Cloud Computing Programming Assignment 1

Performance Measure

**CPU Benchmark:**

This CPU benchmarking experiment we ran on the m1.medium KVM instance.

I have done the benchmarking using FLOPS and IOPS. Following graph shows the performance results according to the operations:

Here, I performed the benchmarking for 1,2,4,8 threads.

As per the observations, as the thread count increases the Giga FLOPS increases. Also, as the thread increases it will increase the IOPS. It is because the concurrent processing makes it increase the number of FLOPS or IOPS to complete.

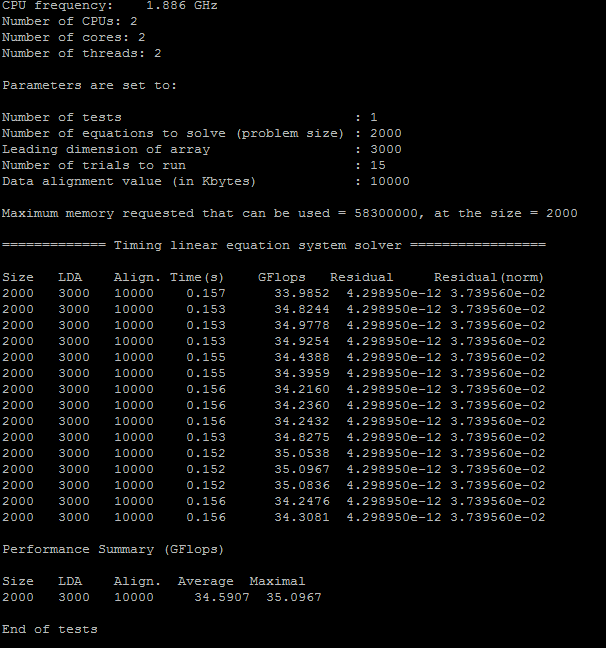
**LinPack Benchmark:**

We ran the linpack benchmark. Following are the results regarding it:

Commands to run:

chmod 700 xlinpack\_xeon64

./ xlinpack\_xeon64



Theoretical performance = number of cores \* clock cycle \* instructions per cycle

= 2.3 \* 2 \* 16

= 73.6

Efficiency of our benchmark = (FLOPS for 1 thread / theoretical peak performance) \* 100

= (9.6/73.6) \* 100

= 13.043 %

Efficiency of Linpack = (Linpack peak performance / theoretical peak performance) \* 100

