HPC container runtime performance overhead: At first order, there is none

Alfred Torrez, Reid Priedhorsky, Timothy Randles
{atorrez,reidpr,trandles}@lanl.gov
Los Alamos National Laboratory
High Performance Computing Division

1 INTRODUCTION

HPC centers are facing increasing demands for greater software flexibility to support faster and more diverse innovation in computational scientific work. Containers, which use Linux kernel features to allow a user to substitute their own software stack for that installed on the host, are an increasingly popular method to provide this flexibility. Because standard container technologies such as Docker are unsuitable for HPC, three HPC-specific technologies have emerged: Charliecloud [8],[1] Shifter [5], and Singularity [6].

A common concern is that containers may introduce application performance overhead. This question has been addressed a modest number of times in the literature; however, prior work has not tested a broad set of HPC container technologies on a broad set of benchmarks. For example, our previous work contains a basic performance test of Charliecloud [8], and Brayford et al. tested Charliecloud on machine learning applications [2]. Jacobsen and Canon evaluated application load times but not performance [5]. Several studies evaluated performance of Singularity [7, 10, 11], perhaps also with non-HPC technologies such as LXC [12]. Others evaluate non-HPC container technologies [1, 3, 4].

This poster[2] addresses the gap by comparing performance of the three HPC container implementations and bare metal on multiple dimensions (CPU performance, memory performance, application performance, and memory usage) using industry-standard benchmarks (SysBench, STREAM, and HPCG).

We found no meaningful performance differences between the four environments with the possible exception of modest variation in memory usage. This is broadly consistent with prior results.

This result suggests that HPC users should feel free to containerize their applications without concern about performance degradation, regardless of the container technology used. It is an encouraging development on the path towards greater adoption of user-defined software stacks to increase the flexibility of HPC.

2 EXPERIMENTS

We ran our tests on LANL's CTS-1 clusters Grizzly (1490 nodes, 128 GiB RAM/node; HPCG) and Fog (32 nodes, 256 GiB RAM/node; SysBench, STREAM, and HPCG). Our methods and results follow, organized by evaluation dimension.

2.1 CPU performance

SysBench[3] is a suite of microbenchmarks to measure various aspects of performance. We used it to time a 36-way threaded computation of the prime numbers below 40 million. We ran 100 tests for each of the four environments (bare metal and the three container technologies) in a round-robin fashion to balance out potential node problem impacts on just one technology.

Median performance across all four technologies occupied a spread of only 0.07% (129.27–129.36 seconds), while the spread between minimum and maximum of all 400 runs was 0.4% (128.99–129.55 seconds). All four technologies were essentially equal in performance.

kernel	bare metal	Charliecloud	Shifter	Singularity
copy	10.28	10.29	10.28	10.29
scale	10.45	10.45	10.45	10.46
add	11.16	11.16	11.16	11.16
triad	11.22	11.22	11.21	11.22

Table 1: Median STREAM performance in GiB/s.

2.2 Memory performance

STREAM[4] measures sustained memory bandwidth and computation rate for four simple memory-bound vector kernels. We ran 100 single-threaded tests per technology. We compiled with STREAM_ARRAY_SIZE set to 2 billion to match the recommended 4× cache and pinned the process to a semi-arbitrary core using the Slurm argument --cpu_bind=v,core,map_cpu:23.

Median performance across all four technologies and all four kernels occupied a spread of 0.07%; see Table 1. Minimum performance occupied a spread of 0.1%, and maximum 0.4%. All four technologies were again essentially equal in performance.

2.3 Application performance

The High Performance Conjugate Gradients (HPCG) Benchmark[5] is a relatively new benchmark developed as a complement to the High Performance Linpack (HPL) benchmark but enhanced to more accurately reflect current HPC applications. We used a cube dimension of 104 and a run time of 60 seconds, all 36 cores per node, one MPI rank per core, and one thread per rank.

We ran these tests in two phases. In the first, we ran HPCG on power-of-two node counts from 1 to 512, with 4–10 runs per condition (due to bad nodes). In the second, we ran HPCG on power-of-two node counts from 1 to 32 with 50 runs per condition. All environments showed extremely linear weak scaling, i.e., there is no performance degradation as the problem size and number of nodes increases.

[1] Disclosure: The authors are members of the Charliecloud team.
[2] This work is described in more detail in our archival workshop paper [9].
[3] https://github.com/akopytov/sysbench/
[4] https://www.cs.virginia.edu/stream/
[5] http://www.hpcg-benchmark.org/

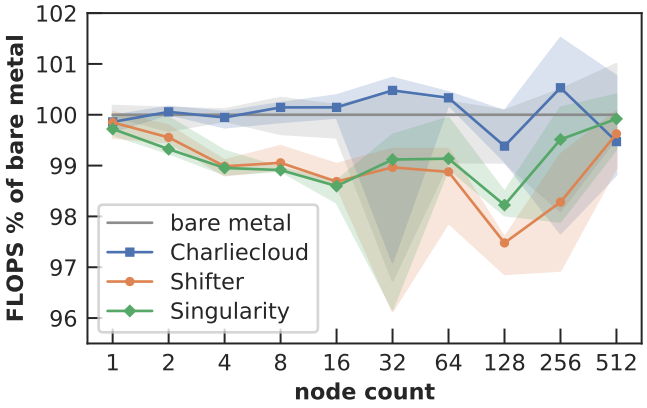

Figure 1: HPCG Phase 1: 4–10 runs per condition.

