On the Impact of Cluster Configuration on RoCE Application Design

Yanfang Le
University of Wisconsin-Madison

Mojtaba Malekpourshahraki
University of Illinois at Chicago

Brent Stephens
University of Illinois at Chicago

Aditya Akella
University of Wisconsin-Madison

Michael M. Swift
University of Wisconsin-Madison

ABSTRACT

RDMA over Converged Ethernet (RoCE) allows RDMA-enabled NICs to operate in datacenter networks. This study focuses on identifying how different aspects of datacenter cluster configuration impact the latency, and throughput, and CPU utilization of different ways of transferring data in RoCE (RDMA verbs). We look into the impact of colocated applications competing for both the CPU and access to the NIC as well as the impact of the network MTU. We find that RDMA applications do not fairly share the NIC, large frames should not be used, and that correct verb choice is dependent on many variables, including application access patterns, object size, and the load of both the local and remote CPU.

CCS CONCEPTS

• **Networks** **Network adapters**; **Network measurement**; **Data center networks**; *Network servers*.

KEYWORDS

RDMA, Datacenter, RDMA Performance, Performance Measurement

ACM Reference Format:
Yanfang Le, Mojtaba Malekpourshahraki, Brent Stephens, Aditya Akella, and Michael M. Swift. 2019. On the Impact of Cluster Configuration on RoCE Application Design. In *APNet '19: 3rd Asia-Pacific Workshop on Networking, August 17–18, 2019, Beijing, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3343180.3343190

ACM ISBN 978-1-4503-7635-8/19/08…$15.00
https://doi.org/10.1145/3343180.3343190

1 INTRODUCTION

RDMA-enabled NICs (RNICs) can transfer data across the network directly to and from application buffers without involving the CPU [9]. This paradigm can offer significant performance benefits over traditional networking stacks, reducing both latency and CPU overheads [5, 13]. After initial popularity in high performance computing, RNICs are beginning to be deployed in datacenter networks [5, 6, 18, 19, 26], including large online service providers such as Microsoft [8, 26]. This is because RDMA can improve the performance of the on-line data-intensive (OLDI) applications frequently run in datacenters [5, 13, 22].

Datacenters host many competing applications, and load in datacenters is bursty and diurnal [2, 7, 23] . Not only are there many different ways to design applications that use RDMA for messaging (RDMA applications), but the correct configuration for an RDMA application may depend on both the load placed on a cluster and cluster configuration, *e.g.*, how jobs are placed and how the network is configured. Prior work has benchmarked the performance of different aspects of the RDMA protocol itself as implemented by RNICs [5, 13, 16, 25] and the performance of RDMA congestion control [19, 20, 26]. However, the impact of differences in cluster configuration remains unclear, despite different configurations having the potential to significantly affect RDMA performance.

This paper presents the first measurement study of RDMA performance that focuses on the impact of cluster configuration—both the placement of jobs in the cluster and the configuration of the cluster's network. In particular, our goal is to help guide decisions that must be made by operators as part of deploying an RDMA cluster. For example, whether or not RDMA applications need to be run on dedicated servers is a decision that has a significant impact on capital expenditures and operational complexity. Similarly, an effective configuration could potentially increase application performance and reduce overall server CPU utilization.

Specifically, we seek to answer the following questions that have been previously unanswered by prior work:

What is the impact of applications competing for CPU resources on RDMA performance? Unlike HPC clusters, datacenter clusters often host different competing applications and/or tenants. While core datacenter services may be run in isolation [8], it may not be feasible to isolate every application in a datacenter.

What RDMA verbs should an application use? RDMA provides a rich set of operations (verbs). Not all verbs have equal CPU costs, so competing with other applications for CPU is expected to harm the performance of some verbs more than others. We investigate the impact of CPU utilization on RDMA performance in order to determine if RDMA applications should be designed differently depending on whether or not they will be run on a dedicated CPU core.

