Containerized Filesystems : A Performance Investigation of Containers across Filesystems

Jia R. Wu
University of Waterloo

Jichen Zhao
University of Waterloo

Abstract

Your Abstract Text Goes Here. Just a few facts. Whet our appetites.

Container as a Service (CaaS) platforms are being offered to personal and enterprise applications, In this paper, we attempt to profile the performance of three separate filesystems, ext4, reiserfs, and xfs running ontop of Arch Linux.

1 Introduction

Filesystems are the implementation of specific rules and principles governing how data is stored and retrieved on storage medium. At the time of writing, there are multiple various filesystems supported on the Linux operating system including ext4, reiserfs, xfs and many more. These filesystems all contain central concepts such as blocks and inodes, but differ subtly in their implementations [2].

Containers, also known as standardized units of software, are executables built ontop of a kernel. These containers offer a standardized execution environment agnostic to the environment the container is installed on. In this paper, we chose to examine Docker, a popular and free platform for creating and running containers. Docker has been championed by researchers as a platform which allows for reproducible research [1]. We are interested in researching whether or not container based filesystem operations incur a significant performance overhead relative to native filesystem operations.

We make the following contributions:

- A comparison of ext4, reiserfs and xfs filesystems using Filebench and IOzone.
- An investigation of filesystem performance in containers versus native linux.

2 Methods

2.1 Environment

Jason Add Description of Your Computer Here

2.2 Docker

Docker version 17.06.2-ce, API version 1.3, Go version go1.8.3, and git commit cec0b72 were used in this project. The base container was pulled from base/archlinux. All relevant code to regenerate the docker container and environment can be found in Section 6.

2.3 Benchmarks

Filebench, a popular filesystem and storage benchmark for I/O profiling is used to profile our three filesystems, as it permits custom and scalable workloads. We utilized five workloads which shipped with Filebench: FILESERVER, VARMAIL, VIDEOSERVER, WEBPROXY and WEBSERVER.

Each of these benchmarks were called 5 times after a fresh installation of the filesystem. Workloads generate temporary files prior to execution. Temporary files were discarded with "rm -rf" after each benchmark prior to running the next benchmark to clear the cache. The runtime of Filebench benchmarks were fixed at 300 seconds with the exception of WEBPROXY which was fixed at 60 seconds. This was to mitigate any ramp-up effects that might occur with our HDD. We had to disable address space layout randomization by setting randomize_va_space=0 in /proc/sys/kernel so that Filebench would execute properly and give consistent results. Unless explicitly mentioned, the benchmarks use the predefined settings given by the authors[1].

2.3.1 FILESERVER

This workload creates a directory tree, and calls a sequence of creates, deletes, appends, reads and writes on various files. The authors of Filebench describe this workload as being similar to SPECsfs. We configured the FILESERVER workload to output files of 1.2GB in size, with one flow for each stat, delete, read, close, append, open, close, write, and create operations.

2.3.2 VARMAIL

In this benchmark, multiple operations of create-append-sync, read-append-sync, read and delete are run in a single directory. The purpose of these operations are to simulate I/O operations on a mail server.

2.3.3 VIDEOSERVER

This workload simulates a videoserver by serving video files from one directory and caching videos in a second directory. Videos were configured to be 3.2GB in size.

2.3.4 WEBPROXY

WEBPROXY simulated I/O on a web proxy server. Multiple files were created, written to, closed, and deleted in parallel. This benchmark was configured to run only for 60 seconds as 300 seconds would cause a segfault. The authors of Filebench are aware of this issue on Github.

2.3.5 WEBSERVER

This benchmark is similar to WEBPROXY, however it runs open-read-close operations on files in a directory tree and has an additional step of appending.

3 Results

3.1 Filebench

Figures 1 & 2 both depict the results Filebench workloasd on each filesystem. Figures 3 & 4 are relative performance graphs of ext4, reiserfs and xfs on Arch Linux and Docker respectively. As illustrated by these figures, ext4 outperforms reiserfs across the board, and outperforms xfs with the exception of the WEBSERVER benchmark. Since ext4 is the highest performing filesystem, results are scaled to it in the relative performance graphs. Values less than 1 mean that the throughput on those filesystems performed worse than on ext4.

When examining the performance of Filebench benchmarks in the Docker container, the conclusions are identical for ext4. When comparing reiserfs and xfs however, there are some interesting differences. The VARMAIL and VIDEOSERVER benchmarks are much more close in terms of performance.

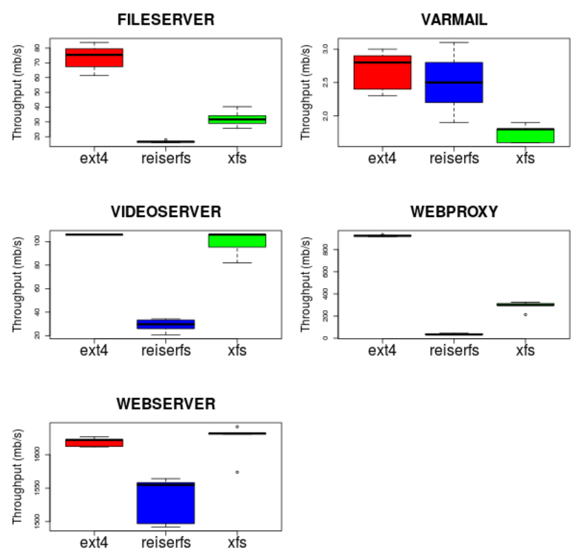

Figure 1: Native Filebench performance per workload.