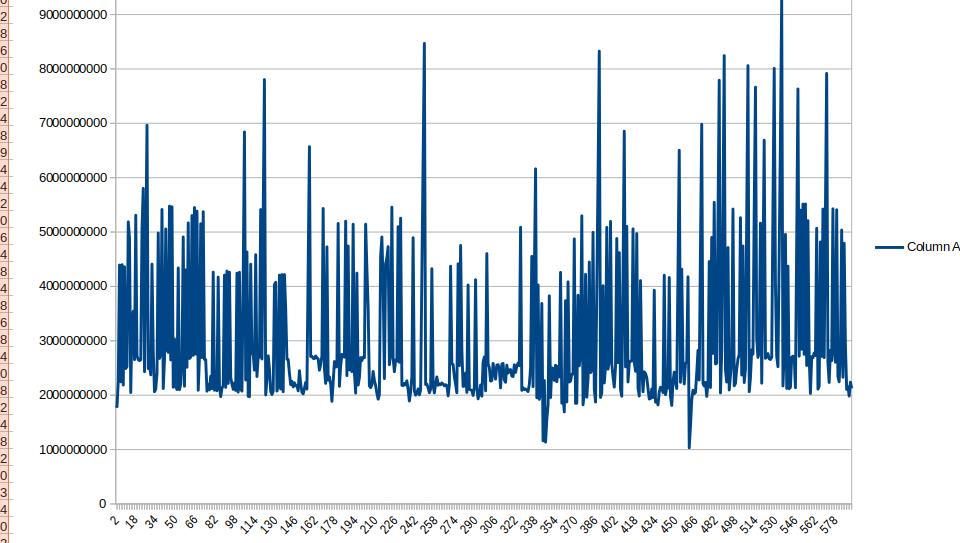
= (30.601/73.6) \* 100

= 41.57 %

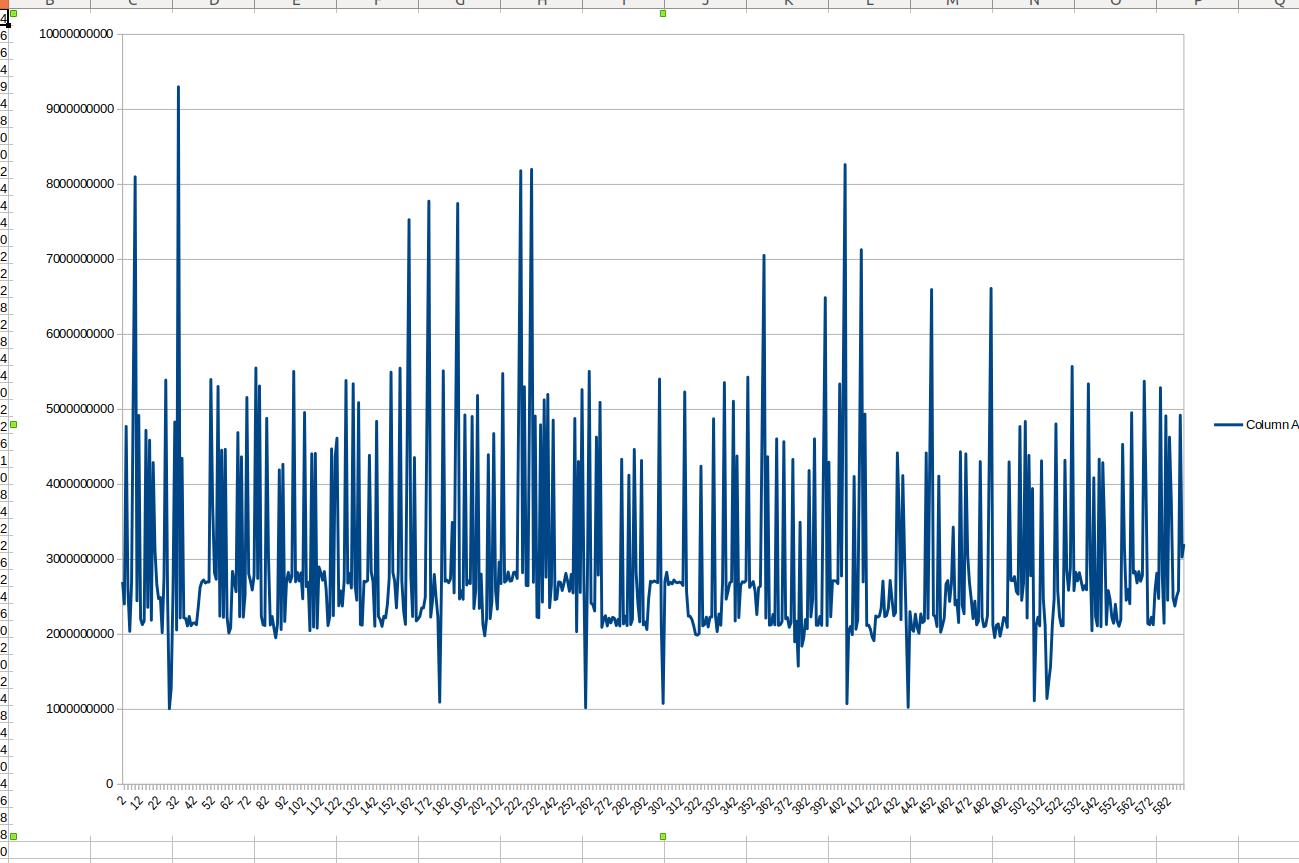
**Running the benchmark for 10 min:**

I performed the FLOPS and IOPS operations for 10 min and plotted the graph as bellow:

1] FLOPS:



2] IOPS:



**Memory Benchmark:**

This memory benchmark we executed on baremetal instance.

