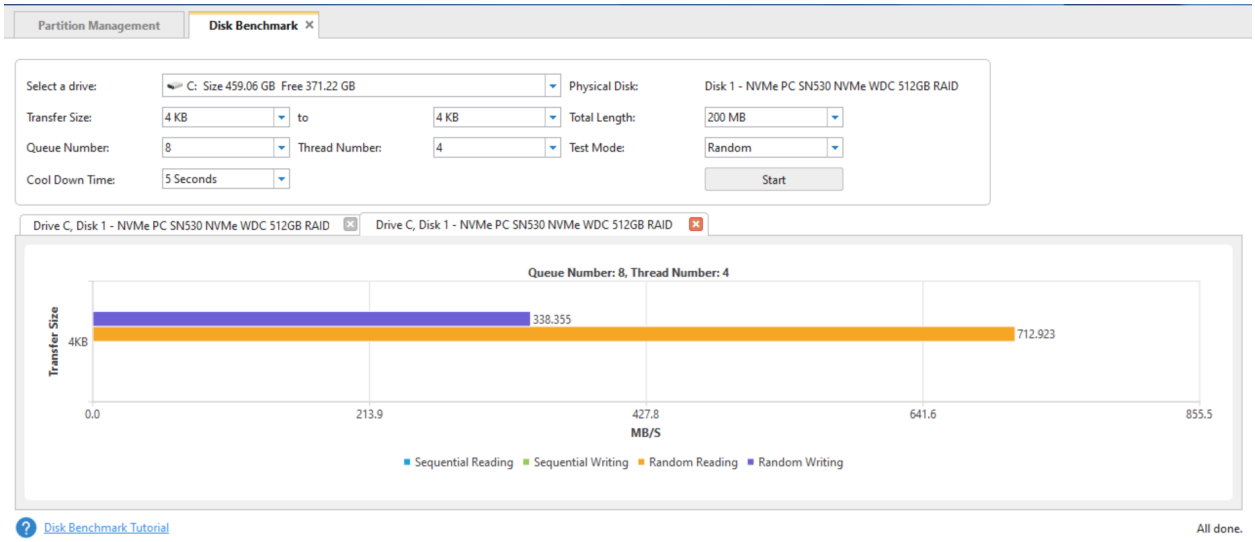
Part 0

From the documentation of various time functions available in C/C++, **clock_gettime** has one of the highest precisions (1 nanosecond). Others like **gettimeofday** and **getrusage** have the precision of 1 microsecond. Since the latency numbers we want to measure are in nanoseconds, clock_gettime is the best choice.

The lowest time we measure with clock_gettime() was 50 ns. This was the time measured by executing a single line of addition (and printing the result outside the timing block of code).

