Poster: Pushing RDMA into Milliseconds RTT Communication

Jisu Ok
KAIST
Daejeon, Republic of Korea
jisu.ok@kaist.ac.kr

Wonsup Yoon
KAIST
Daejeon, Republic of Korea
wsyoon@kaist.ac.kr

Sue Moon
KAIST
Daejeon, Republic of Korea
sbmoon@kaist.ac.kr

Abstract

Thanks to its high performance and efficiency, RDMA has become an essential networking primitive in today's datacenter. However, existing RDMA-based systems mainly focus on accelerating data communication of endpoints away from each other in tens or hundreds of microseconds RTT. This paper aims to address the following question: How far can RDMA be employed without losing its performance and efficiency merit? We present our methodology and analysis to study the potential of RDMA networking over a millisecond-scale RTT path.

CCS Concepts

• **Networks** → *Network experimentation*; *Transport protocols*.

Keywords

RDMA, millisecond RTT network

ACM Reference Format:
Jisu Ok, Wonsup Yoon, and Sue Moon. 2024. Poster: Pushing RDMA into Milliseconds RTT Communication. In *Proceedings of the 20th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '24), December 9–12, 2024, Los Angeles, CA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3680121.3699883

1 Introduction

The promise of Remote Direct Memory Access (RDMA), which provides high throughput and ultra-low latency albeit low CPU utilization, has expedited datacenter workloads and RDMA is a primary networking primitive. Examples vary from RDMA-based key-value stores, transaction systems, and file systems to GPU data communication. As a result, RDMA's portion in datacenter network traffic has been steadily growing [4].

However, existing efforts and deployments from both academia and industries mainly focus on usage scenarios of RDMA where communicating endpoints reside in the microsecond-scale RTT range. RDMA-based systems usually operate under an intra-rack configuration where RTT is typically a few or at most tens of microseconds. Among hosts beyond a rack and between podsets, latency still stays in hundreds of microseconds [5]. RDMA has been proven to function as an efficient and effective method over such network paths, but its behavior on farther connections has drawn less attention.

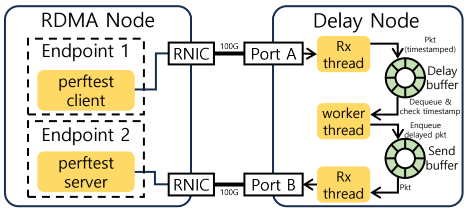

Figure 1: Experiment Setup. Only one direction of traffic flow is depicted for simplicity.

In this paper, we throw a question on RDMA's extendibility: How far can RDMA be employed without losing its performance and efficiency merits? As a first step to address it, we explore RDMA's performance over network paths whose RTT varies from tens of microseconds to milliseconds.[1] We introduce our experimental setup, which uses a network delay emulator, and present preliminary results showing RDMA throughput over millisecond RTT communication.

2 Emulating Millisecond-scale Path

Establishing a physically multi-hopped path in a testbed is challenging, particularly when the desired network latency falls in the millisecond range and the end-to-end path distance spans hundreds of kilometers. Instead we take the emulation approach for scalable experiments and precise control of path RTT. We have built and placed a delay emulation node between two communication endpoints as shown in Figure 1. The delay node is based on DPDK and follows the design of Rx/Worker/Tx thread separation [1]. The Rx thread receives packets from a port and enqueues them to the delay buffer. The Tx thread does similar delivery from the send buffer to the other port. The worker thread artificially injects delay into each packet by holding it up in the delay buffer for the pre-configured amount of time. Since the delay node is a transparent link layer element, it provides an illusion to the endpoints as if the endpoints are communicating directly with each other over a path with variable propagation delay.

The delay emulation system has to meet the following two requirements. First, queue buildup should be minimal except for in the delay buffer, which we intentionally induce by holding packets until their transmission times. The other source of queueing delay from packets being stacked up in the send buffer harms the accuracy of the emulation. Second, unintended packet loss must not occur on the data path of delay injection. Considering the RDMA's well-known vulnerability to packet loss, even a small number of packet drops would affect the communication performance and interfere with our measurement. Along with the engineering effort (*e.g.*, tuning system parameters such as queue sizes) to manage the

[1] 1 ms of delay corresponds to about 200 km of end-to-end path length, assuming the propagation speed is $2 \cdot 10^8$ m/s.

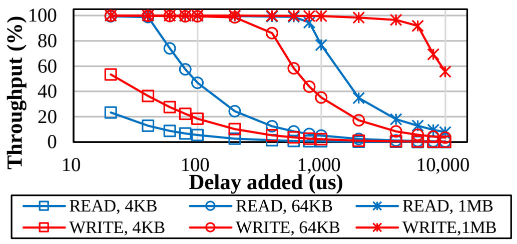

Figure 2: Added delay and normalized throughput.