Should RDMA applications use polling or interrupts? Polling can reduce RDMA latency [5, 13]. Does this still hold if other applications are competing for the CPU?

What is the impact of RDMA applications competing for RNIC resources on performance? To improve cluster scalability, it may be desirable to colocate competing RDMA applications on a server. We investigate how competing RDMA applications share an RNIC.

How does RDMA performance change if the underlying network allows jumbo frames? All Ethernet devices we are aware of that support RoCE (DCB) also support frames larger than 1500 B, *i.e.*, jumbo frames. We investigate the impact of using different message sizes on the performance of different RDMA verbs to determine if operators should allow RDMA applications to generate jumbo frames.

In order to answer these questions, we performed experiments on a dedicated CloudLab [3] cluster of 17 servers that each use a 10 GbE Mellanox ConnectX-3 NIC [17] to connect to an HP 45XGc Ethernet switch [10]. From our experiments, we draw the following conclusions:

Choosing the best performing verb is difficult. Even when run in isolation, choosing the correct verb depends on many variables, including object size, memory access patterns, and whether latency or throughput is more important.

Polling is faster only without CPU contention. If an application wants low latency and energy efficiency, then it will need to dynamically switch between polling and interrupts.

CPU contention does not significantly hurt RDMA performance. Competing for the CPU changes whether interrupts or polling should be used. Otherwise, RDMA application performance is not significantly impacted by changes in CPU load.

Competing RDMA applications do not fairly share the RNIC. It may be desirable to run multiple competing RDMA applications on a single server. Unfortunately, RDMA applications do not fairly share the NIC based on either the number of outstanding verbs or active queue pairs. Instead, RDMA applications performing longer transfers (4 MB) starve latency sensitive applications.

Jumbo frames increase tail latency. Surprisingly, our results show that there is little incentive to having RNICs generate large frames. Although jumbo frames provide a small throughput benefit, they significantly increase tail latency.

The rest of this paper is organized as follows. First, Section 2 provides background information on RDMA. After that, Section 3 explains our experimental methodology, and Section 4 presents our experimental results. Next, Section 5 discusses our results, and Section 6 discusses related work. Finally, Section 7 concludes.

2 BACKGROUND

RoCE (RDMA over Converged Ethernet) enables RDMA to operate over datacenter Ethernet/IP networks [11, 12]. This section introduces RDMA, and then discusses different ways to configure RDMA messages (verbs).

2.1 RDMA Overview

Remote Direct Memory Access (RDMA) allows a host to directly access the memory of a remote host without involving the operating system. This eliminates the overheads associated with TCP/IP networking, such as context switching and memory overhead for data copying. Because of this, RDMA can provide a low-latency messaging with consistent performance [5, 13, 18].

In order to accomplish this, userspace applications interface directly with RDMA NICs by sending RDMA *verbs* to queues on the RNIC. The most commonly used verbs are READ, WRITE, and SEND/RECV. READ is used to fetch data from the memory of a remote host, and WRITE transfers data into the memory of a remote host. SEND/RECV are used together. SEND transfers a datagram to a remote host, and RECV is a special operator that is used to receive the message from a SEND. READs and WRITEs are *one-sided*—they do not involve the remote host's CPU. In contrast, SEND/RECV are two sided—the remote host's CPU is involved in the transport because it must post a RECV before a SEND arrives for the SEND to complete. READs, WRITEs, and SENDs are posted to *send queues*, and RECVs are posted to *receive queues*. Each queue pair (QP) is associated with a completion queue (CQ) which is used to signal the completion of events.

2.2 Configuring RDMA Messaging

Not only do RDMA applications have freedom in verb choice, but individual verbs also have multiple different modes of operation. We now highlight some important configuration choices relevant to this study:

Interrupts vs. Polling: One benefit of RDMA is that it reduces the CPU load of networking. However, this does not prevent applications from polling. RDMA allows an application to chose between receiving events (delivered by interrupts) when verbs complete and polling for the completion of RDMA verbs. Further, despite WRITEs being one-sided and not involving the remote CPU, the remote host can still poll memory to identify the arrival of new data [13]. We study the differences between using interrupts and using polling because both their end-to-end network performance and CPU load differ.