Part 1

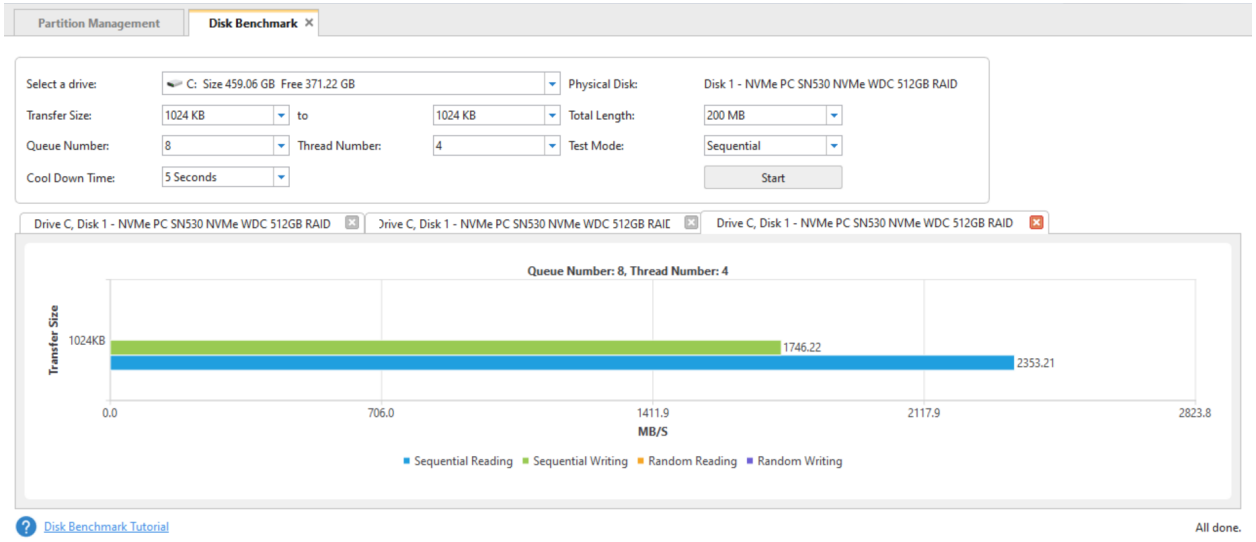
1. L1 cache reference: We measured the reference latency to be 2 ns.
2. Branch misprediction: We measure the latency to be around 20 ns
3. L2 cache reference: We measured the reference latency to be 6 ns.
4. Mutex lock/unlock: Locking time was typically in the range of 40 ns. The unlocking time was around 25 ns.
5. Compress 1k bytes with zippy: We used zlib library of C to perform the compression. Our implementation took 24 us for compressing a buffer of 1000 bytes.
6. Sending 1k bytes over 1 Gbps network: We sent 1k bytes from one CSL machine to the other. Our round trip time was about 160 us so the sending time was about 80 us. Measurements are taken using the UDP server/client library by setting the input packet size to 1k bytes.
7. Main memory reference: The time measured was 15 nanoseconds.
8. Read 4K randomly from SSD: For this, we used a profiling software tool to perform random reads. This gave us the random read bandwidth from which we calculate the 4K read time to be 5.6 us.



9. Round trip time within the same datacenter: We reserved two Google Cloud instances in the same physical zone and pinged one from the other. The round trip time was 1.5 ms.

```
aditisingh2297@instance-1:~$ ping 34.125.171.120
PING 34.125.171.120 (34.125.171.120) 56(84) bytes of data.
64 bytes from 34.125.171.120: icmp_seq=1 ttl=61 time=1.52 ms
64 bytes from 34.125.171.120: icmp_seq=2 ttl=61 time=1.56 ms
64 bytes from 34.125.171.120: icmp_seq=3 ttl=61 time=1.64 ms
64 bytes from 34.125.171.120: icmp_seq=4 ttl=61 time=1.81 ms
64 bytes from 34.125.171.120: icmp_seq=5 ttl=61 time=1.72 ms
64 bytes from 34.125.171.120: icmp_seq=6 ttl=61 time=1.75 ms
64 bytes from 34.125.171.120: icmp_seq=7 ttl=61 time=1.49 ms
64 bytes from 34.125.171.120: icmp_seq=8 ttl=61 time=1.75 ms
64 bytes from 34.125.171.120: icmp_seq=9 ttl=61 time=1.85 ms
64 bytes from 34.125.171.120: icmp_seq=10 ttl=61 time=1.43 ms
^C
--- 34.125.171.120 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9017ms
rtt min/avg/max/mdev = 1.428/1.649/1.849/0.138 ms
```

10. Read 1 MB sequentially from SSD: We used the same tool as in (7) and got a latency of 400 us.



11. Read 1MB sequentially from memory: Time measured was 700 us.
12. Send packet to Netherlands and back. We pinged a website hosted in Netherland from our local machine and the round trip time varied between 120 ms to 200 ms.

```
singh273@AditiSingh-PC:~$ ping government.nl
PING government.nl (178.22.85.8) 56(84) bytes of data.
64 bytes from www.rijksoverheid.nl (178.22.85.8): icmp_seq=1 ttl=44 time=118 ms
64 bytes from www.rijksoverheid.nl (178.22.85.8): icmp_seq=2 ttl=44 time=506 ms
64 bytes from www.rijksoverheid.nl (178.22.85.8): icmp_seq=3 ttl=44 time=224 ms
64 bytes from www.rijksoverheid.nl (178.22.85.8): icmp_seq=4 ttl=44 time=120 ms
64 bytes from www.rijksoverheid.nl (178.22.85.8): icmp_seq=5 ttl=44 time=118 ms
64 bytes from www.rijksoverheid.nl (178.22.85.8): icmp_seq=6 ttl=44 time=186 ms
^C
--- government.nl ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5007ms
rtt min/avg/max/mdev = 117.716/211.985/505.734/137.347 ms
```

Part 2: UDP

Overhead of sending a message:

Measured between send message call in client to sendto function call in udp layer (includes structure packing).

