Evaluating the Use of RDMA in High Performance Containers

Honors Computer Science Bachelor's Thesis Proposal Document

Author: Emerson Ford

Advisor: Ryan Stutsman

Honors Faculty Advisor: Tom Henderson

Director of Undergraduate Studies: James de St. Germain

January 25, 2021

Abstract

Containers are an increasingly popular packaging framework for complex applications, offering benefits such as lightweight isolation, portability, and ease of deployment. These benefits offer a solution to a myriad of issues that have long been present in high performance computing world, creating a compelling narrative for container adoption in these environments. Unfortunately, standard container runtimes rely on relatively slow, virtualized network stacks to achieve network isolation and portability, which are incompatible with the kernel bypass networking technology RDMA. This has all but prohibited the widespread adoption of containers in high performance computing environments where RDMA-reliant applications are quite common.

Fortunately, recent projects such as Microsoft's FreeFlow and Mellanox's RDMA hardware have created solutions for enabling RDMA inside of containers while maintaining varying degrees of the benefits of standard containers. However, despite the strong claims made by these solutions, many of their characteristics such as performance, isolation sacrifices, and scalability are either not well documented or simply not known; characteristics that are critical to assess for several high performance computing use cases. This paper will attempt to remedy this issue by identifying, measuring, and comparing these characteristics for the various solutions available today for enabling RDMA in containers. Ultimately, this should provide high performance computing end-users a more holistic perspective on which solutions may be most viable for their environment and use case.

1 Introduction

Containers offer a promising solution to a host of issues that have long plagued the high performance computing (HPC) world, notably dependency management, portability, and reproducible environments, through the use of statically built images and standardized container runtimes [53]. For users, cluster administrators, and application developers, this would provide more freedom and ease in application development and deployment as it eliminates the burden of working around the hundreds of possible environments on

which the application may be deployed. As an added benefit, this also opens the door for easy migration and utilization of new HPC platforms such as the cloud, Kubernetes, and SLATE as these platforms provide first-class container support.

Despite these numerous benefits, the adoption of containers in the HPC world has been particularly slow [CITE NEEDED]. Among other issues like the until recent lack of rootless container runtimes, native containers do not support Remote Direct Memory Access (RDMA), a networking technology that provides extremely low latency and high throughput by offloading memory accesses from the CPU to NIC hardware (hence the term "kernel bypass networking") [47].

This is one of the core barriers for container adoption in HPC environments. RDMA has become a necessity in HPC environments due to their large, parallel workloads that often span dozens of machines.

Thus, the use of kernel bypass networking inside of containers requires either a complete circumvention of the container network stack (possible with "host networking" mode) or reworking the kernel bypass networking stack to be compatible with container network overlays. As the former eliminates several portability and isolation benefits that come with standard containers [22], many vendors and research groups have developed solutions for the latter such as Microsoft FreeFlow [22], MasQ [18], and Mellanox Shared HCA [45].

The story of shared, isolated use of RDMA NICs is by no means new as this has been a long standing problem for virtual machines and cloud providers [18] [ADDITIONAL CITE]. In fact, several of the solutions previously mentioned were initially developed for virtual machines and were subsequently applied to containers. However, containers are quite unique in their needs and various usage patterns, which differ significantly from those of virtual machines. In particular, containers are unique in that they are (a) treated as more ephemeral constructs than VMs, (b) can be dynamically scaled up or down on one or many hosts, (c) a single host could run tens to hundreds of containers, and (d) heavily utilize software defined networking to facilitate inter-container communication.

Even for solutions made first-class for containers, they often fail to be satisfactory for every use case of RDMA and containers, thus are not a "one size fits all" solution.

Ultimately, many of these solutions make varying trade-offs in isolation, performance, or portability, which may be made worse or better by the usage patterns they are put under. Unfortunately, these solutions are notorious in not documenting these trade-offs or they fail to provide critical information such as scalability or latency-throughput graphs. This makes it difficult for HPC end-users to determine which solution would operate best for their use cases.

For example, HPC centers that use containers for job-scheduling are likely not worried about security isolation, but will care deeply about maintaining full RDMA performance and reducing CPU overhead. On the other side, HPC centers that use containers for DMZ-like environments or for multi-tenant environments will find RDMA isolation and controllability between containers to be critical to maintain.