Signalling: An application can also choose to eliminate the signalling of the completion of verbs. If an application does not need to be signaled when a verb completes, then disabling signalling for the verb can improve performance by eliminating messages that otherwise would be sent across the network and/or PCIe bus [13].

One-sided vs. Two-sided READs (GETs): If an RDMA application wants to read data from a remote host, it has two choices. One is to perform a READ, and the other is to send either WRITE or SEND to an application on the remote host that processes the request and then WRITEs or SENDs a reply. We call the latter approach a *compound read* or a *two-sided read*. A benefit of using compound reads is that they can be faster than READs if an application has to perform multiple dependent memory accesses (*e.g.*, pointer chasing) [13]. A negative aspect of compound reads is that their performance depend on the CPU utilization of the remote server, unlike one-sided READs. We investigate the impact of competing for the CPU on the performance of both READs and compound reads.

Jumbo Frames: Jumbo frames are Ethernet frames larger than 1500 B. Although Ethernet jumbo frames are not included as part of DCB [4] or any other Ethernet standard, all of the devices we are aware of that support DCB also support jumbo frames. Jumbo frames have the potential to increase throughput, but they can also increase latency when small messages compete with jumbo frames. Because of this, we study the impact of message size on RDMA performance.

3 METHODOLOGY

In order to evaluate RoCE performance, we performed experiments on a cluster of 17 CloudLab [3] servers. Each server has an 8-core ARM Cortex-A57 processor, 64 GB of memory, and a 10 GbE Mellanox ConnectX-3 NIC [17]. In our experiments, all of the servers connect to a single HP 45XGc Ethernet switch [10]. As part of future work, we plan on looking at larger topologies. To minimize the impact of network congestion in our experiments, we perform experiments that are not likely to congest the network, and we use windowing to limit the number of outstanding requests. Unless otherwise specified, we limit the number of outstanding requests to 16.

Similarly, we always use both unconnected transports and inlining when possible. Only WRITEs and SENDs may be unreliable. On the ConnectX-3, the largest message size that may be inlined is 256 B. The largest RDMA packet that we can generate is only 4 KB plus the size of the RoCEv2, Ethernet, VLAN, and IP headers, even though the NIC can send larger Ethernet frames. This is because 4096 is the largest RDMA `IBV_MTU` supported by the NIC. To evaluate the impact of colocating RDMA applications with other applications contending for the CPU, we pin a CPU-bound application with a small memory footprint to each core of every server in the cluster. Specifically, we use a program that solves the *N-queens* problem. Finally, in all of our experiments, PFC is enabled and RDMA traffic runs at a higher priority than all other traffic. Additionally, we run all of our experiments for at least 10 seconds.

4 EVALUATION

This section presents the results from experiments designed to understand how configurable parameters impact RDMA performance. In particular, we look at the impact of CPU utilization, packet sizes, and the choice between interrupts and polling on both the latency and throughput of different RDMA verb choices. Moreover, we focus on the performance of reading data from a remote machine because there are more ways to configure reads than writes.

4.1 Performance in Isolation

To provide a baseline, Figure 1 shows the average latency of different verbs on an otherwise unloaded CPU for different message sizes, including message sizes that require jumbo frames. After that, Figure 2 provides a baseline of the maximum throughput achievable when an application can use the RNIC in isolation.