Overhead of sending a message	**Without compiler optimisation**		**With compiler optimisation**	
	Same machine	**Different machine**	**Same machine**	**Different machine**
Small message size - 24B	*120 - 180ns*	*300 - 600ns*	110 - 170ns	300 - 600ns
Large message size - 65500B	15 - 20us	40 - 90us	13 - 20us	40 - 90us

With compiler optimisation, saw almost the same results due to differences in load in the CSL machine.

Difference in packing time observed in different machines due to load throttling and due to CPU's running at different clock frequency even though the max is configured at 3.2GHz

[spartacus@snares-06] (173)$ client localhost 8004
Packing time : 146.000000 ns
Sending hello world message to server
[spartacus@snares-06] (174)$ lscpu
Model name: Intel(R) Core(TM) i5-4570 CPU @ 3.20GHz
Stepping: 3
CPU MHz: 3213.603
CPU max MHz: 3600.0000

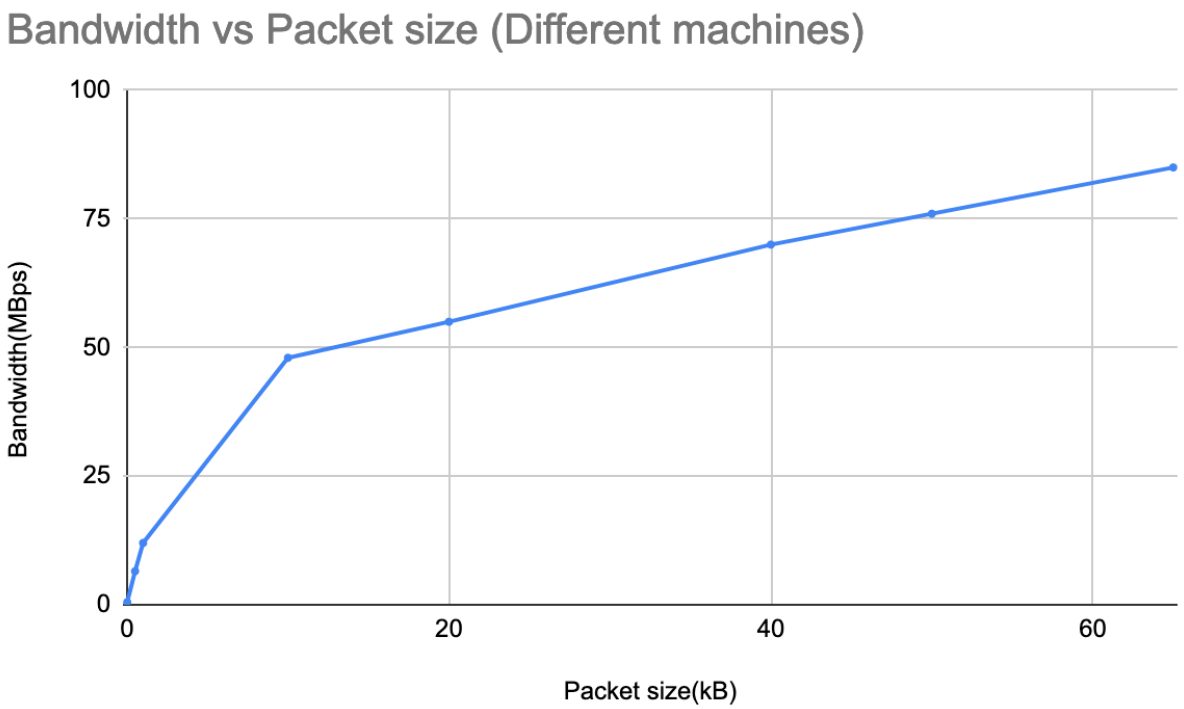
spartacus@royal-10] (53)$ ./client snares-06.cs.wisc.edu 8004
Packing time : 423.000000 ns
Sending hello world message to server
[spartacus@royal-10] (52)$ lscpu
Model name: Intel(R) Core(TM) i5-4570 CPU @ 3.20GHz
Stepping: 3
CPU MHz: 1320.771
CPU max MHz: 3600.0000