We have done the benchmarking using various block size and threads. Following graph shows the performance results according to the operations:

**Sequential Read Write:**

1. **Throughput:**
2. **Latency:**

**Sequential Write:**

1. **Throughput:**
2. **Latency:**

**Random Write:**

1. **Throughput:**
2. **Latency:**

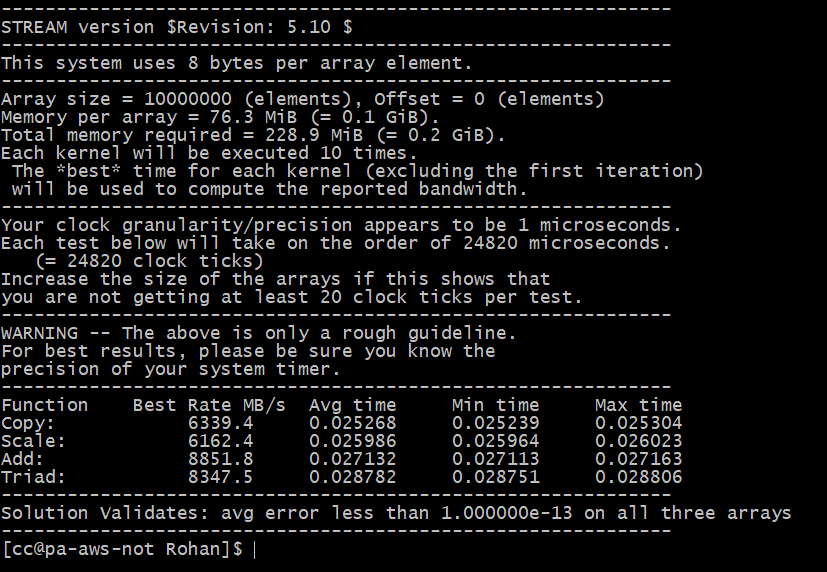
**Observation:**

There is an increase in throughput when the thread count increases. As concurrency increases the throughput.

There is decrease in latency when thread count increases.

There is an increase in throughput when there is increase in block size.

**Stream benchmark:**



**Disk Benchmark:**

This disk benchmark we executed on baremetal instance.

We have done the benchmarking using various block size and threads. Following graph shows the performance results according to the operations:

