Narrowing the Gap: Effects of Latency with Docker in IP Networks

Corbin Higgs
Pendleton High School
Pendleton, SC
corbinhiggs@clemson.edu

Jason Anderson - Advisor
School of Computing
Clemson University
Clemson, SC
jwa2@clemson.edu

I. Introduction

Recent work has shown that lightweight virtualization like Docker containers can be used in HPC to package applications with their runtime environments [1]. In many respects, applications in containers perform similarly to native applications [2, 3]. Other work has shown that containers can have adverse effects on the latency variation of communications with the enclosed application [4]. This latency variation may have an impact on the performance of some HPC workloads, especially those dependent on synchronization between processes [5].

In this work, we measure the latency characteristics of messages between Docker containers, and then compare those measurements to the performance of real-world applications. Our specific goals are to:

- Measure the changes in mean and variation of latency with Docker containers
- Study how this affects the synchronization time of MPI processes
- Measure the impact of these factors on real-world applications such as the NAS Parallel Benchmark (NPB).

II. Methodology

Typical Docker applications use the Linux bridge or Open vSwitch to direct traffic between network interfaces and applications in containers. To understand how applications are affected by both the software bridge and the container itself, we established four environments to conduct each benchmark:

- *native* – native application with normal access to the network
- *bridged* – native application using a veth interface attached to a Linux bridge
- *direct* – Docker application with physical interface directly assigned to the container
- *docker* – Docker application using a veth interface attached to a Linux bridge

III. Microbenchmarks

We measured the mean latency imposed by the test environments by conducting a ping-pong test between two MPI processes on separate nodes. We placed the receive side of the test in the test environment, while the sender ran natively to avoid doubling the effect. As shown in Fig. 1, *native* had the highest mean latency, while tests using extra software layers of the Linux bridge and veth interface (*bridge* and *docker*) had lower means than direct interface access.

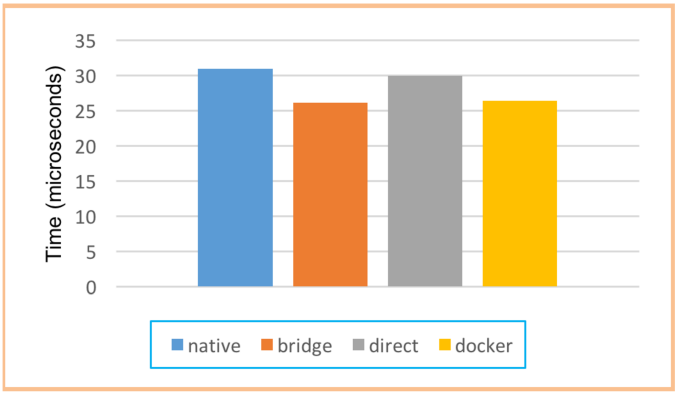

Fig. 1. Mean round trip time of MPI pingpong test.

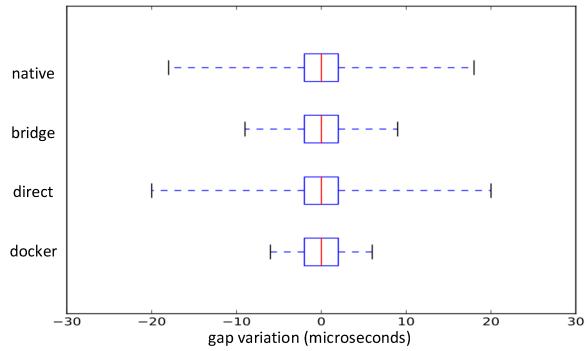

Fig. 2. 1st, 25th, 50th, 75th, 99th percentiles of send-side gap variation.

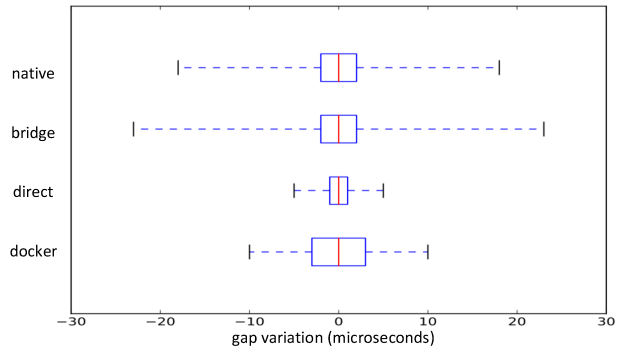

Fig. 3. 1st, 25th, 50th, 75th, 99th percentiles of receiver-side gap variation.