Round Trip Transmission:

Round trip time	Without compiler optimisation		With compiler optimisation	
	Same machine	**Different machine**	**Same machine**	**Different machine**
Small message size - 24B	40 - 90us	170 - 250us	40 - 87us	170 - 250us
Large message size - 65500B	15 - 20us	0.8 - 1ms	13 - 19us	0.8 - 0.9ms

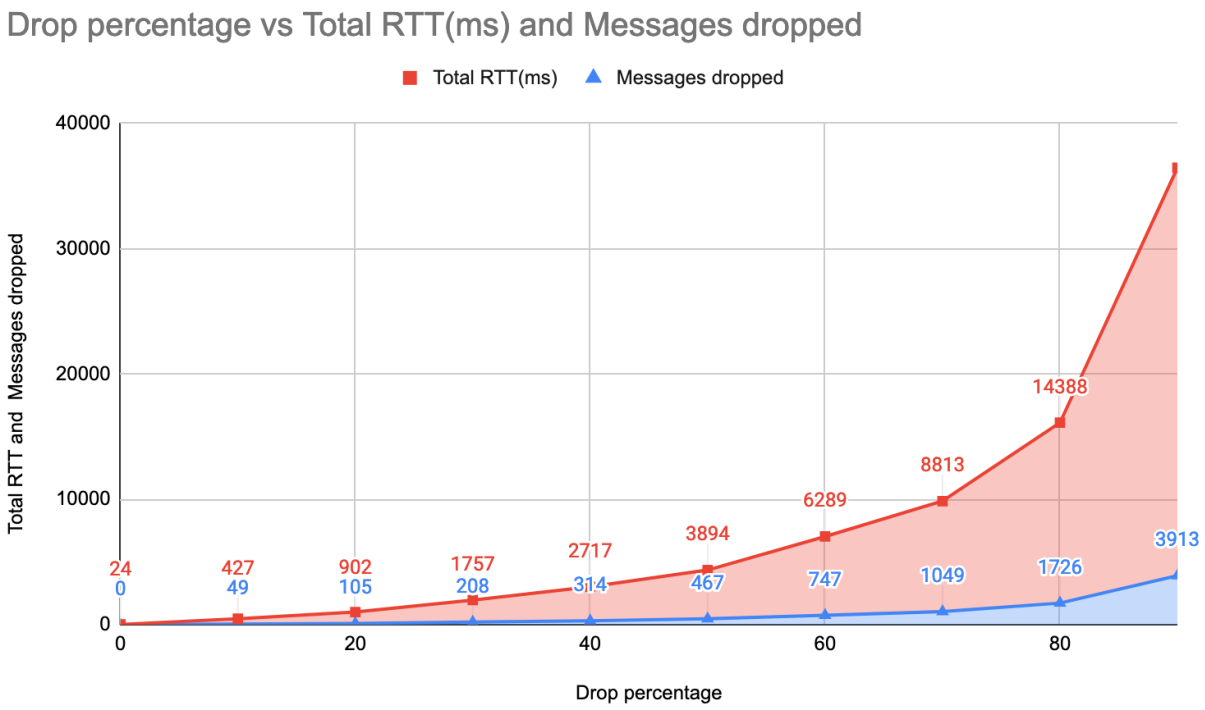
Bandwidth:

Bandwidth	Without compiler optimisation		With compiler optimisation	
	Same machine	**Different machine**	**Same machine**	**Different machine**
Small message size - 24B	2 MB/s	350 - 400 KB/s	1.9 MB/s	340 - 400 KB/s
Large message size - 65500B	1.8 - 2.5 GB/s	70 - 90 MB/s	1.9 - 2.5 GB/s	72 - 86 MB/s



Reliability:

Maximum bandwidth - timeout = 1.5-2x the RTT



Part 3

Grpc

1. overhead of marshaling a message:

Run each Serialization/Deserialization 500 times

type	Time (ms)	Time (ms) -O3
int	272.21	258.51
double	269.36	256.14
string	272.77	264.13
complex structure	280.74	255.51

2. round-trip time for a small message, two Google cloud virtual machines, one in Las Vegas, one in Taiwan.

```
message HelloRequest {
  string name = 1;
}
message HelloReply {
  string message = 1;
}
```

client/server on the same machine?	Time (ms)
Yes (Las Vegas)	3.07
Taiwan -> Las Vegas	405.05

3. Is the first round trip much slower than subsequent ones?

   Round trip between Taiwan and Las Vegas:

   Yes, the first round trip is slower.

No.	Time (ms)	Time (ms) (-O3)
1	405.98	403.40
2	400.80	397.27
3	400.67	399.20
4	400.42	399.33
5	400.7	396.73

4. Bandwidth when sending from Taiwan to Las Vegas (client streaming):

File size (MB)	Time (s)	Bandwidth (MB/s)
8.72	1.77	4.93
25	2.78	8.99
66	4.71	14.01
129	7.98	16.17
185	9.73	19.01
310	15.34	20.21
545	26.8	20.34

Thrift

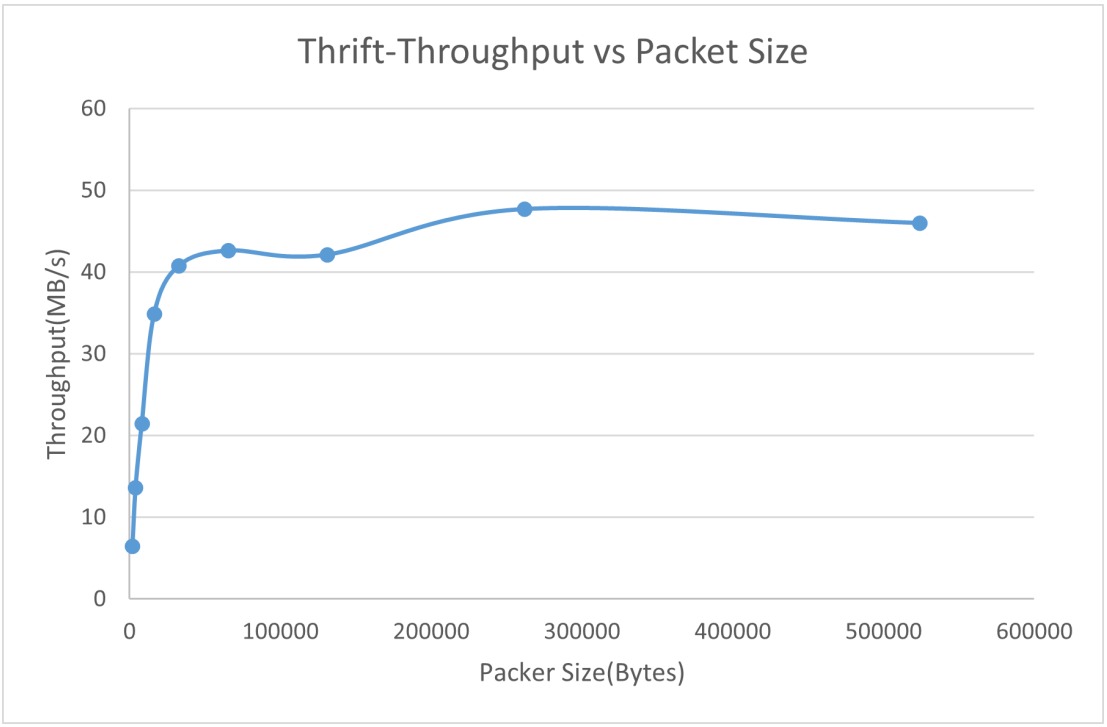
	Int message	Double message	String message size 16 - 512	String message size 1024	String message size 2048	String message size 4096	String message size 8192	String message size 16K	String message size 32K	Complex
No optimization	2.27 us	2.27	2.4 us	2.4	3.3	4.15	5.1	9.8	19.72	3.2
With optimization	0.52 us	0.59	0.6	0.69	1.4	2.4	3.8	7.3	18.8	0.9

