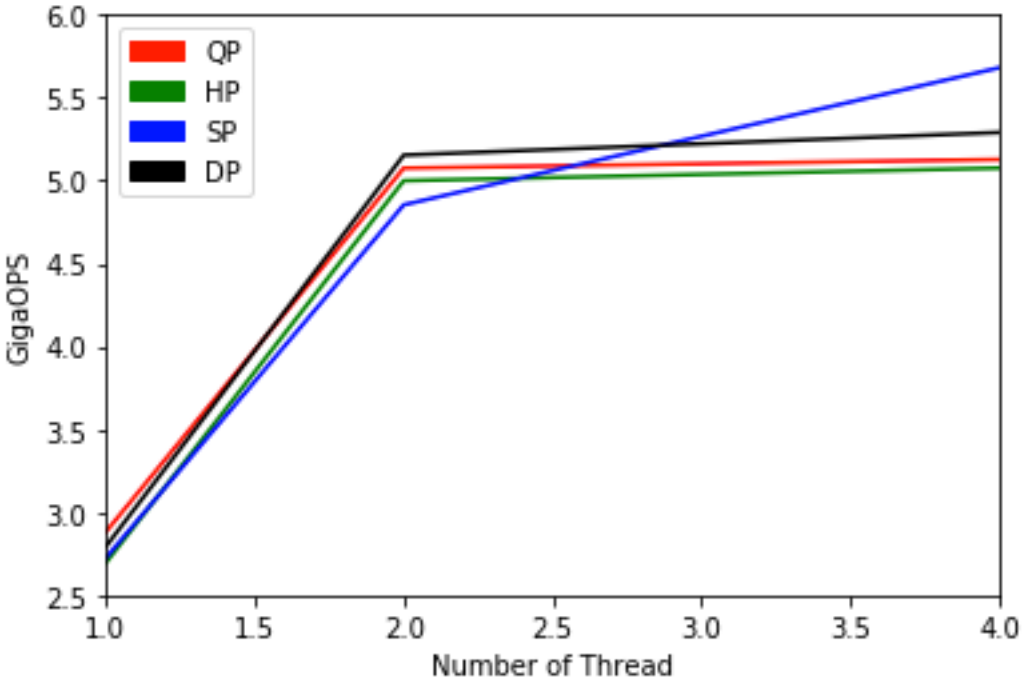

Figure 1

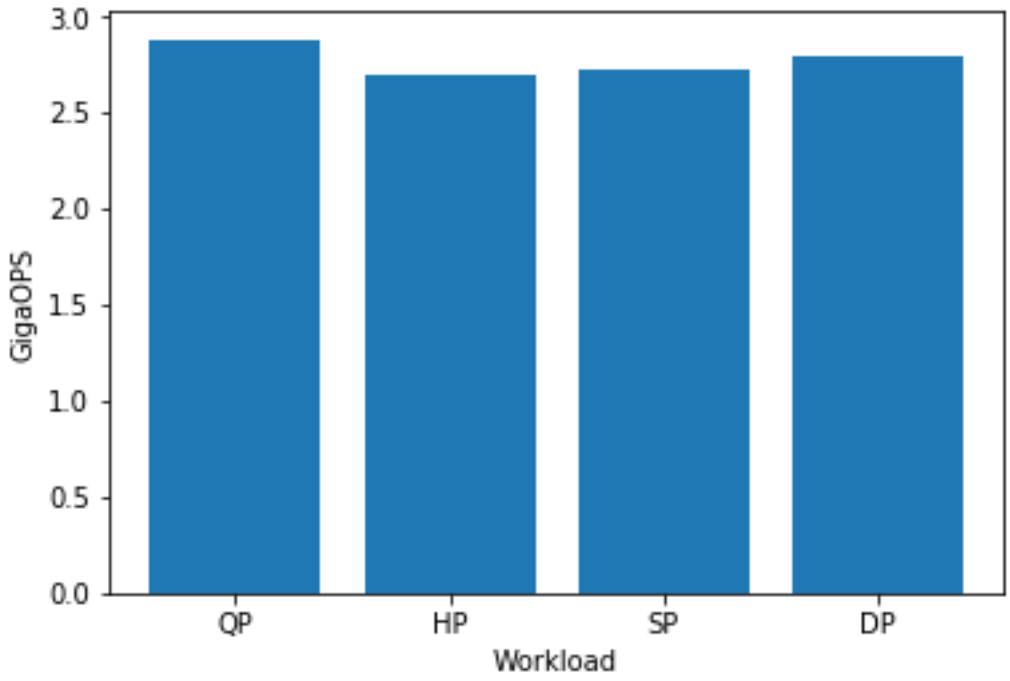

Figure 2

Observations:

1) Figure 1 shows the relationship of speed of processor(GigaOPS) and number of thread. From figure 1, we can see that the speed of processor increases almost 2 times from 1 thread to 2 threads. However, it seems have no improvement from 2 threads to 4 threads
Reason: Here are only 2 CPUs per node on hardware.

2) Figure 2 shows the relationship of speed of processor(GigaOPS) and workload when thread number is 1. From figure 2, we can see that the difference of GigaOPS of different workload is very small.
Reason: At 64 bits operating system, char, short, int and double variables all can be store in one address.