Latency. To measure the latency of messaging, we performed one remote memory access at a time between two servers. Figure 1 shows the average latency of four different methods of fetching data from a remote machine. *READ-Intr* is the latency of READs that are signalled via interrupts. Next, the *WRITE/SEND-Poll* line is the latency it takes to complete a WRITE/SEND compound read that uses polling. Kalia *et al.* [13] argue this is the fastest verb for implementing a key-value store. Compound reads built with SENDs may either use interrupts or polling. The line *SEND/SEND-Intr* shows the performance of a compound SEND with interrupts. Although we looked at SEND/SEND-Poll, we elide these results because it never achieved a throughput of more than 2 Gbps. Finally, READ-Poll is the latency of a READ when the application chooses to ignore the completion queue and instead

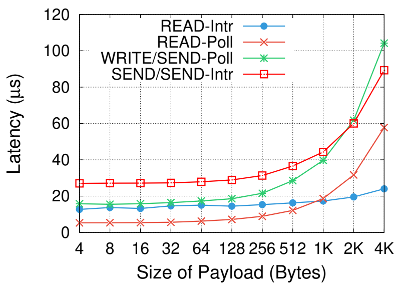
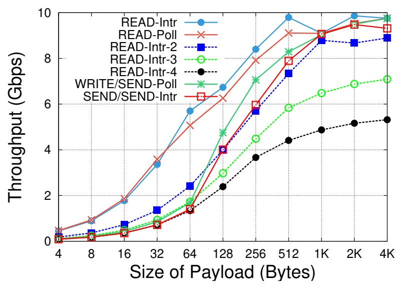
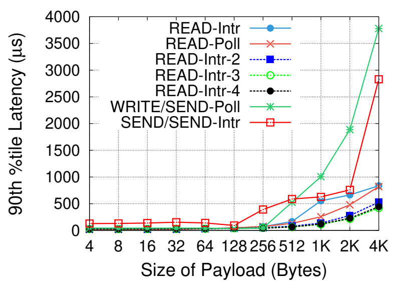


Figure 1: Latency when verbs are sent one at a time between two servers on an otherwise unloaded CPU.



(a) Throughput



(b) 90th %tile latency in Figure 2a.

Figure 2: The throughput and tail latency of many servers reading data as fast as possible from a remote host, saturating the remote RNIC.

poll memory directly for the arrival of data. Prior work has not evaluated this configuration, and we find that it performs surprisingly well for small payload sizes.

First, we find that Figure 1 confirms the results of prior studies that have used similar configurations [5, 13]. As reported by Kalia *et al.* [13], for small payload sizes, WRITE/SEND-Poll have about the same latency as READ-Intr. This means that compound reads (WRITE/SEND) can match the latency of READs, at least for inlinable payloads (≤256B). Also, SEND is expected to be slow [13], and this is confirmed by this experiment. Until large payload sizes, a SEND/SEND-Intr is twice the latency of a WRITE/SEND-Poll.

However, a few aspects of Figure 1 are surprising. First, READ-Poll has less than half of the latency of both signalled READs and WRITE/SEND-Poll. This approach is inspired by Kalia *et al.* [13], who found that polling memory leads to lower latency WRITEs than relying on the completion queue. We find that polling memory significantly reduces the latency of READs as well. This implies that, if a system needs to fetch only a single small value from remote memory, then READ-Poll would provide the best performance. However, this approach fully utilizes the CPU.

In Figure 1, we also find that large frame sizes increase the latency of many verbs by significantly more than the increased time it takes to transfer the message from memory and across the network. A payload of 4096 B takes 3.3 μs to transfer across a 10 Gbps link, and transferring the message to and from memory should take significantly less time than it takes to cross the network [13]. However, READ-Intr is the only verb that does not see an unexpected increase in latency as message size increases.

Throughput. For small payload sizes, throughput may be limited by the maximum number of operations per second the RNIC can process. At large payload sizes, network bandwidth may be the limiting factor on throughput. Independent of payload size, the extra CPU and RNIC involvement of two-sided operations can hurt throughput.

To understand these effects, Figure 2a illustrates the impact of message size and the choice of interrupts or polling on the performance of different RDMA verbs. To stress the performance of the RNIC while avoiding the problems associated with network congestion [26], we use many clients to read data from a single server. In this experiment, 16 clients connect to a single server. On the server, RDMA application processes are pinned to each of the server's 8 CPUs. The clients then establish a connection with each server process and maintain a window of 16 outstanding reads on each connection. With one-sided reads, accessing an object may take multiple memory accesses. The READ-Intr-{2,3,4} lines plot the throughput of object access if 2, 3, or 4 total RDMA operations are required to access the data. In these lines, the clients first issue 1, 2, or 3 READs for 8 B of data before reading an object of the payload size.

