# Performance Analysis of Containerized OrangeFS in HPC Environment

Izzet Yildirim [1] Meng Tang [1] Anthony Kougkas [1] Xian-He Sun [1]

[1] Illinois Institute of Technology

SCALABLE COMPUTING SOFTWARE LABORATORY

ILLINOIS INSTITUTE OF TECHNOLOGY

## Introduction

Containerization is on the rise in cloud computing due to its benefits in reproducibility, portability, and its lightweight properties. The need for scalable and reproducible products and the necessity to interoperate on everything from local machines to large scale HPC resources have also made the HPC solutions adopt containers expeditiously [1]. However, the necessity of developing HPC-focused containers appeared from various scaling and security issues of general-purpose containers such as Docker [2]. As a result, three such containers have emerged: Singularity, Shifter, and Charliecloud. While existing studies suggest that these specialized containers create very low to negligible overhead for HPC applications [3], they mostly focus on CPU and memory performance rather than I/O specific analysis.

In this paper, we specifically examine the I/O performance of different container solutions on OrangeFS, a parallel storage system. Our analysis shows that Singularity covers almost all aspects, from mobility to answering security concerns, with an almost perfect score. It performs nearly identical to bare metal within a single server-single client setup. On the other hand, Linux Containers (LXC) fails at I/O performance by producing almost two times more I/O requests than the bare metal, whereas Docker registers unnecessary amount of network packets under the same workload. Our aim is to provide such insights to help HPC practitioners so that they can obtain the best I/O performance when running HPC container solutions.

## Approach

1. **OrangeFS Deployment**: As the goal of this study is to lay the groundwork for more detailed experiments, we focus on employing a single server-single client OrangeFS setup rather than quickly ramping up to advanced setups that have many OrangeFS nodes. We assign the same OrangeFS server as both data and metadata manager, thus no data replication in our setup.
2. **Container Deployment**: We deploy five containers as part of our study: Singularity, Docker, LXC, Podman and Charliecloud. Each container makes use of a definition file in that we can define the installation scripts and the dependencies that our application needs. All containers are built on top of a lightweight CentOS image and follow very similar steps while building the deployable OrangeFS images. We then deploy these images to our HPC setup and directly interact with the containerized OrangeFS applications through the images.
3. **OrangeFS Through Containers**: For HPC-focused containers, we only used bind-mount for container storage setup. Since we rely on the kernel integration of OrangeFS to unleash its full potential, bind-mount is the only option available for Docker that allows us to mount specialized file systems (e.g. PVFS2) onto containers.

## References

[1] G. M. Kurtzer, V. Sochat, and M. W. Bauer, "Singularity: Scientific containers for mobility of compute," *PLoS One*, vol. 12, no. 5, 2017.

[2] R. Priedhorsky and T. Randles, "Charliecloud: unprivileged containers for user-defined software stacks in hpc," *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, no. 36, pp. 1–10, 2017.

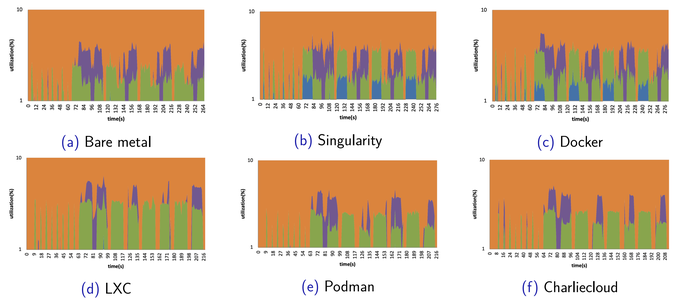
[3] A. J. Younge, K. Pedretti, R. Grant, and R. B. Brightwell, "A tale of two systems: Using containers to deploy hpc applications on supercomputers and clouds.," 11 2017.

[4] H. Shan and J. Shalf, "Using ior to analyze the i/o performance for hpc platforms," 6 2007.