Memory Benchmark:

Hardware information:
Description: DDR4 DIMM Synchronous 2133 MHz (CAS CL15)
Product: HMA42GR7MFR4N-TF
Vendor: 00AD00B300AD
Size: 16GiB (this is the physical capacity, but you are using only 4GiB)
Width: 64 bits
Clock: 2133 MHz (CAS CL15)

Computation:
Theoretical throughput = DRAM * Number of data transfers per clock * Memory bus width * Number of interfaces = 2,133,000,000 * 2 * 64 * 2 = 546.048 Gb/s = 68.256GB/s
Theoretical latency = 0.4ns

Throughput of my program = Total workload/execution time = 100,000,000,000/execution time
Latency of program = execution time/number of operations = execution time/100,000,000

Since the smallest block size of pmbw is 1KB, I use the data of 1KB block size in pmbw to compute latency.
Latency of pmbw = execution time/number of operations/1,024

The throughputs of my program and pmbw are shown on table 2.
The latency of my program and pmbw are shown on table 3.

Work-load	Concu-rrency	Block Size	MyRAMBench Measured Throughput (GB/sec)	pmbw Measured Throughput (GB/sec)	Theoretical Throughput (GB/sec)	MyRAMBench Efficiency(%)	pmbw Efficien-cy(%)
RWS	1	1KB	2.548	16.004	68.256	3.7	23.4
RWS	1	1MB	2.746	18.110	68.256	4.0	26.5
RWS	1	10MB	2.555	16.941	68.256	3.7	24.8
RWS	2	1KB	5.178	29.062	68.256	7.6	42.6
RWS	2	1MB	5.190	35.591	68.256	7.6	52.1
RWS	2	10MB	4.963	32.642	68.256	7.3	47.8
RWS	4	1KB	5.484	36.817	68.256	8.0	53.9
RWS	4	1MB	5.575	38.820	68.256	8.2	56.9
RWS	4	10MB	4.673	35.734	68.256	6.8	52.4
RWR	1	1KB	1.342	5.072	68.256	2.0	7.4
RWR	1	1MB	2.544	0.522	68.256	3.7	0.76
RWR	1	10MB	3.128	0.408	68.256	4.6	0.60
RWR	2	1KB	1.587	9.543	68.256	2.3	14.0
RWR	2	1MB	4.682	1.263	68.256	6.9	1.9
RWR	2	10MB	6.080	0.807	68.256	8.9	1.2
RWR	4	1KB	1.680	18.723	68.256	2.5	27.4
RWR	4	1MB	4.996	2.685	68.256	7.3	3.9
RWR	4	10MB	6.193	0.856	68.256	9.1	1.3

Table 2

Work-load	Concu-rrency	Block Size	MyRAMBench Measured Latency(us)	pmbw Measured Latency(us)	Theoretical Latency (us)	MyRAMBench Effiency(%)	pmbw Efficiency (%)
RWS	1	1B	0.002984	0.000624	0.0004	13.4	64.1
RWS	2	1B	0.001813	0.000344	0.0004	22.1	116
RWS	4	1B	0.001722	0.000272	0.0004	23.2	147
RWR	1	1B	0.097451	0.001972	0.0004	0.41	20.3
RWR	2	1B	0.284328	0.001050	0.0004	0.14	38.1
RWR	4	1B	0.343422	0.000534	0.0004	0.12	74.9