In particular, containers are unique in several key ways:

- Containers are treated as ephemeral
- A single machine could run tens to potentially hundreds of containers
- Container networking is nearly all software-defined

In particular, container semantics dictate the following for container networking:

- Support for data plane policies such as traffic shaping, QoS, and metering
- Support for control plane policies such as firewall rules, IP translation, and routing
-
- Running HPC jobs using a scheduler like SLURM.
- Running HPC jobs using a scheduler like Kubernetes.
-

Therefore, the specific performance characteristics to be identified are: throughput, latency, packets/requests per second, CPU overhead, memory overhead, and scalability. The latter characteristic is particularly important as container environments typically run

multiple containers on a single host, thus the question of how performance characteristics change as the number of containers on a single host increases is a concern in these environments.

Therefore, the specific isolation characteristics to be identified are: namespace-like isolation, control plane policy enforcement, data plane policy enforcement, and fairness.

Usage characteristics:

Ease of Application Enablement: how difficult is it to

Difficulty of instantiating the solution

What types of hardware does this solution support

Maturity

2 Background

HPC applications tend to make heavy use of specialized, highly performant network fabrics like Infiniband to shuffle data back and forth between different hosts. Further, these applications also often utilize RDMA to bypass the kernel's networking stack. Unfortunately, many container runtimes rely on the kernel's networking stack to create a *virtualized* networking stack for enforcing network isolation.

To decrease the bottleneck that is multi-machine communication, these workloads tend to heavily utilize a networking technology called Remote Direct Memory Access (RDMA); a form of kernel bypass networking that allows for access to a remote host's memory with extremely low latency and high throughput. However, container runtimes rely on the kernel's network stack to create a "virtualized" overlay network for enforcing network isolation, thus any kernel bypass networking solutions would seem to be incompatible with existing container network stacks [1]. While containers can be run without network isolation in a mode called "host networking" that allow for the direct use of the RDMA network interface inside of containers, this reduces portability and disables network isolation, eliminating many of the benefits of containers [9].

These solutions can generally categorized as either software based/paravirtualized or hardware-based. Software based/paravirtualized solutions generally either entirely emulate an RDMA NIC or use software for just the control plane of the NIC, however they tend to be more resource intensive and struggle to provide the full performance capability of the underlying hardware. Hardware based solutions rely on features built into the NIC hardware itself, however they tend not to be generalizable to commodity hardware, are less flexible, and suffer scaling issues due to limited hardware resources.

The reason for this lack of support is two-fold. First, due to a lack of configurable isolation capabilities inside of NIC hardware, container runtimes rely on "virtualized" network stacks created through the host kernel for network overlays and isolation [CITE NEEDED]. These virtualized network stacks have been shown to be both CPU intensive and significantly reduce container networking performance [1]. Second, due to the reliance on the host kernel, any form of networking that bypasses the kernel is inherently incompatible with existing container networking stacks.

3 Proposed Work

4 Schedule

References

Containers

[6] Joe Breen et al. "Building the SLATE Platform". In: *Proceedings of the Practice and Experience on Advanced Research Computing*. Association for Computing Machinery, 2018. DOI: 10.1145/3219104.3219144. URL: https://doi.org/10.1145/3219104.3219144.

[25] John Kreisa. *Introducing the Docker Index: Insight from the World's Most Popular Container Registry*. Docker. Feb. 4, 2020. URL: https://www.docker.com/blog/introducing-the-docker-index/.

[39] *SLATE Homepage*. SLATE CI. URL: https://slateci.io.

[52] A. J. Younge et al. "A Tale of Two Systems: Using Containers to Deploy HPC Applications on Supercomputers and Clouds". In: *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. 2017, pp. 74–81. DOI: 10.1109/CloudCom.2017.40.

[53] Andrew J. Younge. "Containers in HPC". In: *Workshop on NSF and DOE High Performance Computing Tools*. Sandia National Laboratories. July 11, 2019. URL: https://oaciss.uoregon.edu/NSFDOE19/talks/nsfdoe_workshop_ajy.pdf.

Container Networking

[1] Ubaid Abbasi et al. "A performance comparison of container networking alternatives". In: *IEEE Network* 33.4 (2019), pp. 178–185.

[3] Gabriele Ara et al. "Comparative Evaluation of Kernel Bypass Mechanisms for High-performance Inter-container Communications." In: *CLOSER*. 2020, pp. 44–55.

[4] Gabriele Ara et al. "On the use of kernel bypass mechanisms for high-performance inter-container communications". In: *International Conference on High Performance Computing*. Springer. 2019, pp. 1–12.