## Evaluation

1. **Methodology**: To evaluate the container solutions, we first conducted experiments on a bare OrangeFS server-client setup without any virtualization. We use the results of these experiments as our base performance metrics. Next, to understand the impacts of virtualization, we employ containerized servers one at a time and measure the performance of the host machine on CPU and memory utilization, and disk I/O and network throughput. For capturing the performance precisely, we installed Intel's Performance Analysis Tool on both server and client nodes and configured the client to trigger recording when it starts running the experiment workload.
2. **Workload**: Our experimental setup essentially comprises a server and a client of a parallel storage system. The workload we used in our experiments consists of two different files that are fed into our OrangeFS setup via a benchmark tool called IOR. We selected IOR due to its accuracy in measuring the performance of an I/O application [4] and its nature of being suitable for parameterization through a configuration file. Two types of workloads that are run in 8 repetitions:
   - 1GB workload with 1MB transfers in 16MB blocks over 64 segments
   - 8GB workload with 1MB transfers in 64MB blocks over 128 segments
3. **Testbed**: Chameleon is our experiment platform and it provides a configurable environment for large-scale cloud research. We specifically used its compute nodes called "Haswell", with each having dual Intel® Xeon® E5-2670 CPU running at 2.3GHz with a total of 24 cores, and 128GB of RAM. The compute nodes are connected to each other via a 10 Gbps Ethernet network.

## Results



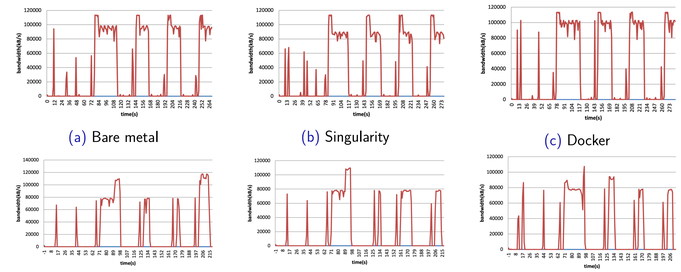

(a) Bare metal (b) Singularity (c) Docker

(d) LXC (e) Podman (f) Charliecloud

Figure 1: CPU Utilization

Figure 1 shows the CPU utilization with average values of idle (orange), steal (blue), iowait (purple), and nice priority in user space (green). They all seem to behave similarly in how they use CPU against all workloads. We merely observe small spikes on Docker and LXC, but they may be interpreted as a sign of potential risk. Podman seems to produce the less overhead under various workloads.

## Results

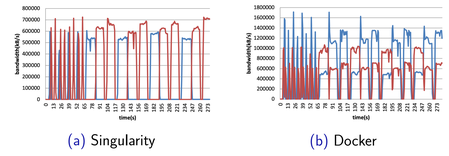



(a) Bare metal (b) Singularity (c) Docker

(d) LXC (e) Podman (f) Charliecloud

Figure 2: Disk I/O Throughput

Figure 2 shows the sum of the total number of requests at a given time. Docker outperforms other containers by producing more throughput in a longer time. LXC seems to reach larger numbers from time to time but the overall result seems to be inconsistent and noisy. Singularity performs similarly to bare metal with barely 10% overhead.




(a) Singularity (b) Docker

Figure 3: Network Bandwidth

Figure 3 shows plot of network bandwidth comparison between Singularity and Docker. Bandwidth for read is in red and for transmit is in blue. Apart from Docker, all other solutions behave similarly in terms of network utilization and they seem to use a similar amount of bandwidth against all workloads. Docker seems to consume almost twice network bandwidth for the same workload.

## Conclusions

We have presented our empirical analysis on the I/O performance of container solutions for HPC environments with their comparison to bare metal. We studied five container solutions: Singularity, Docker, LXC, Podman and Charliecloud. Our evaluations show that Singularity suits such environments and overall performs better than other container solutions. More importantly, our study proves that OrangeFS can work within any container and can perform the tasks that it was designed to do and that we can still interact with OrangeFS and work with its API while it is hosted in a container. This is promising and opens ways to further explore the area and think about the possibilities.



iyildirim@hawk.iit.edu