Table 3

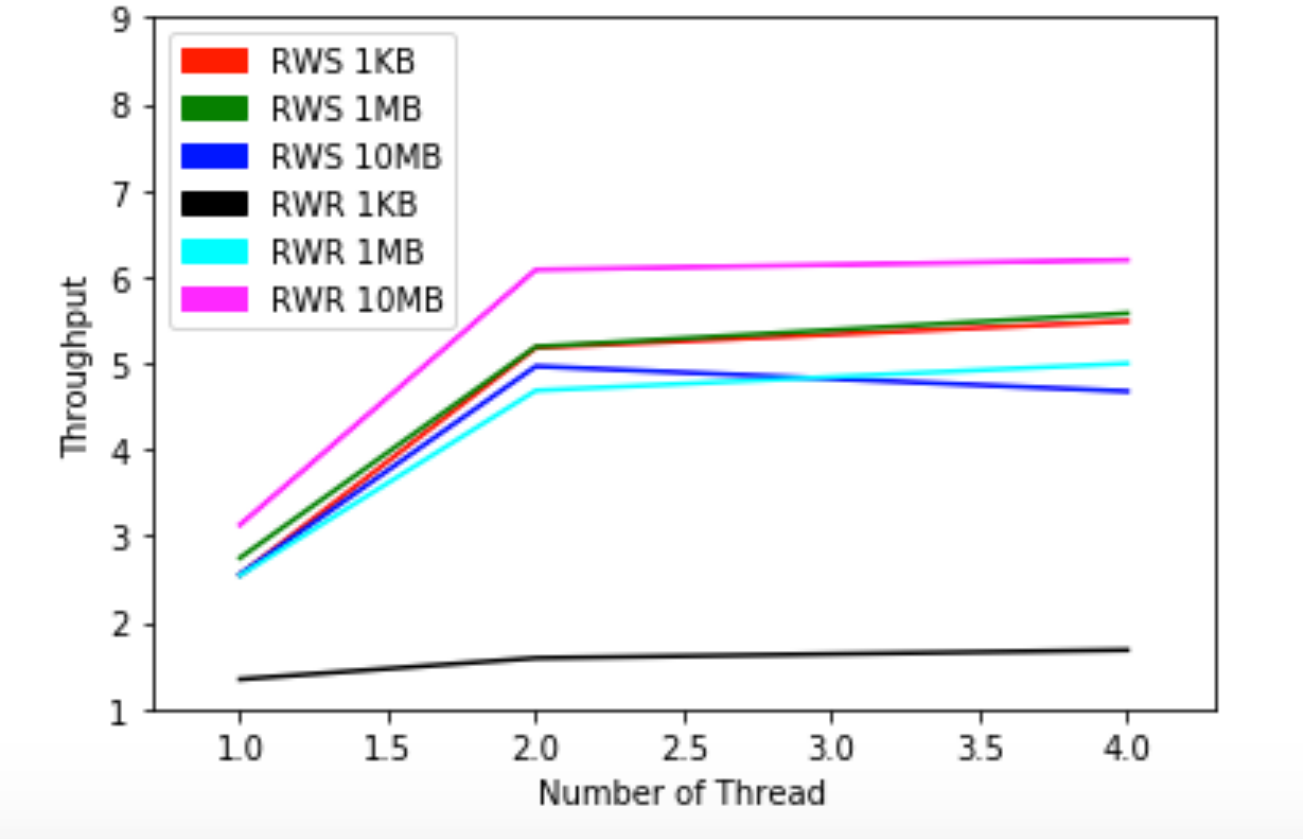

Figure 3

Observations:
1) Figure 3 shows the relationship of Throughput(MB/s) and number of threads. From figure 3, we can see that the throughput of memory almost doubles when number of thread increases from 1 to 2. But when number of thread increases from 2 to 4, throughput decreases a little.
Reason: The memory has only 2 hardware threads, when threads are more than 2, they will influence each other and throughput will decrease a little.

2) Figure 4 shows the relationship of Throughput(MB/s) and block size. For RWS access pattern, the throughputs are not change a lot by different block size. And throughputs get their max value when block size is 1MB. For RWR access pattern, throughput increases a lot as block size.
Reason: Unreasonable change in RWR result from I use rand() function in every loop of my program, this function cost a lot of time. As the value of block size increases, the number of loops decreases. So the rand() function cost less when block size is large. 1MB may be suitable for this memory by the way.

3) Figure 5 shows throughput of RWS and RWR when block size is 10MB and thread is 1. The reason I choose 10MB block size is that it has least rand() function cost and result can be more correct. From figure 5, we can see that the throughput of RWR is larger than RWS.
Reason: Random access pattern is faster than sequential access pattern.

4) There are some data of pmbw latency is smaller than theoretical latency.
Reason: My method of computing latency by throughput and block size will reduce value of latency because I assume that memory take latency time for every byte read and write.