**Sequential Read Write:**

1. **Throughput:**
2. **Latency:**

**Sequential Read:**

1. **Throughput:**
2. **Latency:**

**Random Read:**

1. **Throughput:**
2. **Latency:**

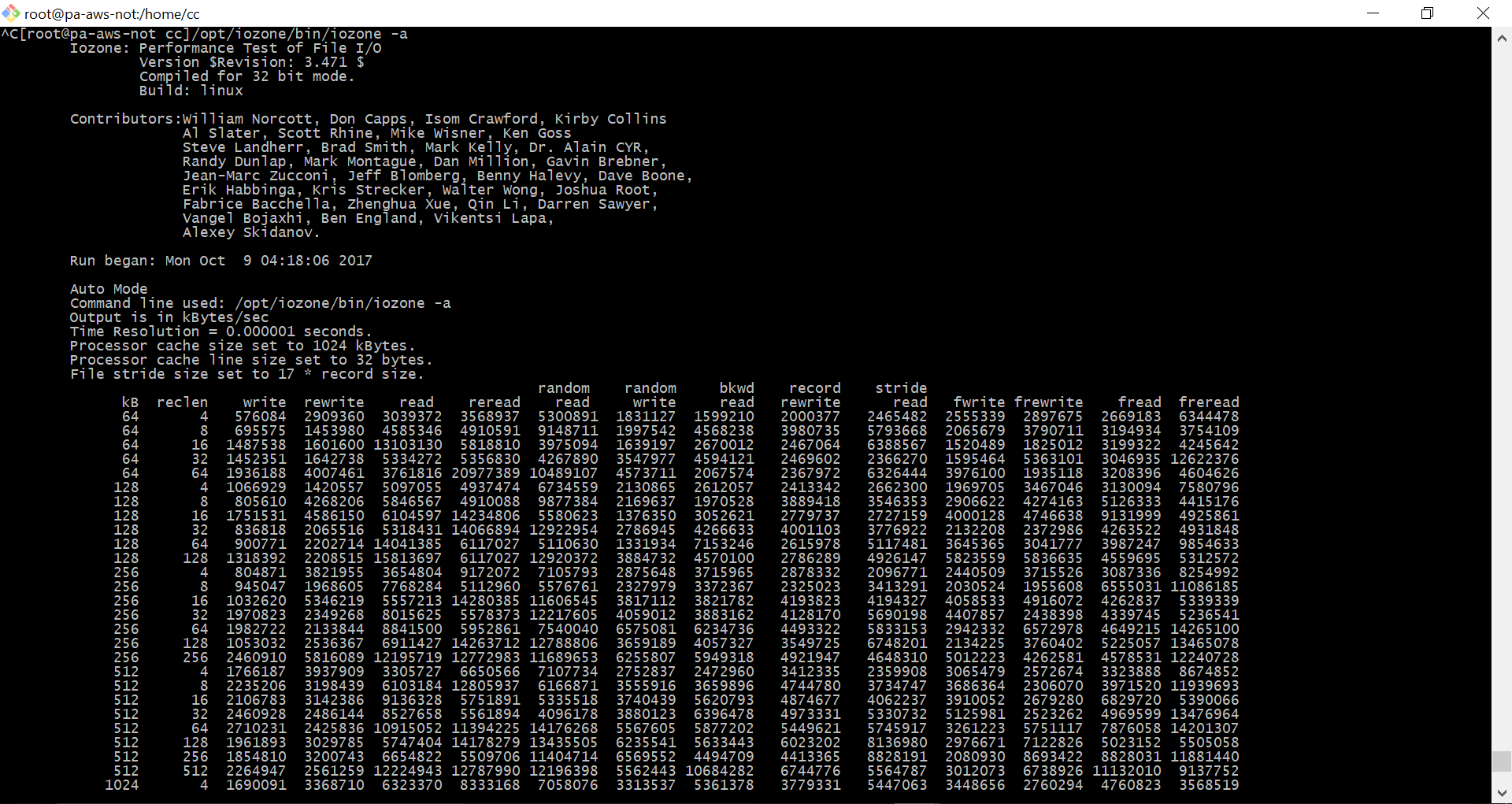
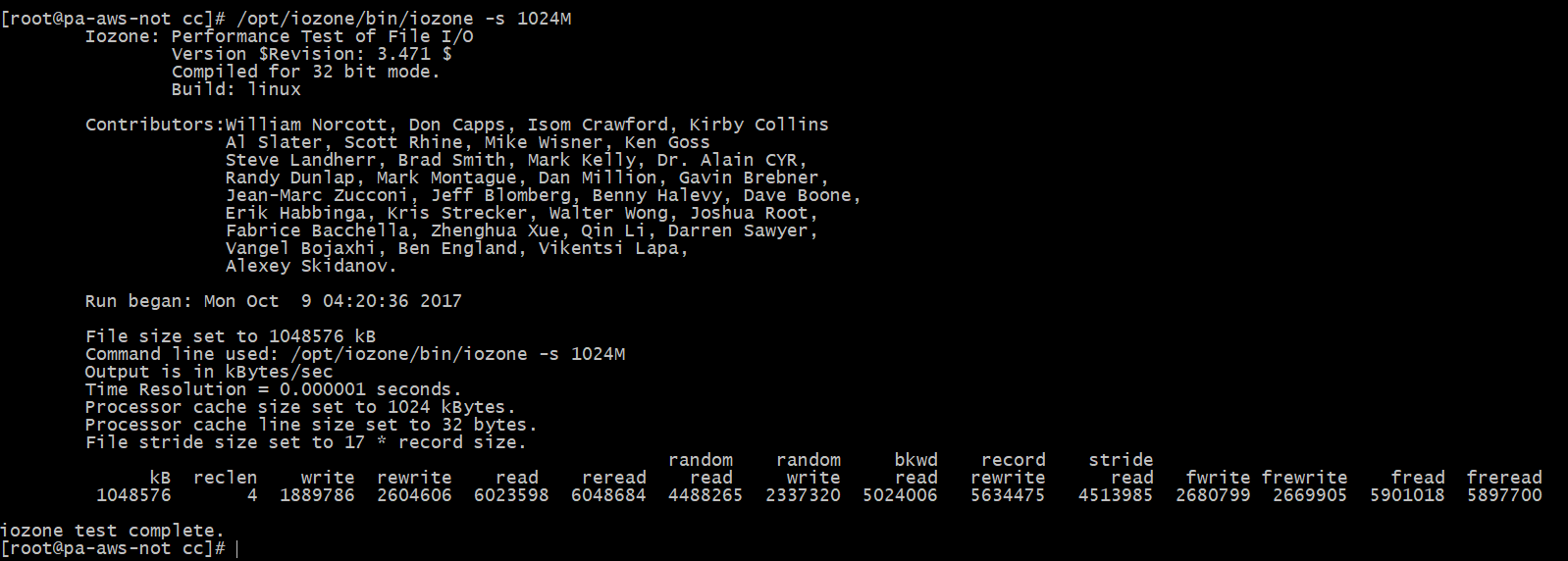
**Observation:**