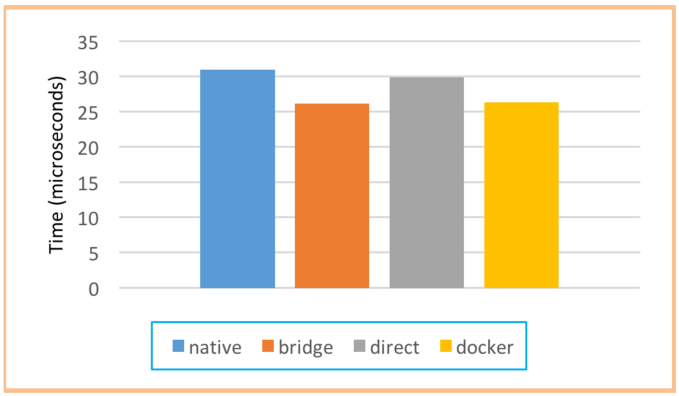

Fig. 4. Mean time to synchronize 8 nodes.

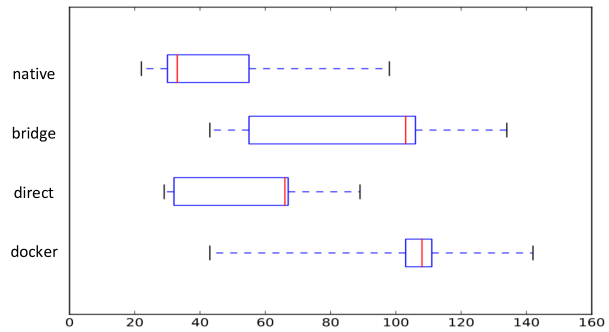

Fig. 5. 1st, 25th, 50th, 75th, 99th percentiles of synchronization time variation.

To measure the latency variation, we send a constant rate stream of messages between MPI processes on separate nodes. To help determine where variation occurs, we further divide the tests into send-side and receive-side variation. We then calculate and report this inter-message spacing in Figs. 2 and 3.

On the sending side, using the Linux bridge seemed to decrease latency variance. On the receiving side, however, the opposite seems to be true; the Linux bridge increased the latency variance. However, a Docker container had the effect of decreased variance in both network configurations.

IV. Synchronization

For these tests, we measured the time required for 8 MPI processes on separate nodes to synchronize to an MPI_Barrier() call. Unlike the microbenchmarks, all nodes were under the testing environment. We report the mean and distribution of times in Figs. 4 and 5.

As opposed to the microbenchmarks, *native* and *direct* had lower means and variation, while measurements in environments using the Linux bridge were more variable.

We also observe that the environments using containers have a higher mean synchronization time, despite the lower receive-side latency variation and equivalent mean and send-side latency variation of the microbenchmarks.

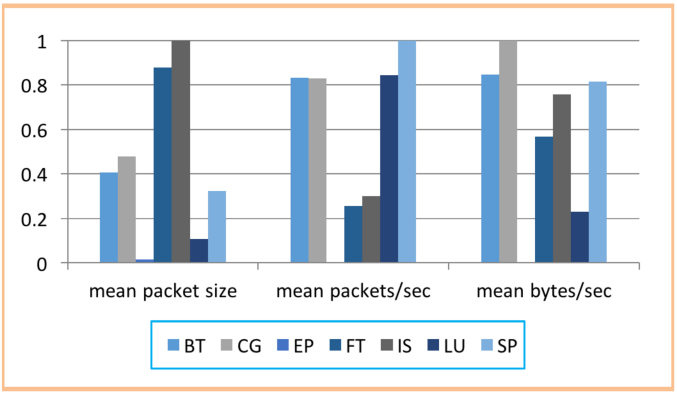

Fig. 6. Normalized comparison of data transfer characteristics.

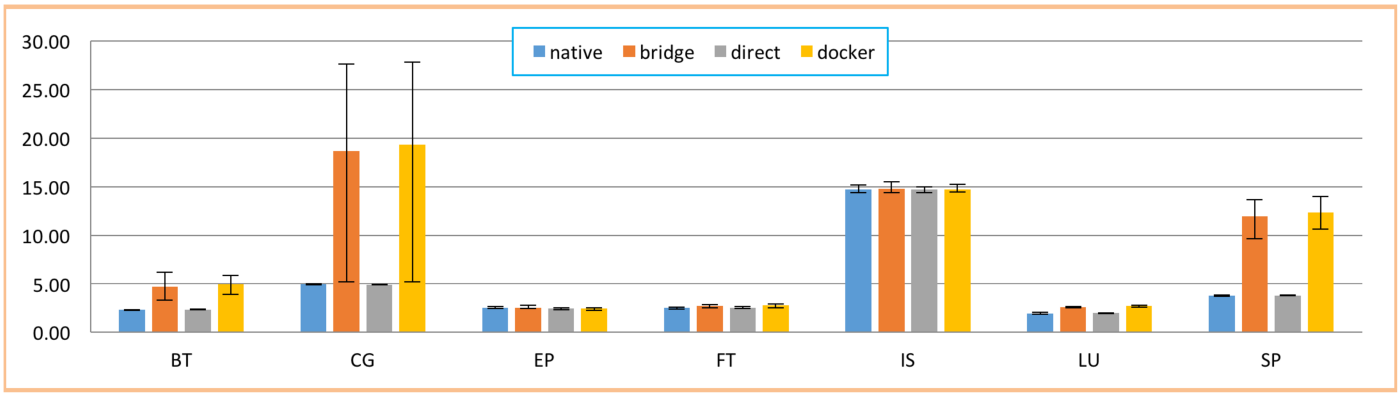

Fig. 7. Minimum, mean, and maximum runtimes of NPB benchmarks.