In Figure 2a, we first see that throughput is limited by the maximum number of operations per second of the RNIC for payloads of 256 B or smaller. Figure 2a also shows that, as we would expect, choosing either interrupts or polling has little impact on throughput when there is no CPU contention. Finally, this figure shows that one-sided READs always have higher throughput than two-sided compound reads.

Interestingly, our results sometimes differ from those in prior work [5, 13]. For example, Kalia *et a.* [13] found that the throughput of WRITE/SEND could match the throughput of READs for 32 B messages (~6.7 Gbps). We expect that these differences are due to differences in RNICs, as prior work [5, 13] has used 40 Gbps and 56 Gbps RNICs, while our study uses 10 Gbps RNICs. This result further illustrates the difficulty of writing RDMA applications. In addition to verb choice, the best performing design depends on RNICs.

A major difference in our results is that we find it is not always better to use compound reads to access objects on our RNICs. For objects that are 64 B or smaller, one-sided READs (READ-Intr-2) are faster than two-sided accesses (WRITE/SEND-Poll) even if two messages are necessary, although two-sided operations are still faster for larger objects.

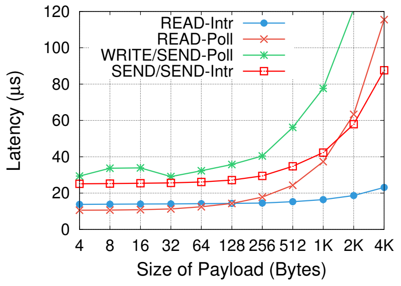
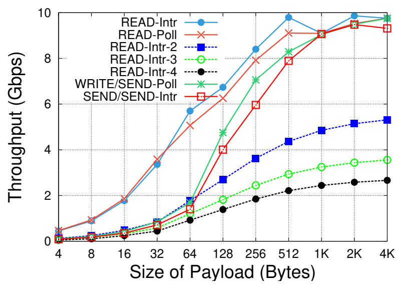
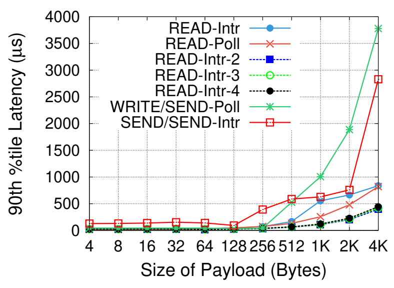


Figure 3: Latency when the CPU is fully utilized and verbs are sent one at a time between two servers.



(a) Throughput with CPU Contention



(b) 90th %tile latency in Figure 4a.

Figure 4: The impact of CPU contention on the throughput and tail latency of servers reading data as fast as possible from a remote host.

Further, we find that READ-Intr-3 can match the performance of WRITE/SEND-Poll for small objects, although the performance advantage of WRITE/SEND-Poll becomes apparent at larger payloads.

Impact of RNIC load on latency. When the RNIC is heavily loaded, the latency of communication can increase. There are two different ways an RNIC may be loaded. An application may be limited by the number of operations per second supported by the RNIC, the line-rate of the RNIC, or both. In contrast with the experiment in Figure 1, where the RNIC is lightly loaded, Figure 2b shows the 90th percentile latency of the requests issued in the experiment in Figure 2a, where the RNIC is fully utilized.