There is slightly decrease in throughput when the thread count increases. It is due to overheads of concurrency.

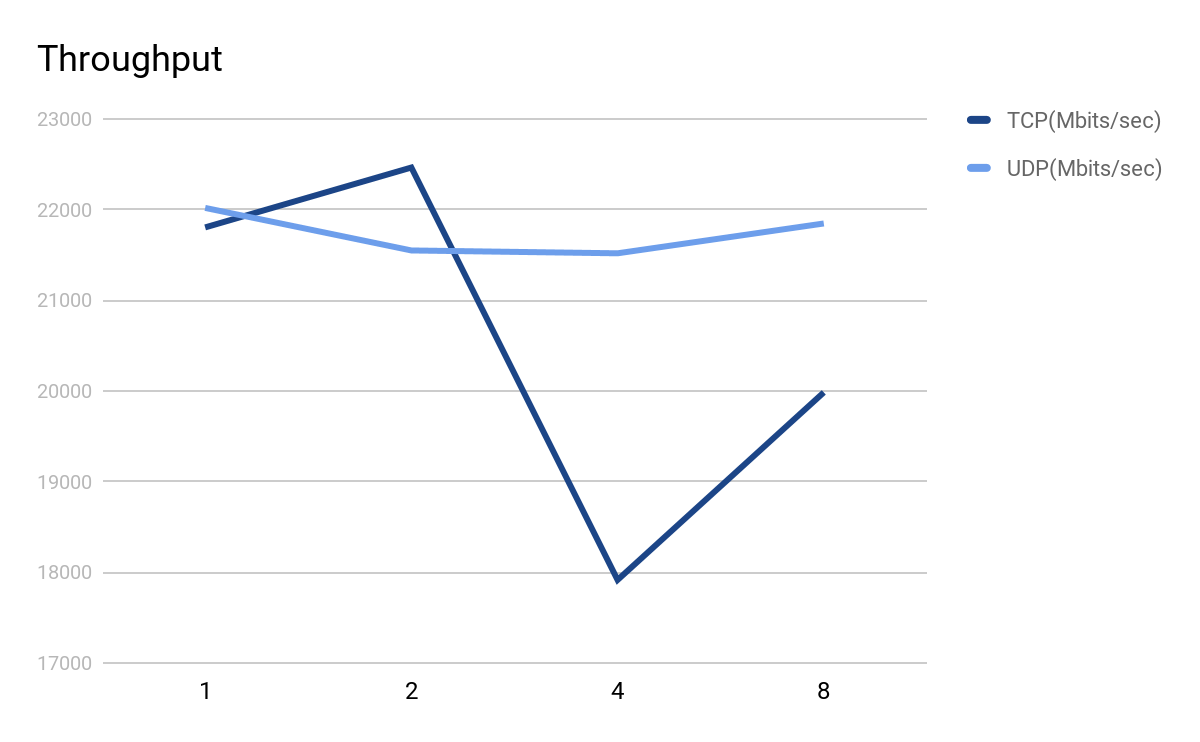
There is an increase in latency when the thread count increases.