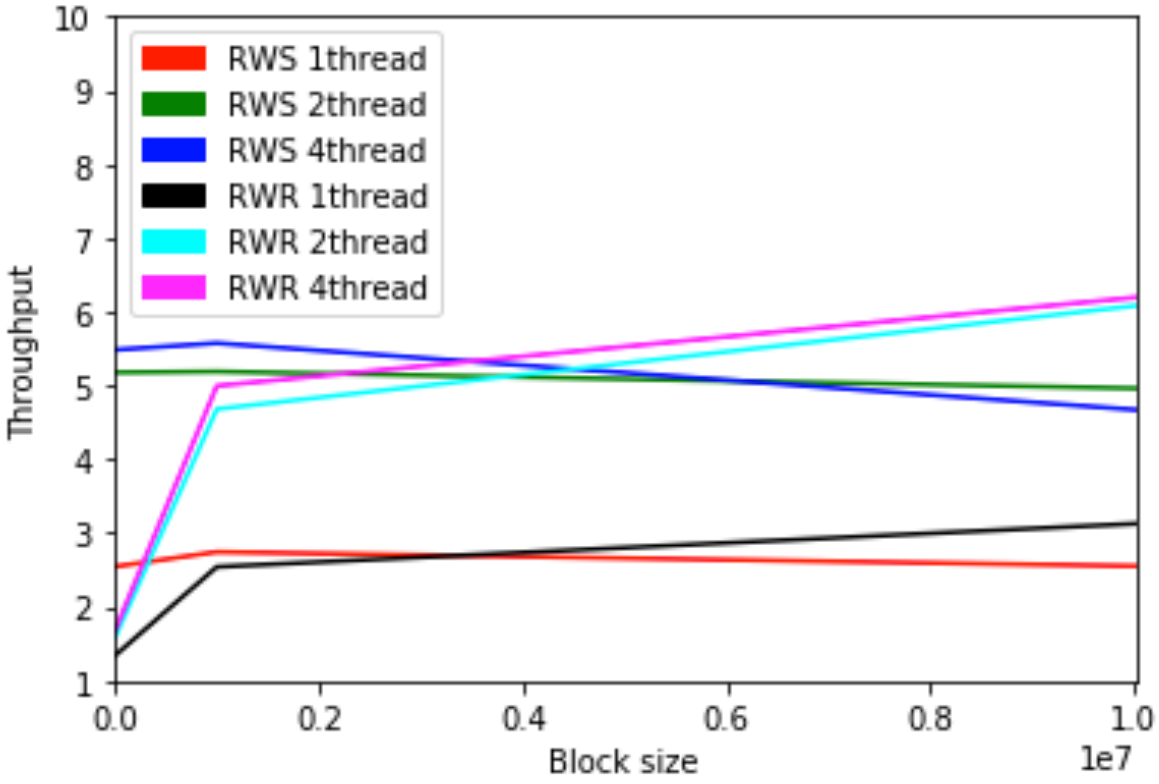

Figure 4

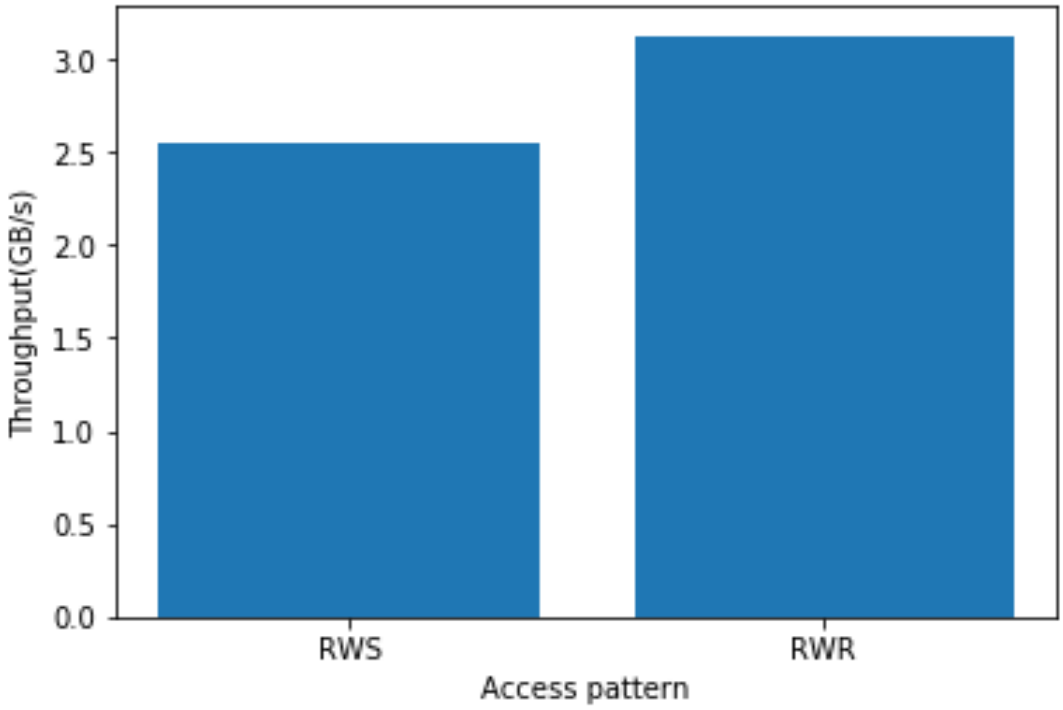

Figure 5

Disk Benchmark:

Hardware information:
Micron 5100 PRO 2.5"480GB,SATA,6Gb/s,3D NAND,7mm,1.5DWPD 2

Computation:
Theoretical throughput = 6Gb/s = 750MB/s
Theoretical IOPS = 74,000
Theoretical latency = 1/IPOS = 0.0135ms

Throughput of my program = total workload / execution time = 10,000,000,000 / execution time
Latency of my program = execution time / number of operations = execution time / 1,000,000
IOPS of my program = 1/latency

Latency of IOZone = execution time / number of operations = execution time / (total workload/block size)
IOPS of IOZone = 1/latency

The throughputs of my program and IOZone are shown on table 4.
The latency of my program and IOZone are shown on table 5.
The IOPS of my program and IOZone are shown on table 6.

Work-load	Concu-rrency	Block Size	MyDiskBench Measured Throughput (MB/sec)	IOZone Measured Throughput (MB/sec)	Theoretical Throughput (MB/sec)	MyDiskBench Efficiency(%)	IOZone Efficien-cy(%)
RS	1	1MB	205.5	183.7	750	27.4	24.5
RS	1	10MB	194.0	194.7	750	25.9	26.0
RS	1	100MB	204.7	193.8	750	27.3	25.8
RS	2	1MB	203.4	157.5	750	27.1	21.0
RS	2	10MB	203.2	163.4	750	27.1	21.8
RS	2	100MB	196.6	134.8	750	26.2	18.0
RS	4	1MB	194.4	160.9	750	26.0	21.5
RS	4	10MB	204.1	183.0	750	27.2	24.4
RS	4	100MB	207.8	124.5	750	27.8	16.6
WS	1	1MB	48.74	232.9	750	6.5	31.1
WS	1	10MB	118.8	237.6	750	15.8	31.7
WS	1	100MB	172.6	250.2	750	23.0	33.4
WS	2	1MB	68.50	186.6	750	9.1	24.9
WS	2	10MB	138.5	260.8	750	18.5	34.8
WS	2	100MB	195.6	191.1	750	26.1	25.5
WS	4	1MB	70.43	252.4	750	9.4	33.7
WS	4	10MB	163.7	276.9	750	21.8	37.0
WS	4	100MB	218.2	259.1	750	29.1	34.5
RR	1	1MB	692.3	3225	750	92.3	430
RR	1	10MB	4696	5637	750	626	752
RR	1	100MB	1635	2554	750	218	341
RR	2	1MB	2421	8398	750	329	1120
RR	2	10MB	5270	8583	750	703	1144
RR	2	100MB	1912	6393	750	255	852
RR	4	1MB	2268	4652	750	302	620
RR	4	10MB	4809	8150	750	641	1087
RR	4	100MB	1890	3002	750	252	400
WR	1	1MB	58.22	224.9	750	7.8	30.0

WR	1	10MB	110.1	248.3	750	14.7	33.1
WR	1	100MB	179.3	251.2	750	23.9	33.5
WR	2	1MB	53.32	224.9	750	7.1	30.0
WR	2	10MB	132.9	243.6	750	17.7	32.5
WR	2	100MB	203.0	232.8	750	27.1	31.0
WR	4	1MB	118.1	260.7	750	15.7	34.8
WR	4	10MB	140.5	271.6	750	18.7	36.2
WR	4	100MB	222.1	196.8	750	29.6	26.2