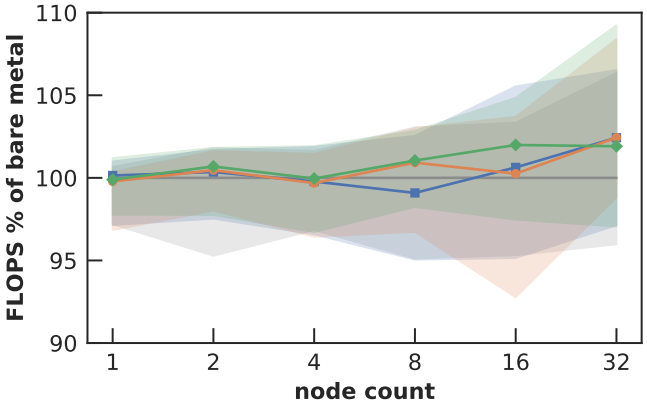

Figure 2: HPCG Phase 2: 50 runs per condition.

Figure 1 shows performance in the first phase as percent of bare metal median. Performance is all close to bare metal, within a few percent, but not as close as the prior two tests. Because bare metal and Charliecloud formed one cluster, and Shifter and Singularity a second, we wanted to do a larger number of runs to see if this was an artifact.

Figure 2 shows second-phase performance. The three container technologies track one another very closely. Bare metal is slightly slower (about 2%) at 32 nodes. We believe, however, that this is an artifact of subtle shared library linking differences between bare metal and the three container environments. That is, these results are again consistent with zero performance difference between any of the technologies.

2.4 Memory usage

To understand node memory usage with STREAM, we computed MemTotal – MemFree from /proc/meminfo, sampled at 10-second intervals.

Bare metal total node usage was a median of 50.8 MiB. Charliecloud added 1200 MiB, Shifter 16 MiB, and Singularity 37 MiB. We believe that Charliecloud's memory cost is mostly its 1.2 GiB image stored in a tmpfs; Charliecloud's alternate image workflow that mounts SquashFS image files rather than unpacking should eliminate this.

We also evaluated memory usage of the benchmark itself by subtracting the baseline sample taken before STREAM started. Under this view, median bare metal usage was 43.9 GiB. Charliecloud added 66 MiB, Shifter 970 MiB, and Singularity 330 MiB. It is not yet clear to us why the ordering reverses.

For HPCG, we sampled at 10-second intervals the writeable/private field of pmap(1), which reports memory consumption of individual processes. Median memory usage for all three container technologies is, to two significant figures, 0.64% lower than bare metal at 1 node, 0.53% lower at 8 nodes, 0.53–0.54% lower at 64 nodes, and 1.2% higher at 512 nodes, a minimal difference.

The container technologies do seem to add a small memory overhead in ancillary processes, not the application, up to about 2%.

3 DISCUSSION

We evaluated performance of the three key HPC container technologies using multiple industry-standard benchmarks. We found that performance impact of containers is minimal to nonexistent. These results can reassure users that the flexibility gained by using containers does not come at the cost of performance, and that performance need not be a factor in choosing a container technology.

We emphasize that our work is a first glance, and there is more ahead. For example, many of our experiments showed some quirks that require deeper analysis, and a wider set of performance tests is necessary, most notably filesystems, networking, MPI, and real applications.

ACKNOWLEDGEMENTS

This work was supported in part by the Advanced Simulation and Computing Program; the Exascale Computing Project under project 17-SC-20-S; and the LANL Institutional Computing Program, which is supported by U.S. Department of Energy National Nuclear Security Administration under contract 89233218CNA000001. LA-UR-19-27967.

REFERENCES

[1] Ashijeet Acharya et al. 2018. A performance benchmarking analysis of hypervisors containers and unikernels on ARMv8 and x86 CPUs. In *Proc. EuCNC*.
[2] David Brayford et al. 2019. Deploying AI frameworks on secure HPC systems with containers. In *Proc. HPEC*.
[3] MinSu Chae, HwaMin Lee, and Kiyeol Lee. 2017. A performance comparison of Linux containers and virtual machines using Docker and KVM. *Cluster Computing* (2017).
[4] Wes Felter et al. 2015. An updated performance comparison of virtual machines and Linux containers. In *Proc. ISPASS*.
[5] Douglas M. Jacobsen and Richard Shane Canon. 2015. Contain this: Unleashing Docker for HPC. In *Proc. CUG*.
[6] Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. 2017. Singularity: Scientific containers for mobility of compute. *PLOS ONE* (2017).
[7] Emily Le and David Paz. 2017. Performance analysis of applications using Singularity container on SDSC Comet. In *Proc. PEARC*.
[8] Reid Priedhorsky and Tim Randles. 2017. Charliecloud: Unprivileged containers for user-defined software stacks in HPC. In *Proc. SC*.
[9] Alfred Torrez, Tim Randles, and Reid Priedhorsky. 2019. HPC container runtimes have minimal or no performance impact. In *Proc. CANOPIE HPC Workshop @ SC*.
[10] Yinzhi Wang, R. Todd Evans, and Lei Huang. 2019. Performant container support for HPC applications. In *Proc. PEARC*.
[11] Andrew J. Younge et al. 2017. A tale of two systems: Using containers to deploy HPC applications on supercomputers and clouds. In *Proc. CloudCom*.
[12] Ákos Kovács. 2017. Comparison of different Linux containers. In *Proc. TSP*.