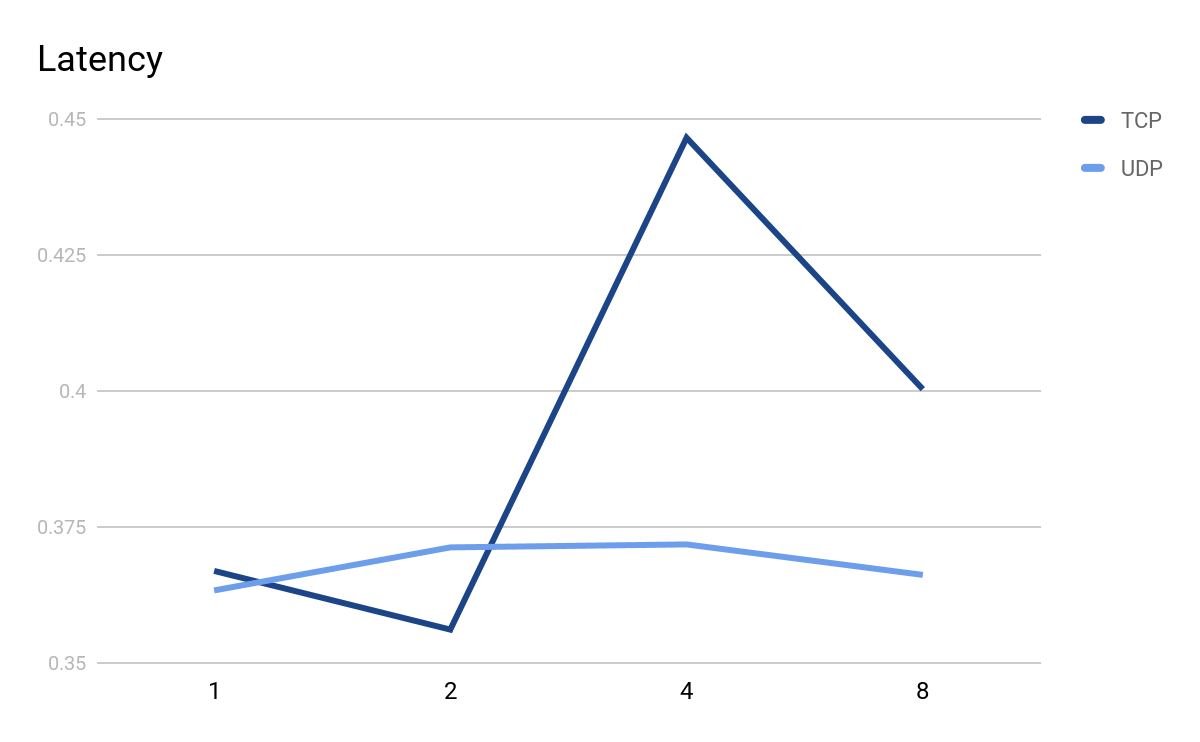
According to results, disk we are evaluating is the HDD.