Figure 2b shows that the verb latency of small messages is stable, even if the messaging rate of the RNIC is limiting throughput. For READs 64 B and smaller, the latency is always 34 μs or smaller. Similarly, the 90th percentile latency of WRITE/SEND-Poll is consistently around ~45 μs for requests 256 B and smaller. However, at large payload sizes, when the line-rate of the NIC limits throughput, tail latency latency can significantly increase. The tail latency of READ-Intr is over 500 μs for 1K payloads, and the tail latency of both reads is around ~830 μs for 4K payloads. In Figure 2b, the tail latency of WRITE/SEND and SEND/SEND are *2.8 ms* and *3.8 ms*. This tail latency is close to that reported by DeTail [24] when TCP is used to transfer 8 KB packets on a network with PFC enabled. Because the tail latency of READ-Intr-{2,3,4} is impacted the least, we expect that these overheads are related to memory management.

4.2 Impact of CPU Contention

As the number of RDMA applications run in a cluster increases, it quickly becomes economically infeasible to isolate RDMA applications from each other and other applications running in the datacenter. If an application designer expects an application will not be run in isolation, they may want to change how their application is designed because competing for the CPU is expected to reduce the performance of some verbs more than others and hurt polling more than interrupts.

Latency. First, Figure 3 shows the latency of issuing one verb at a time between two servers (as in Figure 1) when the N-queens program competes for the CPU. Interestingly, Figure 3 shows that, for small payload sizes, the latency of different verbs is independent of whether polling or interrupts are used. At small payload sizes, the lines that involve only the local CPU (READ-Sig and READ-Poll) have half the latency of the lines that involve both the local and remote CPU (WRITE/SEND-Poll and SEND/SEND-Intr).

On the other hand, the latency of compound reads and polling increases at larger payload sizes. Like Figure 1, Figure 3 also shows that READ-Intr is the lowest latency verb at large payload sizes. In contrast, SEND/SEND-Intr takes roughly four times longer than READ-Intr, although it is faster than WRITE/SEND-Poll.

Throughput. Figure 4a shows the impact of CPU contention on the throughput of different ways of fetching data. If other applications are competing for the CPU, we would expect the throughput of two-sided verbs and polling to drop more than one-sided verbs and interrupts, and this is the case. READ-Intr provides the highest throughput for all message sizes. However, at small message sizes, the throughput of using interrupts and polling is similar. As before, READs provide the highest throughput. Although this may seem to imply that one-sided READs should always be used if the CPU is heavily utilized, this is still not always the case. If the CPU is fully utilized, a single compound read is significantly faster than performing even two dependent READs.

Tail latency with CPU load. Surprisingly, the latency of transfers in Figure 4b does not differ significantly from Figure 2b. However, because both of these figure show a significant increase in tail latency when large message sizes are used, we conclude that jumbo frames should not be used because of their impact on the latency.

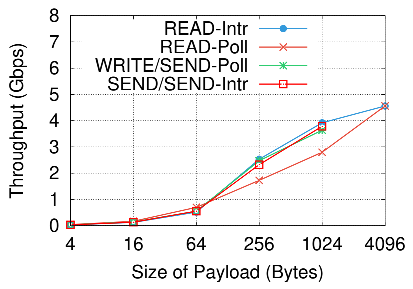


Figure 5: The throughput of small message transfers when competing with large (4 MB) background flows.

4.3 Impact of RNIC Contention

So far, we have not looked at the impact of multiple applications competing for access to the RNIC. Microsoft reportedly uses RDMA to access both storage and key-values stores [5, 8, 26]. While Microsoft has said that message transfers in their RDMA deployment are often less than 4 MB in size [26], most key-value store accesses are much smaller. Over 90% of all values in a Facebook memcached cluster were under 256 B [1].

Figure 5 shows how traffic from show message flows changes when competing with 4 MB background flows. Most interestingly, this figure shows that competing applications do not fairly share the RNIC. Although CPU contention does not significantly impact the performance of some verbs, RNIC contention significantly hurts the throughput of small messages for all verbs.

5 DISCUSSION

Prior work claims that WRITE/SEND-Poll is the proper verb choice for applications that need to perform two or more dependent memory accesses [13]. However, our results show that verb choice is more complicated. These differences from prior work are likely due to hardware differences (10 Gbps vs 40 Gbps). Our hope is that these findings spark more investigation into the performance differences between RNICs and help lead to future RDMA applications that are designed to be RNIC platform independent.