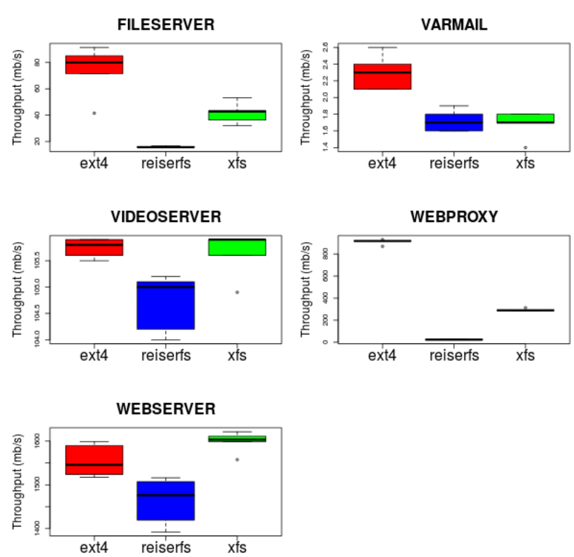

Figure 2: Docker Filebench performance per workload.

4 Discussion

4.1 Threats To Validity

Filebench is not fully representative of a filesystem's performance. Filebench primarily evaluates the I/O performance of a disk, and is not tuned to evaluate the caching

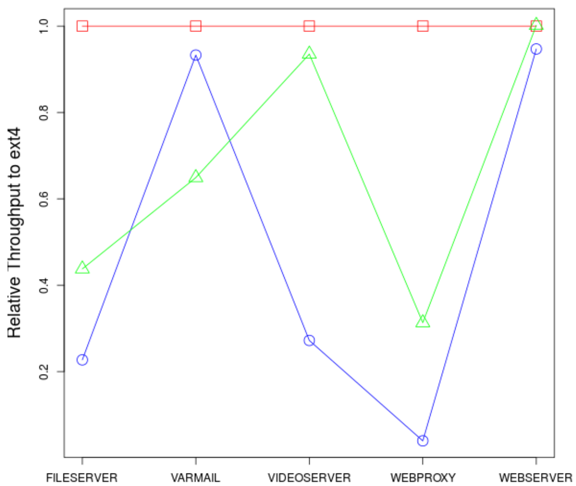

Figure 3: Relative workload performance on native Arch Linux. Red squares are ext4, blue circles are reiserfs and green triangles are xfs. Throughput is scaled to ext4. Higher numbers mean higher throughput relative to ext4.

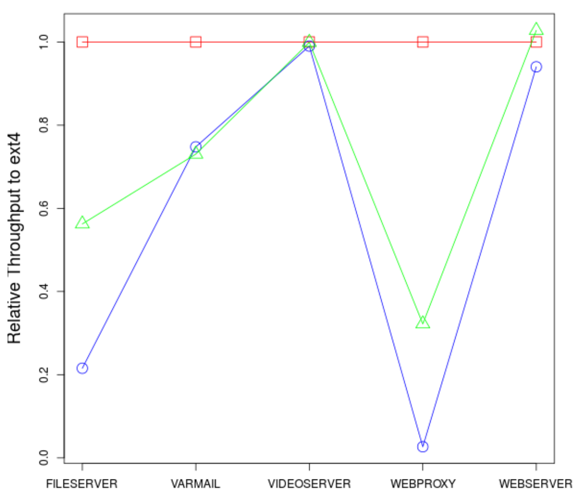

Figure 4: Relative workload performance within Docker. Red squares are ext4, blue circles are reiserfs and green triangles are xfs. Throughput is scaled to ext4. Higher numbers mean higher throughput relative to ext4.

and metadata dimensions of a disk [3]. We observed a peculiarity in our results, such as how VARMAIL and VIDEOSERVER results differ greatly in our Arch Linux environment, but are very similar in the Docker environment. This suggests that there is some factor affecting our results. Thus, we refrain from making absolute claims that one filesystem is better than another, and only provide suggestions.

	ext4	reiserfs	xfs
FILESERVER	1.01	0.95	1.29
VARMAIL	0.86	0.69	0.97
VIDEOSERVER	1.00	3.64	1.07
WEBPROXY	0.98	0.66	1.01
WEBSERVER	0.96	0.95	0.99

Table 1: Benchmark results on Docker scaled to native Arch Linux. Lower values mean Docker benchmarks performed worse.

5 Acknowledgments

We thank Dr. Tim Brecht for providing guidance about the project such as recommending papers relevant to benchmarking filesystems.

6 Availability

All relevant scripts and data can be retireved from the following repository:

`https://github.com/JRWu/fall2017_cs854/`

Notes

[1]The predefined settings can be located here: https://github.com/filebench/filebench/wiki/Predefined-personalities

References

[1] BOETTIGER, C. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review 49*, 1 (2015), 71–79.

[2] PROJECT, T. L. D. The linux system administrators guide 5.10.filesystems.

[3] TARASOV, V., BHANAGE, S., ZADOK, E., AND SELTZER, M. Benchmarking file system benchmarking: It* is* rocket science. In *HotOS* (2011), vol. 13, pp. 1–5.