**IOZONE benchmark:**



* ***Network Benchmark*** 
  + These programs have been executed on a baremetal compute node
  + For the networking benchmark we are using the baremetal compute node
  + The data packet size is fixed at 4GB



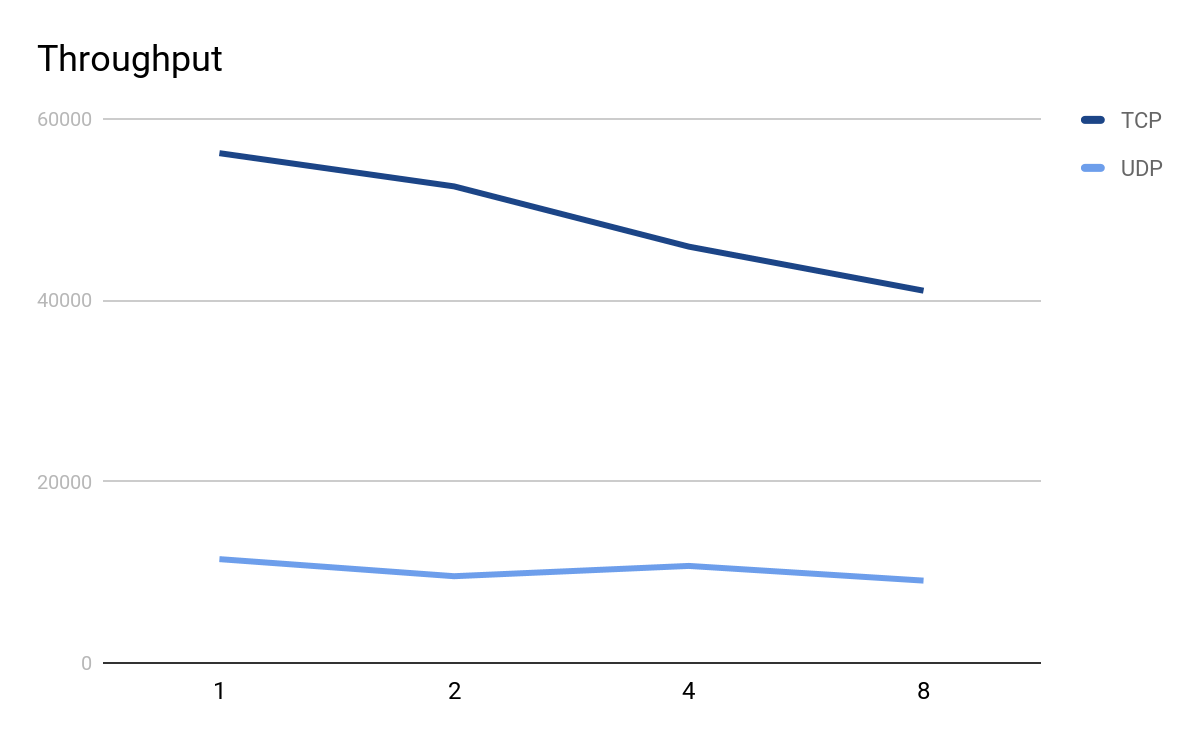
The y axis represents the throughput and x axis represents the number of threads.

Conclusion:

We can see that the performance increased when we moved from 1 thread to 2 threads. However, it decreased on increasing the threads to 4 the reason for this could be the overhead of running the threads as well as other processes running simultaneously.

Incase of UDP the data transfer is stable and does not vary much because of the fact that UDP just sends the packets and the overhead of receiving ACK and resending packets on loss is not present.

