Performance Evaluation of Docker

Yuguang Zhang
University of Waterloo

July 21, 2016

Abstract

Virtualization of operating systems provides a common way to deploy complex applications to a cloud environment. A new form of virtualization based on containers promises to offer better performance than traditional approaches with a virtual machine. This paper explores a method of running performance evaluation experiments using docker and compares container based virtualization with native performance. Several benchmark tools are used to measure overhead in terms of processing, storage, memory and network. [Summary of results]

1 Introduction

2 Overview of Docker

Docker is a platform that allows developers to "build, ship and run any app, anywhere". It is now the standard solution to software deployment, which is a costly and fault prone process.

One way to understand Docker is through a metaphor of shipping physical goods. Before containers for shipping goods were invented, special handling was applied to different types of goods. Moreover, goods were shipped from sea to air, road to sea, sea to railroad, railroad to sea, and more combinations. For each of those cases, goods were handled differently. As a result, shipping was an expensive and difficult problem. To solve the complex problem of shipping goods cheaply to all locations, shipping containers were invented. These standardized containers had specific dimensions and weight. Ships and trucks were designed to carry, load, and unload these predictably shaped items. Different types of goods are packaged inside a container, and it doesnt matter to the carrier what types of goods are in the container. Docker does the same thing for software rather than physical goods. Docker packages software dependencies and libraries in a standard format so that they are deployable anywhere.

2.1 How Docker Compares to VMs

Docker containers are quicker to spin up and deploy to machines. Since Docker uses a layered filesystem, only the changes to the filesystem rather than an entire operating system is shipped. As a result, Docker containers are lightweight to deploy and make changes to. In addition, Docker is entirely a command line utility, which makes it scriptable. Unlike starting and stopping a virtual machine through GUI management consoles, it is relatively easy to automate starting and stopping Docker containers.

Docker containers are lightweight and more scalable than virtual machines for running multiple instances on a single physical machine. In a scalability comparison done by Ann Mary Joy [2], containers scaled up 22 times faster than virtual machines for a load balanced WordPress application. In this experiment, the AWS EC2 auto scaling feature is used. Auto scaling is triggered when the CPU utilization goes above 80%. Jmeter is used to send concurrent requests to WordPress. Spinning up a new container to load balance the requests took only eight seconds compared to three minutes for a new virtual machine.

3 Related Works

Morabit et al. [4] compared hypervisor and container performance with running applications in a native environment. They used the Y-cruncher, NBENCH, and Linpack to measure CPU performance under KVM, Docker, LXC, and OSV. For disk I/O performance, sequential reads and writes of 25 GB was tested with Bonnie++. The results show that disk I/O efficiency is still a bottleneck with KVM. LXC and Docker introduce negligible overhead, though there is a trade off in terms of security.

Felter et al. [3] compare KVM and Docker performance with running applications in their native environment. In their benchmark, the Sysbench oltp benchmark is ran against a single instance of MySQL. The benchmark compared MySQL throughput in terms of transactions per second on Docker with an AUFS volume, Docker with normal networking using NAT, and Docker using host networking and a mounted volume, native Linux, and KVM. The results showed that MySQL transactions under KVM qcow were about 30% compared to native. Docker volumes have noticeably better performance than files stored in AUFS. They also found that NAT introduces overheads for workloads with high packet rates.

Xavier et al. [5] analyzed MapReduce clusters and in HPC Environments comparing and contrasting the current container-based systems including Linux VServer, OpenVZ and Linux Containers (LXC). They evaluated the performance of MapReduce jobs on a cluster running different container systems. The results showed that all container-based systems offered a near-native performance. They also found that LXC was closest to native performance.

4 Methodology

List of software and hardware specs

5 Benchmark Results

This section presents the results of the analysis. The benchmarks measure CPU, Memory, Disk I/O, and Network I/O performance.

5.1 CPU Performance

The Pi calculation

5.2 Disk I/O Performance

5.3 Memory Performance

6 Discussion

7 Conclusions and Future Work

7.1 Integration with Datamill

Datamill workers have three standard partitions: boot, controller, and benchmark. The boot partition is for the kernel and initramfs, the controller partition holds a barebones Gentoo installation, and the controller partition is cloned to run benchmarks in the benchmark partition. As a result, the allocated disk space for the controller partition is minimal to give as much space to run experiments as possible on the benchmark partition.

Docker stores data such as images and containers in the users directory on disk. If the containerized experiments were run on the controller partition, the file system on the controller partition will inevitably run out of disk space. Therefore, the proposed way to run containerized experiments is by starting the Docker daemon with an argument to specify its directory. For example to run Docker from `/mnt/benchmark`, the command to start the Docker daemon is `docker -g /mnt/benchmark -d`. This will create a set of files and folders internal to Docker in `/mnt/benchmark`. After running an experiment files and folders in `/mnt/benchmark` are deleted, removing the containers and images.

References

[1] Kopytov, Alexey. "SysBench: A System Performance Benchmark, 2004."

[2] Joy, Ann Mary. "Performance comparison between linux containers and virtual machines." Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in. IEEE, 2015.

[3] Felter, Wes, et al. "An updated performance comparison of virtual machines and linux containers." Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On. IEEE, 2015.

[4] Morabito, Roberto, Jimmy Kjllman, and Miika Komu. "Hypervisors vs. lightweight virtualization: a performance comparison." Cloud Engineering (IC2E), 2015 IEEE International Conference on. IEEE, 2015.

[5] Xavier, Miguel Gomes, Marcelo Veiga Neves, and Cesar Augusto Fonticielha De Rose. "A performance comparison of container-based virtualization systems for mapreduce clusters." 2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. IEEE, 2014.