[5] Gulsum Atici and Pinar Sarisaray Boluk. "A Performance Analysis of Container Cluster Networking Alternatives". In: *Proceedings of the 2nd International Conference on Industrial Control Network And System Engineering Research*. 2020, pp. 10–17.

[10] Tyler Duzan. "How Container Networking Affects Database Performance". In: (Mar. 18, 2020). URL: https://www.percona.com/blog/2020/03/18/how-container-networking-affects-database-performance/.

[12] Anusha Ginka and Venkata Satya Sameer Salapu. *Optimization of Packet Throughput in Docker Containers*. 2019.

[21] Narūnas Kapočius. "Performance Studies of Kubernetes Network Solutions". In: *2020 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream)*. IEEE. 2020, pp. 1–6.

[26] Kyungwoon Lee, Youngpil Kim, and Chuck Yoo. "The impact of container virtualization on network performance of IoT devices". In: *Mobile Information Systems* 2018 (2018).

[27] Jiaxin Lei et al. "Tackling parallelization challenges of kernel network stack for container overlay networks". In: *11th {USENIX} Workshop on Hot Topics in Cloud Computing (HotCloud 19)*. 2019.

[40] Kyoungjae Sun, Hyunsik Yang, and Wangbong Lee. *Considerations for Benchmarking Network Performance in Containerized Infrastructure*.

[50] Miguel G Xavier et al. "Performance evaluation of container-based virtualization for high performance computing environments". In: *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE. 2013, pp. 233–240.

[55] Yang Zhao et al. "Performance of container networking technologies". In: *Proceedings of the Workshop on Hot Topics in Container Networking and Networked Systems*. 2017, pp. 1–6.

RDMA in Containers

[2] Jacob Anders et al. "RDMA Enabled Kubernetes for High Performance Computing". In: *KubeCon 2019*.

[7] Alibaba Cloud. *Using RDMA on Container Service for Kubernetes*. Feb. 20, 2019. URL: https://medium.com/@Alibaba_Cloud/using-rdma-on-container-service-for-kubernetes-c7a4484c22b5.

[18] Zhiqiang He et al. "MasQ: RDMA for Virtual Private Cloud". In: *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 2020, pp. 1–14.

[19] Yang Hu, Mingcong Song, and Tao Li. "Towards "Full Containerization" in Containerized Network Function Virtualization". In: *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*. 2017, pp. 467–481.

[22] Daehyeok Kim et al. "FreeFlow: Software-based Virtual {RDMA} Networking for Containerized Clouds". In: *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*. 2019, pp. 113–126.

[28] Coleman Link et al. "Container Orchestration by Kubernetes for RDMA Networking". In: *2019 IEEE 27th International Conference on Network Protocols (ICNP)*. IEEE. 2019, pp. 1–2.

[29] Liran Liss. *Containing RDMA and high performance computing*. 2015.

[36] Maksym Planeta et al. "TardiS: Migrating Containers with RDMA Networks". In: *arXiv preprint arXiv:2009.06988* (2020).

[48] Animesh Trivedi, Bernard Metzler, and Patrick Stuedi. "A case for RDMA in clouds: turning supercomputer networking into commodity". In: *Proceedings of the Second Asia-Pacific Workshop on Systems*. 2011, pp. 1–5.

[49] Dongyang Wang et al. "vSocket: virtual socket interface for RDMA in public clouds". In: *Proceedings of the 15th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. 2019, pp. 179–192.

RDMA Specific Research

[14] Chuanxiong Guo et al. "RDMA over commodity ethernet at scale". In: *Proceedings of the 2016 ACM SIGCOMM Conference*. 2016, pp. 202–215.

[31] Radhika Mittal et al. "Revisiting network support for RDMA". In: *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 2018, pp. 313–326.

[35] Shailesh Mani Pandey and Rajath Shashidhara. *SROCE: Software RDMA over Commodity Ethernet*.

[37] Haonan Qiu et al. "Toward effective and fair RDMA resource sharing". In: *Proceedings of the 2nd Asia-Pacific Workshop on Networking*. 2018, pp. 8–14.

[51] Jaichen Xue et al. "Fast congestion control in RDMA-based datacenter networks". In: *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*. 2018, pp. 24–26.

[54] Yiwen Zhang et al. "Performance isolation anomalies in RDMA". In: *Proceedings of the Workshop on Kernel-Bypass Networks*. 2017, pp. 43–48.

[56] Yibo Zhu et al. "Congestion control for large-scale RDMA deployments". In: *ACM SIGCOMM Computer Communication Review* 45.4 (2015), pp. 523–536.