* Iperf benchmark:



On comparison we can see a lot of difference in the benchmark throughputs and our benchmark gives a throughput equal to 50% of that of iperf for TCP and 50% more than iperf for UDP.

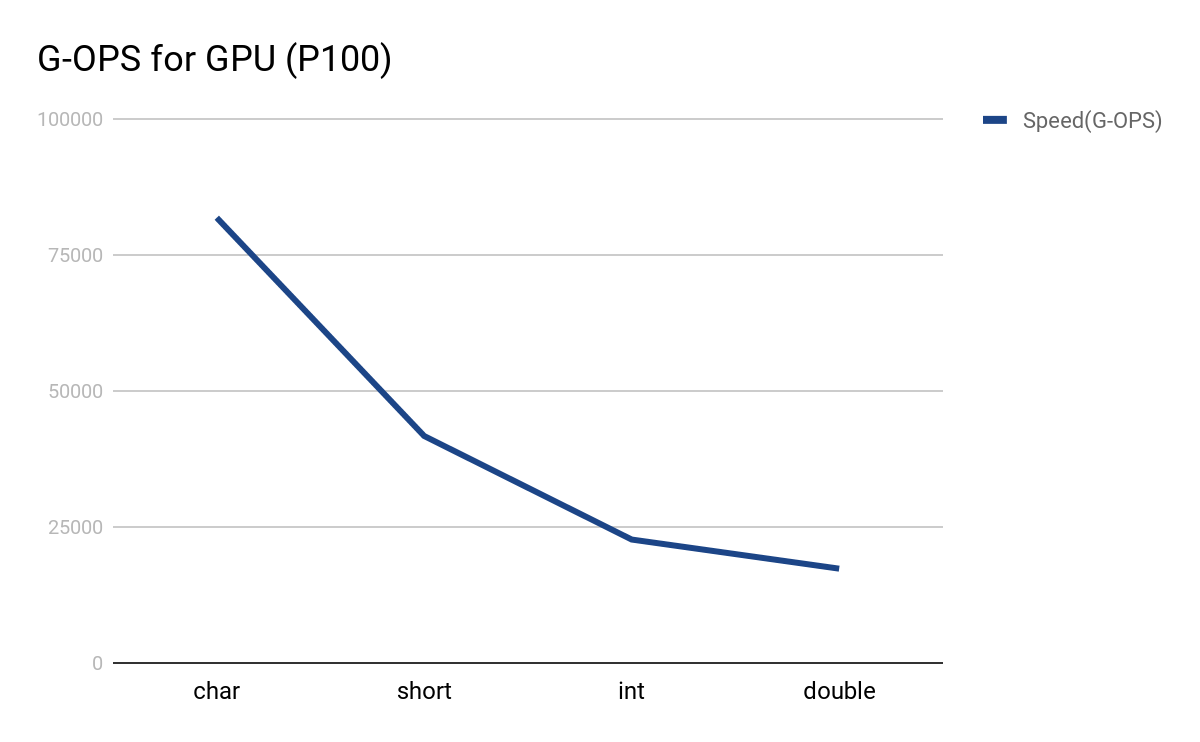
References:

1.<http://www.slashroot.in/iperf-how-test-network-speedperformancebandwidth>

2.<https://iperf.fr/iperf-doc.php#doc>

3.<http://linuxthrill.blogspot.com/2016/04/iperf-test-network-throughput-delay.html>

* ***GPU Benchmarking***
  + Here are the test results of GPU Speeds for double, int ,short and char values.



Conclusion: As the floating operation bit size increases, the operations per second decrease as the GPU has to deal with larger data per operation.

This speed can be increased by utilizing the full potential of the GPU.

References:

1. <https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf>
2. <https://www.pdc.kth.se/resources/computers/historical-computers/zorn/how-to/how-to-compile-and-run-a-simple-cuda-hello-world>
3. <https://devblogs.nvidia.com/parallelforall/easy-introduction-cuda-c-and-c/>