Table 4

Work-load	**Concu-rrency**	**Block Size**	**MyDiskBench Measured Latency(ms)**	**IOZone Measured Latency (ms)**	**Theoretical Latency (ms)**	**MyDiskBench Efficiency(%)**	**IOZone Efficien-cy(%)**
RR	1	1KB	0.180	0.229	0.0135	7.5	5.9
RR	2	1KB	0.181	0.113	0.0135	7.5	12.0
RR	4	1KB	0.171	0.092	0.0135	7.9	14.7
RR	8	1KB	0.165	0.088	0.0135	8.2	15.3
RR	16	1KB	0.156	0.213	0.0135	8.7	6.3
RR	32	1KB	0.131	0.070	0.0135	10.3	19.3
RR	64	1KB	0.120	0.081	0.0135	11.3	16.7
RR	128	1KB	0.109	0.121	0.0135	12.4	11.2
WR	1	1KB	1.246	1.621	0.0135	1.1	0.8
WR	2	1KB	1.496	0.975	0.0135	0.9	1.4
WR	4	1KB	1.671	1.847	0.0135	0.8	0.7
WR	8	1KB	1.811	0.650	0.0135	0.8	2.1
WR	16	1KB	0.871	3.094	0.0135	1.6	0.4
WR	32	1KB	0.325	0.652	0.0135	4.2	2.1
WR	64	1KB	0.392	0.789	0.0135	3.4	1.7
WR	128	1KB	0.212	4.903	0.0135	6.4	0.3

Table 5

Work-load	Concu-rrency	Block Size	MyDiskBench Measured IOPS	IOZone Measured IOPS	Theoretical IOPS	MyDiskBench Efficiency(%)	IOZone Efficien-cy(%)
RR	1	1KB	5549	4366	74000	7.5	5.9
RR	2	1KB	5512	8850	74000	7.5	12.0
RR	4	1KB	5846	10869	74000	7.9	14.7
RR	8	1KB	6045	11364	74000	8.2	15.3
RR	16	1KB	6403	4695	74000	8.7	6.3
RR	32	1KB	7620	14286	74000	10.3	19.3
RR	64	1KB	8339	12346	74000	11.3	16.7
RR	128	1KB	9147	8264	74000	12.4	11.2
WR	1	1KB	802	617	74000	1.1	0.8
WR	2	1KB	668	1026	74000	0.9	1.4
WR	4	1KB	598	541	74000	0.8	0.7
WR	8	1KB	552	1538	74000	0.8	2.1
WR	16	1KB	1148	323	74000	1.6	0.4
WR	32	1KB	3080	1534	74000	4.2	2.1
WR	64	1KB	2552	1267	74000	3.4	1.7
WR	128	1KB	4719	204.0	74000	6.4	0.3

