RDMA Microbenchmark Report

8/18/22

The RDMA Microbenchmark was written to allow a client machine to test the maximum sustainable throughput of a DUT (which could be a BlueField-2 DPU or any other host). The load generated by the client machine can be configured by via two knobs: 1. the number of client threads and 2. the client thread’s window size.

RDMA was used as the protocol for data movement since it enables the NIC to directly DMA messages to userspace memory on the host, providing at least an order of magnitude improvement in data movement latency compared to traditional kernel-based network stacks. The microbenchmark is written with native low-level RDMA Verbs.

Each client/server thread is pinned to a different core and has a unique queue pair (QP), created at run-time, and is identified by its unique QP number. One key requirement of this microbenchmark is to allow any client thread to send and receive messages to and from any server thread. Therefore, we opt to use UD (unreliable datagram) QPs to enable scalability in allowing local threads to communicate with any remote thread. With UD QPs, each QP on the client can send or recv messages from any QP on the server. Each client thread sends messages to server threads in round-robin fashion. Fig. 1 below shows the high-level picture of the benchmark.

Diagram

Description automatically generated

Fig. 1 High-level operational setup of the microbenchmark

As the maximum sustainable message rate of any DUT is dictated by the service time of each individual request, we instruct each server thread to perform just an echo (which is, in fact, a memcpy and a post\_send) to gauge the maximum message rate that the DUT cores can handle.

This RDMA microbenchmark performs two tasks:

1. Generates load to test the packet processing limits of the DUT.
2. Measures response latency.

The client knows how to connect to the server as it is provided the server’s IP address, starting QP number, and the number of server threads.

More Information on how to run the microbenchmark (with the two different modes of operation), along with all possible command line arguments is included in the repo’s README.

Experiments Run

1. What is the Client’s Maximum Send Rate?

The first experiment was designed to measure the maximum send rate of our client machine. In In this experiment, the client and server machines are each equipped with 4 Neoverse N1 cores. In this experiment, the client and server each spawn four threads with each thread pinned onto a unique core. Moreover, the client spawns an additional thread for latency measurement purposes. All client threads send messages to the server threads in round-robin fashion.

Fig.2 shows the setup for this experiment, with the red arrows indicating the network path through which packets flow. The client’s send rate saturates at 30.8 M msgs/s. The network is not the bottleneck in this experiment since, at 100Gbps, the peak rate supported by the network is 145M msgs/sec.

Graphical user interface

Description automatically generated

Fig. 2 Experiment one’s setup testing the client’s load generation limits.

1. Can the DPU’s Arm cores keep up with the client’s send rate?

The second experiment was designed to test the maximum message rate supported by the BlueField’s Arm cores as a separate host. In this experiment, the client is equipped with 4 Neoverse N1 cores while the server (BlueField-2 DPU) is equipped with 8 Arm Cortex-A72 cores. In this experiment, the client and server spawn four and eight threads, respectively, with each thread pinned onto a unique core. Moreover, the client spawns an additional thread for latency measurement purposes. All client threads send messages to the server threads in round-robin fashion.

Fig.3 shows the setup for this experiment, with the red arrows indicating the network path through which packets flow. The BlueField-2’s Arm cores’ packet processing rate saturates at 20.2 M msgs/s. Once again, the network is not the bottleneck.

Graphical user interface

Description automatically generated with medium confidence

Fig. 3 Experiment two’s setup testing the BlueField-2’s packet processing limits as a separate host.

1. What is the BlueField-2’s maximum sustainable message rate as a bump-in-the-wire (message forwarder)?

The third experiment was designed to test the maximum sustainable message rate supported by the BlueField’s Arm cores as a bump-in-the-wire between the client and server. This is an interesting experiment to run because many smartNICs are deployed as bump-in-the-wire accelerators. In this experiment, the client and server are each equipped with 4 Neoverse N1 cores while the BlueField-2 DPU is equipped with 8 Arm Cortex-A72 cores. In this experiment, the client and server each spawn four threads and the DPU eight threads, with each thread pinned onto a unique core. Moreover, the client spawns an additional thread for latency measurement purposes. On ingress, all client threads send messages to the DPU cores in round-robin fashion, and the DPU cores forward the messages in round-robin fashion the server cores. On egress all messages are send back to the respective client/DPU core which initially sent the request.

Fig.4 shows the setup for this experiment, with the red arrows indicating the network path through which packets flow. The BlueField-2’s Arm cores’ packet processing rate saturates at 9.52 M msgs/s. Once again, the network is not the bottleneck.

Graphical user interface, diagram, application

Description automatically generated

Fig. 4 Experiment three’s setup testing the BlueField-2’s packet processing limits as a bump-in-the-wire.

It is expected that the BlueField-2’s sustained throughput would drop by a ½. In experiment 2, the BlueField-2 compute power was all directed toward processing packets from one endpoint. However, in this experiment, the same compute is being applied to process packets from two different endpoints. Therefore, the drop in sustained throughput is expected.

We use tail latency (p99) to determine the saturation points for each of the three experiments. The saturation point is defined at the point where the tail latency starts to skyrocket. Fig. 5 shows the 99th percentile latency plotted against the sustained message rate for each of the three experiments.

Exp.1 – red

Exp.2 – yellow

Exp.3 – green

Chart, line chart

Description automatically generated

The BlueField-2 as a separate host can be a great co-processor for the host CPU. In experiment one, the host saturated at 30.8 M msgs/sec, while in experiment two, the DPU’s Arm cores saturated at 20.2 M msgs/sec. Theoretically, if the host receives a load larger than 30.8 M msgs/sec, it can redirect or balance the excess load to the DPU, which will be able to sustain another 20.2 M msgs/sec. Theoretically, this increases the system’s overall sustainable throughput to 51 M msgs/sec.

For any questions: please feel free to contact me at [hamed@gatech.edu](mailto:hamed@gatech.edu).