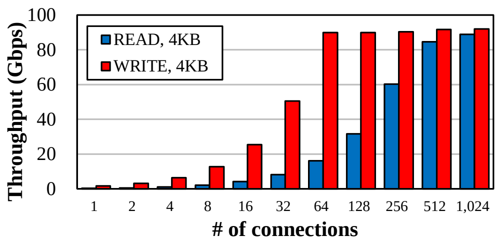

Figure 3: Number of connections and throughput.

above requirements, we monitor the length of each queue during the experiments and check whether any persistent queue buildup occurred. We also verify the existence of packet loss using ports and system-internal statistics. If any number of losses are detected, results from that run are excluded from our analysis. Our current version of the emulation system bears up to 16 Mpps traffic without packet loss. Scaling up the packet delay emulator further is also an interesting research direction and we leave this as a future work.

We measure RoCEv2 RDMA throughput between two endpoints using perftest [3], a well-known benchmarking tool for RDMA Verbs API. The endpoints and the delay node run on Ubuntu 22.04 servers with Intel Xeon Gold 6330 CPU @ 2.00 GHz and NVIDIA ConnectX-6 Dx 100G RNIC. PFC was enabled for each physical link. MLNX_OFED 23.10, DPDK 23.11, and perftest 6.17 were used.

3 Preliminary Results

Figure 2 plots the changing RDMA throughputs as path delay increases, with three different message sizes: 4KB, 64KB, and 1MB. The throughput in the y-axis is not absolute, but normalized to the baseline throughput where no artificial delay is added (*i.e.*, the shortest path possible).

From the figure we observe that the larger the message size is, the more resilient the throughput is to the path delay. When the message size is 1 MB, both READ and WRITE maintain more than 90% of their baseline throughput with the delay less than 1 ms. On the other hand, for small message sizes, the throughput drops steadily as delay increases. Mere 20 μs of added delay diminishes the throughput by more than 40% when the message size is 4 KB.

We note that the throughput decrease in longer path delay is not coming from the effect of larger PFC headroom as mentioned in [6]. Our emulation system adds delay to the packets virtually by holding them in an intermediate queue and the length of the physical link on which PFC operates remains as constant (a few meters). Congestion control (DCQCN in our configuration) also does not affect this result since the delay node does not touch the ECN bits of packets. As there is no change in ECN bits, DCQCN's congestion detection works unscathed even when path RTT varies [4].

Although the use of small message sizes under milliseconds of RTT achieves far lower throughput than the line rate, employing multiple parallel connections enhances the link utilization. As opposed to Figure 2 which reports a single connection performance, we increase the number of RDMA connections and present aggregate throughput in Figure 3. The added delay is fixed to 1 ms. Though a single connection achieves only 0.26 Gbps and 1.60 Gbps for 4KB READ and 4KB WRITE, respectively, the aggregate throughput increases linearly with more connections until it reaches the upper bound. This upper bound saturates the link bandwidth considering the RoCEv2 header overhead.

However, READ and WRITE exhibit different behaviors in their throughput reduction. READ experiences more severe degradation than WRITE in all message sizes (Figure 2). Consequently, it requires a greater number of connections to saturate the link (Figure 3). One possible cause for such distinct tendency is difference in the transport mechanisms of READ and WRITE [2]. READ throughput is bottlenecked by the maximum number of outstanding requests and this impact is exacerbated in longer delay settings. Results here suggest an insight for designing RDMA-based systems over millisecond-scale network: by favoring WRITE over READ whenever possible, the system is expected to deliver better performance and more stability.

4 Conclusion & Future Work

This paper investigates the performance of RDMA over network paths of millisecond latency. The preliminary study based on a network delay emulation demonstrates that RDMA still retains its high throughput merit of saturating the line rate when we use a large message size or multiple parallel connections. This implies that increased RTT alone does not limit the RDMA's potential as a communication method over the long path. Results also suggest another insight for different modes of RDMA operation: READ throughput is more vulnerable to RTT increase than that of WRITE.

In this work we have only varied RTT while keeping zero packet loss. Along with RTT, the probability of packet drop or corruption also increases as the network path lengthens. We plan to test RDMA's performance characteristics under the varying loss rate. Experimenting with a more comprehensive set of application workloads is another future work.

Acknowledgments

This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2022R1A2C200906213)

References

[1] [n. d.]. DPDK Sample Applications User Guides. https://doc.dpdk.org/guides-23.11/sample_app_ug/packet_ordering.html
[2] [n. d.]. InfiniBand Architecture Specification Volume 1 Release 1.4.
[3] [n. d.]. perftest. https://github.com/linux-rdma/perftest
[4] Wei Bai et al. 2023. Empowering Azure Storage with RDMA. In *NSDI 2023*. USENIX Association, Boston, MA, 49–67.
[5] Yixiao Gao et al. 2021. When Cloud Storage Meets RDMA. In *NSDI 2021*. USENIX Association, 519–533.
[6] Cisco Systems Inc. 2009. Priority Flow Control: Build Reliable Layer 2 Infrastructure.