RTT-measure

Gongyi Shi

October 7, 2023

Introduction

I perform an RTT evaluation via CloudLab using two m510 nodes on the same stack in Ubuntu.

Background and Summary

1. For each measurement approach (`ping, RDMA READ, RDMA SEND, and DPDK echo`), what hardware or software component on the server generates a response to each request?
   (a) `ping` is implemented using ICMP Echo Request, which is handled by the OS's networking stack.
   (b) `RDMA READ` and `RDMA SEND` access the remote memory directly without involving the remote OS or CPU. They are handled by the RDMA-capable Network Interface Card(NIC) on the client/server side respectively.
   (c) `DPDK` is implemented in user spaces and bypasses the kernel(OS), and thus the packets are handled and processed by user applications.
2. What was the average RTT per packet I observed using each tool?
   (a) `ping`: 29.141 usec
   (b) `DPDK`: 11.576 usec
   (c) `RDMA READ`: 3.40 usec
   (d) `RDMA SEND`: 3.72 usec
3. What software or hardware differences are responsible for the differences in RTTs?
   `RDMA`s are the fastest due to their bypassing of CPU and kernel overheads; `DPDK` is slower because it involves the process of CPU while still bypassing the kernel overhead, and `ping` is the slowest because it also involves kernel.
4. Some of the tools also reported tail and/or max RTT. What are some factors that could cause these tail metrics to be significantly higher than the averages?
   There are lots of factors. For instance, re-transmission due to packet loss; network congestion due to high amount of requests, and race condition when multiple clients try to access the same data.

5. Suppose I wrote a benchmark to measure the maximum achievable throughput with each approach (using at most 1 core on client and server). Which tools do you think would provide the lowest and highest throughputs, and why?

   Still, `RDMA`s > `DPDK` > `ping` due to the same reasons as Q3. Single core introduces multitasking: potential scheduling overhead!

6. How was programming with `DPDK`'s APIs different from programming with Linux's sockets API?

   - `DPDK`: Hardware-aware (DPDK-compatible NIC drivers), low-level, user-handled memory and packet management.
   - Linux's socket: Portable, abstracted, OS-handled memory and packet management.

Results

Measure RTTs with Ping

```
$ gongyi@node-0:~$ sudo ping -f -c 1000000 10.10.1.2
--- 10.10.1.2 ping statistics ---
1000000 packets transmitted, 1000000 received, 0% packet loss, time 29141ms rtt min/avg/max/mdev = 0.011/0.016/0.462/0.007 ms,
ipg/ewma 0.029/0.017 ms
```

Measure RTTs with RDMA

RDMA Read

```
$ gongyi@node-0:~$ ib_read_lat --iters=1000000 --size=64 10.10.1.2
---------------------------------------------------------------------------------------
                    RDMA_Read Latency Test
 Dual-port       : OFF          Device         : mlx4_0
 Number of qps   : 1            Transport type : IB
 Connection type : RC           Using SRQ      : OFF
 TX depth        : 1
 Mtu             : 1024[B]
 Link type       : Ethernet
 GID index       : 2
 Outstand reads  : 16
 rdma_cm QPs     : OFF
 Data ex. method : Ethernet
---------------------------------------------------------------------------------------
```

```
 local address: LID 0000 QPN 0x022c PSN 0x27abb6 OUT 0x10 RKey 0x10010100 VAddr 0x0055a6cd55f000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:128:110:218:30
 remote address: LID 0000 QPN 0x022c PSN 0x38cfe6 OUT 0x10 RKey 0x10010100 VAddr 0x00555d91913000
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:128:110:218:17
---------------------------------------------------------------------------------------
 #bytes #iterations  t_min[usec]   t_max[usec] t_typical[usec]  t_avg[usec]   t_stdev[usec]  99% percentile[usec] 99.9% percentile[usec]
Conflicting CPU frequency values detected: 798.322000 != 1524.376000. CPU Frequency is not max.
Conflicting CPU frequency values detected: 799.386000 != 1938.263000. CPU Frequency is not max.
 64      1000000         3.17          386.75      3.35           3.40           1.99          3.73                8.38
---------------------------------------------------------------------------------------
```

RDMA Send

```
$ gongyi@node-0:~$ ib_send_lat --iters=1000000 --size=64 10.10.1.2
---------------------------------------------------------------------------------------
                    Send Latency Test
 Dual-port      : OFF          Device         : mlx4_0
 Number of qps  : 1            Transport type : IB
 Connection type : RC          Using SRQ      : OFF
 TX depth       : 1
 Mtu            : 1024[B]
 Link type      : Ethernet
 GID index      : 2
 Max inline data : 236[B]
 rdma_cm QPs    : OFF
 Data ex. method : Ethernet
---------------------------------------------------------------------------------------
 local address: LID 0000 QPN 0x022e PSN 0xbc00e5
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:128:110:218:30
 remote address: LID 0000 QPN 0x022e PSN 0xd38adc
 GID: 00:00:00:00:00:00:00:00:00:00:255:255:128:110:218:17
---------------------------------------------------------------------------------------
 #bytes #iterations  t_min[usec]   t_max[usec] t_typical[usec]  t_avg[usec]   t_stdev[usec]  99% percentile[usec] 99.9% percentile[usec]
Conflicting CPU frequency values detected: 801.133000 != 1408.438000. CPU Frequency is not max.
Conflicting CPU frequency values detected: 799.627000 != 2381.560000. CPU Frequency is not max.
 64      1000000         1.74          196.67      1.81           1.86           1.47          2.28                6.15
---------------------------------------------------------------------------------------
```

Measure RTTs with DPDK

```
$ gongyi@node-1:~$ sudo ./dpdk_echo -l2 --socket-mem=128 -- UDP_CLIENT 10.10.1.2 10.10.1.1 14 58 d0 58 ff c3
  EAL: Detected CPU lcores: 16
  EAL: Detected NUMA nodes: 1
  EAL: Detected shared linkage of DPDK
  EAL: Multi-process socket /var/run/dpdk/rte/mp_socket
  EAL: Selected IOVA mode 'PA'
  EAL: VFIO support initialized
  EAL: Probe PCI driver: net_mlx4 (15b3:1007) device: 0000:09:00.0 (socket 0)
  TELEMETRY: No legacy callbacks, legacy socket not created
  initializing with 1 queues
  Port 1 MAC: 14 58 d0 58 8f e3

  Core 2 running in client mode. [Ctrl+C to quit]
  Using static server MAC addr: 14:00:00:00:00:00
  ran for 5.000000 seconds, completed 432715 echos
  client reqs/s: 86543.000000
  mean latency (us): 11.576234
```