Table 6

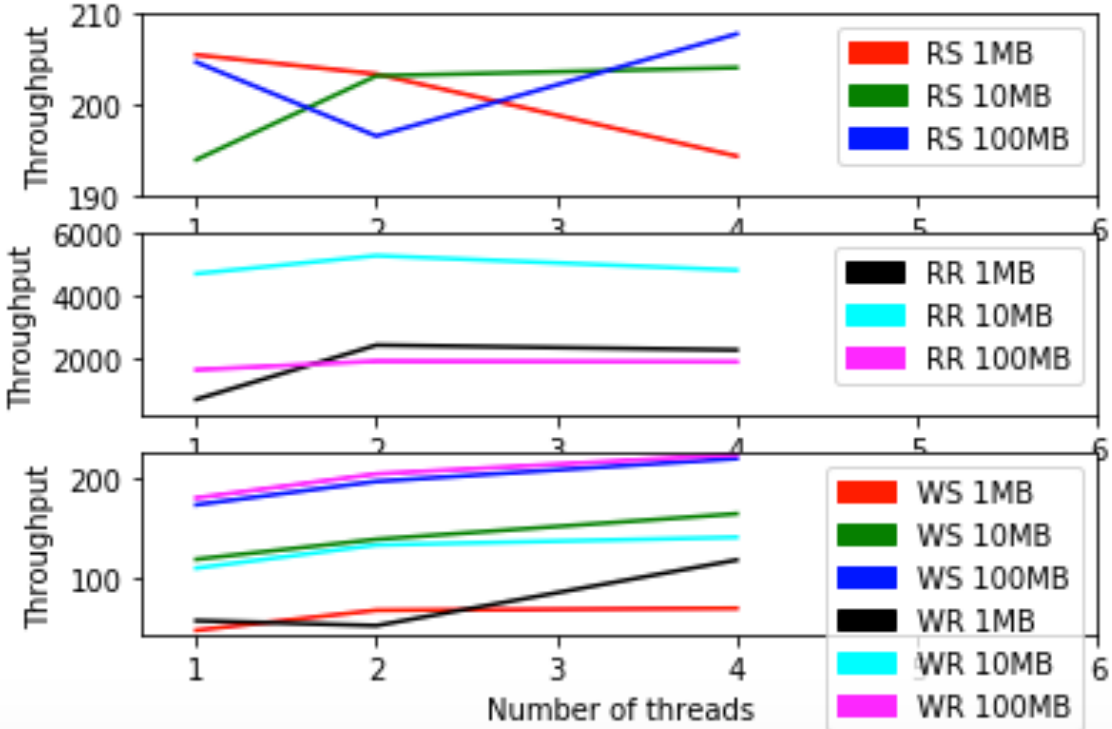

Figure 6

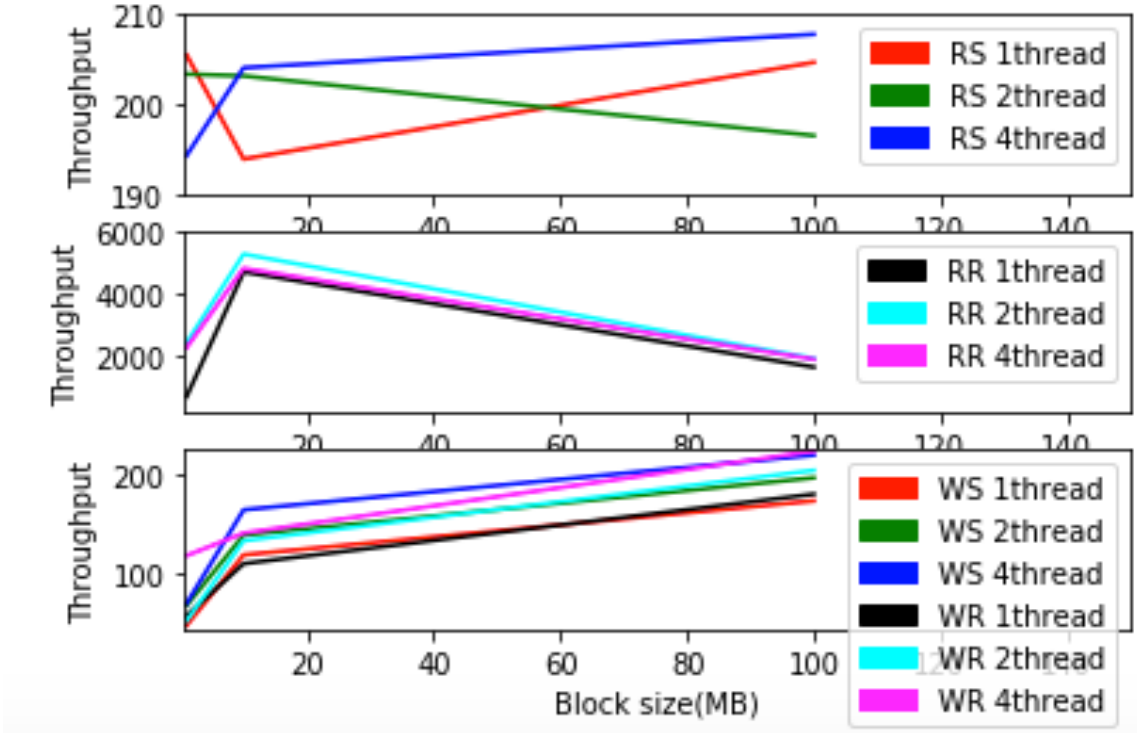

Figure 7

Observations:
1) Figure 6 shows the relationship of throughput and number of threads. From figure 6, we can see that: For read operations(including RS and RR), throughput does not have a stable relationship with number of threads. For write operations(including WS and WR), throughout has a increasing trend relationship with number of threads.
Reason: For this observation, I printed every file pointer by function ftell to see the influence of different threads to the time of reading and writing 10GB data. I found that in RS access pattern different threads often have the same file pointer and in WS access pattern different threads always have different file pointer. Therefore, multiple threads work for writing operation but not work for reading operation.

2) Figure 7 shows the relationship of throughput and block size. From figure 7, we can see that: For RS, throughput does not have a stable relationship with block size. For RR, throughput at 10MB block size is much larger than throughput at 1MB block size and 100MB block size. For writing operations(including WS and WR), throughput increases as block size increasing.

3) Figure 8 shows the relationship of throughput and access pattern when thread number is 1 and block size is 1MB. From figure 8, we can see that throughput(RR)> throughput(RS)> throughput(WR)> throughput(WS).
Reason: The speed of reading operation is faster than the speed of writing operation. The speed random access pattern is faster than the speed of sequential access pattern.