Further, our results in Section 4 show that correct verb choice is dependent on other variables, including the load of both the local and remote hosts and whether or not polling or interrupts are used. This implies that a system should be built differently depending on whether it is optimized for latency, throughput, and whether hosts are expected to be heavily utilized. However, designing a system around these trade-offs can lead to significant application complexity.

Instead, we believe that this problem should be addressed by creating a higher-level RDMA abstraction that hides these many complexities from applications. Although modern OSes dynamically schedule packet I/O and switch between interrupts and polling in response to changes in CPU load, this complexity is hidden from applications [21]. Similarly, RDMA applications could interface with a common library that dynamically adjusts RDMA configuration in response to changes in system load.

However, there are a few difficulties in implementing such a library. Dynamically switching between polling and interrupts for two-sided operations requires coordination between communicating applications to ensure messages are not lost. Further, because different RDMA applications do not fairly share RNIC resources, it is not entirely clear how to design such a system while still maintaining the benefits of OS-bypass provided by RNICs.

6 RELATED WORK

RDMA performance has been a subject of much recent research [5, 13, 14, 16, 19, 20, 25, 26]. However, despite this, the questions answered in this proposal have largely remained unanswered. For example, Le *et al.* [16] measure the impact of verb size on CPU utilization but do not compare the utilization of different verbs. Zhang *et al.* [25] measure RDMA performance in the context of performance isolation between competing RoCE applicationsv. While they similarly find that latency-sensitive flows do not compete fairly with large flows, they do not evaluate the impact of CPU contention on RDMA performance. Dragojević *et al.* [5] and Kalia *et al.* [13] both measure RoCE performance. However, they focus on measuring the performance of different verbs and memory regions. Additionally, Mittal *et al.* [20] performed an analysis of RDMA congestion control and the need for lossless networking with RDMA NICs, which is complementary to the analysis performed in this paper. FlexNIC [15] is able to provide many of the benefits of RDMA without requiring connections, including OS bypass and zero-copy;however, it requires hardware that is not yet commodities, unlike RNICs.

7 CONCLUSIONS

This paper studies the impact of colocated applications in a datacenter contenteding for both the RNIC and CPU on the performance of different ways application developers can design RDMA applications. We find that large frame sizes hurt tail latency and do not significantly improve throughput. We also find that larger transfers can unfairly starve short message transfers when multiple RDMA applications are allowed to share RNIC access. Finally, we find that correct verb choice is dependent on many variables, including object size, CPU load, and the number of dependent memory accesses that are required to find the object. We believe the results in this study motivate creating a higher-level library that hides these complexities from applications.

REFERENCES

[1] Berk Atikoglu, Yuehai Xu, Eitan Frachtenberg, Song Jiang, and Mike Paleczny. 2012. Workload Analysis of a Large-scale Key-value Store. In *SIGMETRICS*. ACM.

[2] Theophilus Benson, Aditya Akella, and David A. Maltz. 2010. Network Traffic Characteristics of Data Centers in the Wild. In *IMC*.

[3] CloudLab [n.d.]. CloudLab. http://cloudlab.us/.

[4] Data Center Bridging Task Group. [n.d.]. http://www.ieee802.org/1/pages/dcbridges.html.

[5] Aleksandar Dragojević, Dushyanth Narayanan, Orion Hodson, and Miguel Castro. 2014. FaRM: Fast Remote Memory. In *NSDI*. USENIX.

[6] Aleksandar Dragojević, Dushyanth Narayanan, Edmund B Nightingale, Matthew Renzelmann, Alex Shamis, Anirudh Badam, and Miguel Castro. 2015. No compromises: distributed transactions with consistency, availability, and performance. In *SOSP*. ACM.

[7] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: A Scalable and Flexible Data Center Network. In *SIGCOMM*.