High Performance Networking

[15] Nathan Hanford, Brian Tierney, and Dipak Ghosal. "Optimizing data transfer nodes using packet pacing". In: *Proceedings of the Second Workshop on Innovating the Network for Data-Intensive Science*. 2015, pp. 1–8.

[16] Nathan Hanford et al. “Analysis of the effect of core affinity on high-throughput flows”. In: *2014 Fourth International Workshop on Network-Aware Data Management*. IEEE. 2014, pp. 9–15.

[17] Nathan Hanford et al. “Improving network performance on multicore systems: Impact of core affinities on high throughput flows”. In: *Future Generation Computer Systems* 56 (2016), pp. 277–283.

[23] Mariam Kiran et al. “Enabling intent to configure scientific networks for high performance demands”. In: *Future Generation Computer Systems* 79 (2018), pp. 205–214.

[24] Ezra Kissel et al. “Efficient wide area data transfer protocols for 100 Gbps networks and beyond”. In: *Proceedings of the Third International Workshop on Network-Aware Data Management*. 2013, pp. 1–10.

Kernel Bypass Networking

[20] Jaehyun Hwang et al. “TCP≈RDMA: CPU-efficient Remote Storage Access with i10”. In: *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*. 2020, pp. 127–140.

[30] Michael Marty et al. “Snap: a microkernel approach to host networking”. In: *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 2019, pp. 399–413.

Documentation

[8] Docker. *Docker Documentation*. URL: https://docs.docker.com.

[9] Docker. *Use host networking*. URL: https://docs.docker.com/network/host/.

[11] The Linux Foundation. *Kubernetes Documentation*. Version v1.19. URL: https://kubernetes.io/docs/home/.

[32] NVIDIA. *NVIDIA MLNX_OFED Documentation*. Version 5.1-0.6.6.0. Aug. 5, 2020. URL: https://docs.mellanox.com/x/5pXuAQ.

[33] NVIDIA. *RDMA Aware Networks Programming User Manual*. Version v1.7. URL: https://docs.mellanox.com/pages/viewpage.action?pageId=34256560.

[38] *RDMA Core Documentation*. Version v31.1. Nov. 3, 2020. URL: https://github.com/linux-rdma/rdma-core/tree/stable-v31/Documentation.

Tutorials

[34] Itay Ozery. *Accelerating Bare Metal Kubernetes Workloads, the Right Way*. Mellanox Technologies. Nov. 10, 2019. URL: https://blog.mellanox.com/2019/11/accelerating-bare-metal-kubernetes-workloads-the-right-way/.

[41] Mellanox Technologies. *Docker RDMA SRIOV Networking with ConnectX4 / ConnectX5 / ConnectX6*. July 30, 2020. URL: https://community.mellanox.com/s/article/Docker-RDMA-SRIOV-Networking-with-ConnectX4-ConnectX5-ConnectX6.

[42] Mellanox Technologies. *Docker RoCE MACVLAN Networking with ConnectX4 / ConnectX5*. June 17, 2020. URL: https://community.mellanox.com/s/article/docker-roce-macvlan-networking-with-connectx4-connectx5.

[43] Mellanox Technologies. *How-to: Deploy RDMA accelerated Docker container over InfiniBand fabric.* June 30, 2019. URL: https://docs.mellanox.com/pages/releaseview.action?pageId=15049785.

[44] Mellanox Technologies. *How-to: Deploy RoCE accelerated Docker container.* July 30, 2019. URL: https://docs.mellanox.com/pages/releaseview.action?pageId=15049758.

[45] Mellanox Technologies. *HowTo Create Docker Container Enabled with RoCE*. Mar. 1, 2020. URL: https://community.mellanox.com/s/article/howto-create-docker-container-enabled-with-roce.

[46] Mellanox Technologies. *RDG: RoCE accelerated K8s cluster deployment for ML and HPC workloads.* June 30, 2019. URL: https://docs.mellanox.com/pages/releaseview.action?pageId=15049828.

Mellanox Infosheets

[13] Dror Goldenberg and Parav Pandit. *Mellanox Container Journey*. Mellanox Technologies, June 2019. URL: http://qnib.org/data/hpcw19/7_END_2_MellanoxJourney.pdf.

[47] Mellanox Technologies. *RDMA and RoCE for Ethernet Network Efficiency Performance*. URL: https://www.mellanox.com/products/adapter-ethernet-SW/RDMA-RoCE-Ethernet-Network-Efficiency.

Appendix

A RDMA Two-Sided Ops Diagram