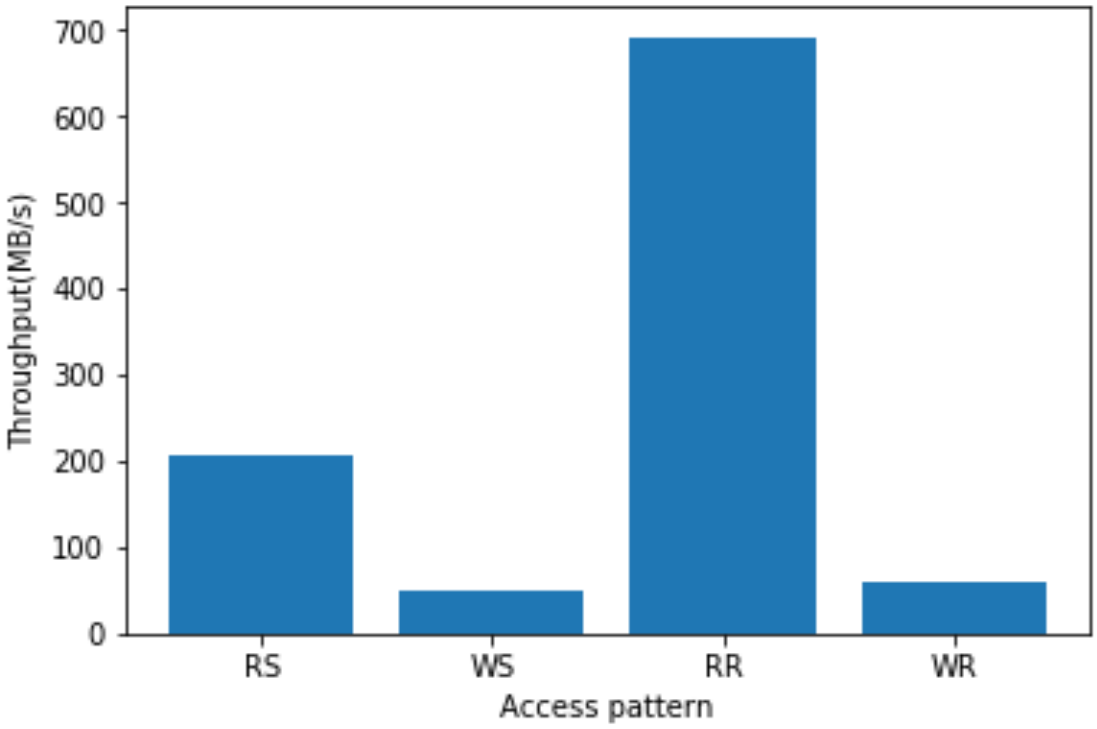

Figure 8

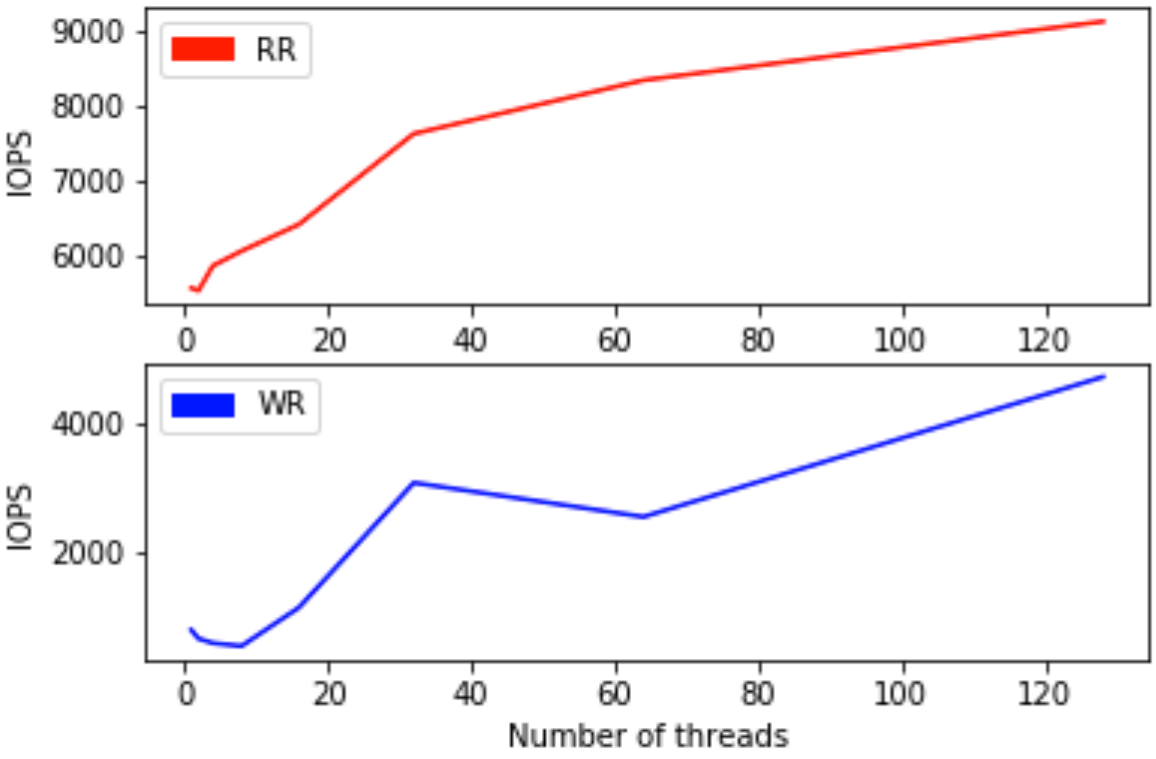

Figure 9

4) Figure 9 shows the relationship of IOPS and number of threads. From figure 9, we can see that: For RR, IOPS decreases at first then increases when number of threads is increasing. For WR, IOPS decreases at first then increases then has another decreasing point then increases straightly when number of threads is increasing.
Reason: IOPS decreases at first because that threads can suppress the speed of other threads when reading or writing. Then as the number of threads increasing, the speed suppressed by other threads will smaller than the speed provided by multiple threads. So the IOPS increases.
The second decreasing point of WR is unusual, which may be caused by disk sudden slow when testing.

Future work:
Some performance of RR is better than theoretical value, which is not correct. I still need to find the reason and fix it.

Network Benchmark:

Computation:
Theoretical throughput = 10Gbps = 10,000Mb/s
Assume that the distance between two compute nodes is less than 100m. So theoretical latency = (distance/speed of light) * 2(round-trip) = (100/300,000,000) * 2 = 0.00067ms

Throughput of my program = total workload/execution time = 100,000,000,000
Latency of my program = execution time/number of round-trip transitions = execution time/1,000,000

The throughputs of my program and iperf are shown on table 7.
The latency of my program and ping are shown on table 8.