[8] Albert Greenberg, Microsoft Azure. 2014. ONS 2014 Keynote.

[9] Jeff Hilland. 2003. *RDMA Protocol Verbs Specification*. Internet-Draft draft-hilland-rddp-verbs-00. Internet Engineering Task Force. https://tools.ietf.org/html/draft-hilland-rddp-verbs-00 Work in Progress.

[10] HP Moonshot-45XGc Switch Module [n.d.]. HP Moonshot-45XGc Switch Module. http://www8.hp.com/us/en/products/moonshot-systems/product-detail.html?oid=7398915.

[11] InfiniBand Trade Association. 2010. Supplement to InfiniBand Architecture Specification Volume 1 Release 1.2.1 Annex A16: RDMA over Converged Ethernet (RoCE). https://cw.infinibandta.org/document/dl/7148.

[12] InfiniBand Trade Association. 2014. Supplement to InfiniBand Architecture Specification Volume 1 Release 1.2.1 Annex A17: RoCEv2. https://cw.infinibandta.org/document/dl/7781.

[13] Anuj Kalia, Michael Kaminsky, and David G. Andersen. 2014. Using RDMA Efficiently for Key-Value Services. In *SIGCOMM*. Chicago, IL.

[14] Anuj Kalia, Michael Kaminsky, and David G. Andersen. 2019. Datacenter RPCs can be General and Fast. In *NSDI*.

[15] Antoine Kaufmann, Simon Peter, Naveen Kr. Sharma, Thomas Anderson, and Arvind Krishnamurthy. 2016. High Performance Packet Processing with FlexNIC. In *ASPLOS*.

[16] Yanfang Le, Brent Stephens, Arjun Singhvi, Aditya Akella, and Michael M. Swift. 2018. RoGUE: RDMA over Generic Unconverged Ethernet. In *ACM Symposium on Cloud Computing (ACM SoCC)*.

[17] Mellanox Technologies. [n.d.]. ConnectXÂő-3 EN Single/Dual-Port 10/40/56GbE Adapters w/ PCI Express 3.0. http://www.mellanox.com/page/products_dyn?product_family=127.

[18] Christopher Mitchell, Yifeng Geng, and Jinyang Li. 2013. Using One-Sided RDMA Reads to Build a Fast, CPU-Efficient Key-Value Store.. In *USENIX Annual Technical Conference*. San Jose, CA.

[19] Radhika Mittal, Terry Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. 2015. TIMELY: RTT-based Congestion Control for the Datacenter. In *SIGCOMM*.

[20] Radhika Mittal, Alexander Shpiner, Aurojit Panda, Eitan Zahavi, Arvind Krishnamurthy, Sylvia Ratnasamy, and Scott Shenker. 2018. Revisiting Network Support for RDMA. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*.

[21] Jeffrey C. Mogul and K. K. Ramakrishnan. 1997. Eliminating Receive Livelock in an Interrupt-driven Kernel. *ACM Transactions on Computer Systems* (Aug. 1997).

[22] Marius Poke and Torsten Hoefler. 2015. DARE: High-Performance State Machine Replication on RDMA Networks. In *HPDC*.

[23] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. 2015. Inside the Social Network's (Datacenter) Network. In *SIGCOMM*. ACM.

[24] David Zats, Tathagata Das, Prashanth Mohan, Dhruba Borthakur, and Randy Katz. 2012. DeTail: Reducing the Flow Completion Time Tail in Datacenter Networks. In *SIGCOMM*.

[25] Yiwen Zhang, Juncheng Gu, Youngmoon Lee, Mosharaf Chowdhury, and Kang G. Shin. 2017. Performance Isolation Anomalies in RDMA. In *Proceedings of the Workshop on Kernel-Bypass Networks (KBNets '17)*. ACM.

[26] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion Control for Large-Scale RDMA Deployments. In *SIGCOMM*. ACM. http://research.microsoft.com/apps/pubs/default.aspx?id=252307