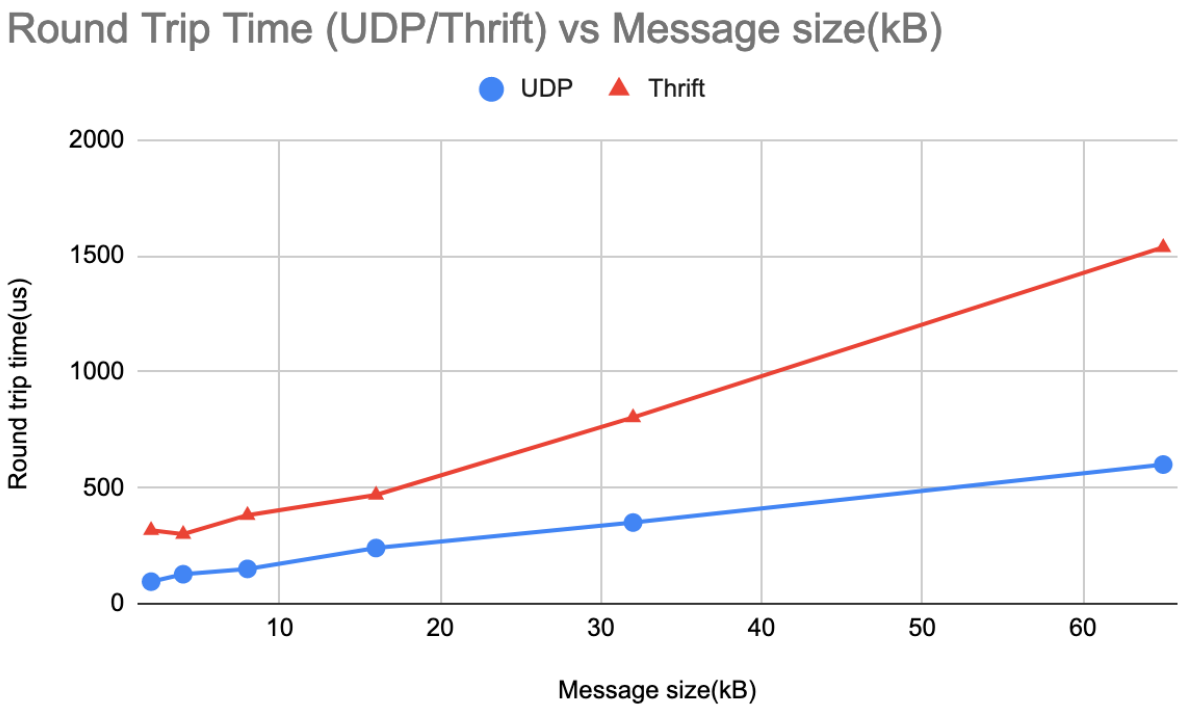
	Int message	Double message	String message size 16 - 512	String message size 1024	String message size 2048	String message size 4096	String message size 8192	String message size 16K	String message size 32K	Complex	Complex Large (16KB string + 3 types lists)
No optimization	2.27 us	2.27	2.4 us	2.4	3.3	4.15	5.1	9.8	19.72	3.2	82
With optimization	0.52 us	0.59	0.6	0.69	1.4	2.4	3.8	7.3	18.8	0.9	19

RTT	1st	2nd	3rd	Avg later
single	220 (190-270)	68	65	50
two	250 (240 - 270)	75	67	65

Ser/Des	Int Msg	Double Msg		Complex Msg	
No optimization	< 0.1 us	0.1 us	0.3 us	1.1 us	1.3 us
With optimization	< 0.1 us	0.1 us	0.2 us	0.2 us	0.3 us

BandWidth 50MB remote and 100MB local


Thrift-Throughput vs Packet Size
Throughput(MB/s)
60
50
40
30
20
10
0
0
100000
200000
300000
400000
500000
600000
Packer Size(Bytes)


Round Trip Time (UDP/Thrift) vs Message size(kB)
UDP
Thrift
Round trip time(us)
2000
1500
1000
500
0
10
20
30
40
50
60
Message size(kB)