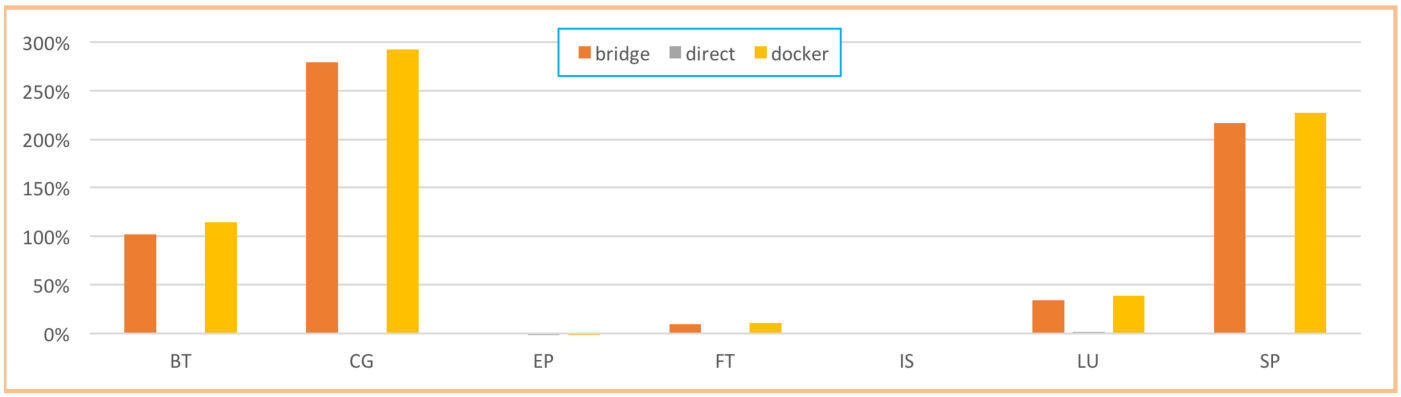

Fig. 8. Percent difference from *native* for NPB benchmarks.

V. Application Benchmarks

We ran the NAS Parallel Benchmark with seven of the nine benchmarks compiled with problem sizes appropriate for a 160 core cluster. First, we measured the total number of packets that were transmitted between nodes for each program, then ran 30 iterations of each benchmark with processes distributed evenly across eight nodes.

To estimate how dependent each benchmark is on communications, we compare each benchmark's mean packet size and transmission rate as compared to *native* run times in Fig. 6. To estimate the impact of each environment on runtime, we compare each test's min, max, mean, and difference from *native* mean in Figs. 7 and 8.

As with the synchronization tests, *bridge* and *docker* added a significant amount to the mean and variation on BT, CG, LU, and SP. We observe that tests with lower packet transmission rates are less affected by the bridge and veth interface.

VI. Discussion

Although we observed in the microbenchmarks that the Linux bridge and veth interface lowered the mean and sender-side variation of message latency between MPI applications, we saw that synchronization time increased for the same test environments. Furthermore, we see that application benchmarks that send many packets per second are more affected by the extra software switch than those with direct access to the network interface.

We hypothesize that a buffer in the bridge or veth software layers may be small enough to cause packets to be dropped, causing TCP resends to occur and lower the overall performance of applications that send bursts of packets. This would explain the performance drops for MPI_Barrier and the NPB benchmarks BT, CG, LU, and SP when the Linux bridge is used.

REFERENCES

[1] Higgins, Joshua, Violeta Holmes, and Colin Venters. "Orchestrating Docker Containers in the HPC Environment." In International Conference on High Performance Computing, pp. 506-513. Springer International Publishing, 2015.

[2] Xavier, Miguel G., Marcelo V. Neves, Fabio D. Rossi, Tiago C. Ferreto, Timoteo Lange, and Cesar AF De Rose. "Performance evaluation of container-based virtualization for high performance computing environments." In 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 233-240. IEEE, 2013.

[3] Ruiz, Cristian, Emmanuel Jeanvoine, and Lucas Nussbaum. "Performance evaluation of containers for HPC." In European Conference on Parallel Processing, pp. 813-824. Springer International Publishing, 2015.

[4] Anderson, Jason, Hongxin Hu, Udit Agarwal, Craig Lowery, Hongda Li, and Amy Apon. "Performance considerations of network functions virtualization using containers." In 2016 International Conference on Computing, Networking and Communications (ICNC), pp. 1-7. IEEE, 2016.

[5] Martin, Richard P., Amin M. Vahdat, David E. Culler, and Thomas E. Anderson. Effects of communication latency, overhead, and bandwidth in a cluster architecture. Vol. 25, no. 2. ACM, 1997.

[6] Cloudlab. http://cloudlab.us/.