Proto-col	Concu-rrency	Block Size	MyNETBench Measured Throughput (Mb/sec)	iperf Measured Throughput (Mb/sec)	Theoretical Throughput (Mb/sec)	MyNETBench Efficiency(%)	iperf Efficien-cy(%)
TCP	1	1KB	64.4	33.5	10000	0.64	0.34
TCP	1	32KB	1471	1280	10000	14.7	12.8
TCP	2	1KB	102.4	62.0	10000	1.02	0.62
TCP	2	32KB	2216	1970	10000	22.2	19.7
TCP	4	1KB	156.7	108	10000	1.57	1.08
TCP	4	32KB	2957	2380	10000	29.6	23.8
TCP	8	1KB	212.9	154	10000	2.13	1.54
TCP	8	32KB	3120	2520	10000	31.2	25.2
UDP	1	1KB	436.9	1.05	10000	4.37	0.01
UDP	1	32KB	4951	1.05	10000	49.5	0.01
UDP	2	1KB	1133	2.10	10000	11.3	0.02
UDP	2	32KB	4460	2.10	10000	44.6	0.02
UDP	4	1KB	1109	4.19	10000	11.1	0.04
UDP	4	32KB	4719	4.19	10000	47.2	0.04
UDP	8	1KB	1144	8.38	10000	11.4	0.08
UDP	8	32KB	4983	8.39	10000	49.8	0.08

Table 7

Proto-col	Concu-rrency	Block Size	MyNETBench Measured Latency(ms)	ping Measured Latency(ms)	Theoretical Latency(ms)	MyNETBench Efficiency(%)	ping Efficien-cy(%)
TCP	1	1B	0.174	0.00775	0.00067	0.39	8.64
TCP	2	1B	0.095	0.00775	0.00067	0.71	8.64
TCP	4	1B	0.049	0.00775	0.00067	1.37	8.64
TCP	8	1B	0.038	0.00775	0.00067	1.76	8.64
UDP	1	1B	0.017	0.00775	0.00067	3.94	8.64
UDP	2	1B	0.017	0.00775	0.00067	3.94	8.64
UDP	4	1B	0.0037	0.00775	0.00067	18.1	8.64
UDP	8	1B	0.0011	0.00775	0.00067	60.9	8.64

Table 8

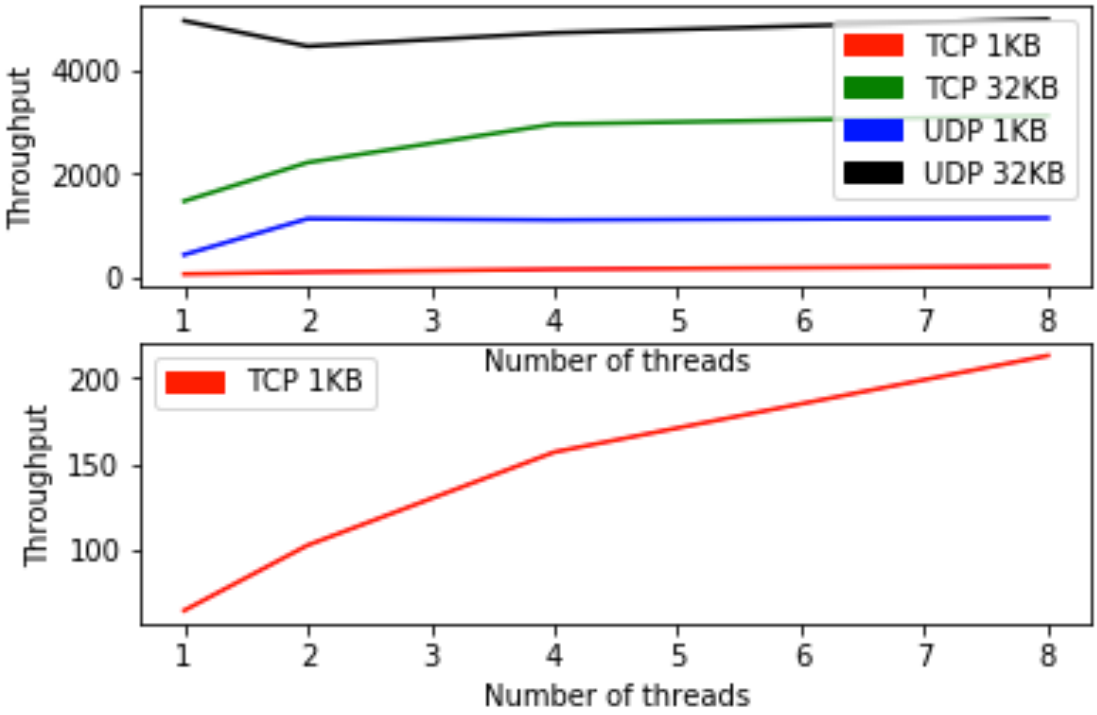

Figure 10

Observations:

Figure 10 shows the relationship of throughput and number of threads. Because the throughput of block size 1KB of TCP protocol is so much small compared to other throughputs, I drew it at the bottom of the graph contains all messages. From figure 10, we can see that:

1) At TCP protocol, throughput increases as number of threads increasing. At UDP protocol, throughput does not have a stable relationship to number of threads.

Reason: TCP protocol is based on connection. The throughput will be increased if multiple connection is created by multiple threads. UDP protocol is based on data transition. The throughout may not influenced by multiple threads.

2) The throughputs of 32KB block size are always larger than the throughputs of 1KB block size.

Reason: With a given transition workload, 32KB block size need less times of transition than 1KB block size.

3) When the block size is the same, throughput of UDP protocol is larger than the throughput of TCP protocol.

Reason: The data transition of TCP is in order. The data transition of UDP is not in order.