Comparison of Firecracker VM and Docker container with deploying a machine learning inference model.

Amirhossein Bahrami

Ah.bahrami@hotmail.com

Task #3: Firecracker is one of the coolest new technologies in computer systems today. It allows you to spin-up a VM in less than 500ms, even faster than a container. In this task, your task is to start and run a machine learning inference model of your choice on a firecracker VM. You will then run docker locally on your computer and deploy the same model in a docker container.. You will then benchmark the performance of the algorithm, focusing on how the underlying system affects the performance of model-serving. Please note, we only care about CPU performance now with no need to consider GPUs.

Contents

Introduction

In this task the aim was to compare the performance of Firecracker VM (FC) and a Docker container (DC) by running a Machine Learning (ML) inference model on both. The model that was implemented was a simple model in order to not cause a problem while running, on poor hardware specifications. Also running a complex model compare to the current model will just consume more resources during importing necessary libraries and the runtime of the code will increase which does not effect on how the outlined VMs will run the code.

It is essentially similar to QEMU but with some features being taken away. Some of the features are:

- No GUI
- It is written on Rust rather than C
- It is intended to be more secure against buffer overflows vs QEMU

A brief explanation regarding the Firecracker VM:

What is FC

Firecracker is an open-source virtualization technology that is purpose-built for creating and managing secure, multi-tenant container and function-based services that provide serverless operational models. Firecracker runs workloads in lightweight virtual machines, called microVMs, which combine the security and isolation properties provided by hardware virtualization technology with the speed and flexibility of containers

Cons of FC

There is no easy way of installing an OS on it with having the ISO. They developers proposed a demo (getting Started) which installs Alpine on FC, but during this research the aim was to find out if it is possible to install any arbitrary OS on it.

The Alpine which developers proposed for FC is very simple and does not have anything on it to do something meaningful with FC. So essentially if you want to run any ML model on it, you need to install everything from scratch which requires more time.

Methodology

In this section, an explanation regarding the implementation of both FC and the DC will be presented. Each section will discuss the procedure of creating and running the FC and DC.

Building and Running FC

1. First, we downloaded our arbitrary OS, in this research CentOS 8, and create a "QEMU" drive of it. Then we can install our OS on this drive via the ISO file of our arbitrary OS.

In this research a minimal installation of the CentOs Stream 8 was taken into account.

2. A custom Kernel has been created. The reason is that the Kernel that comes with CentOs does not have the necessary configurations to run inside FC. These configurations have been taken from the GitHub repository of the FC.

After we create a menuconfig [make menuconfig] we can then create our kernel with [make vmlinux].

3. In order to run the FC a script was used which is basically the same as what the developers have proposed in the getting started page, but has been changed in a way of giving parameters to it, such as parameters for giving the custom kernel or having a network.

Building and Running DC:

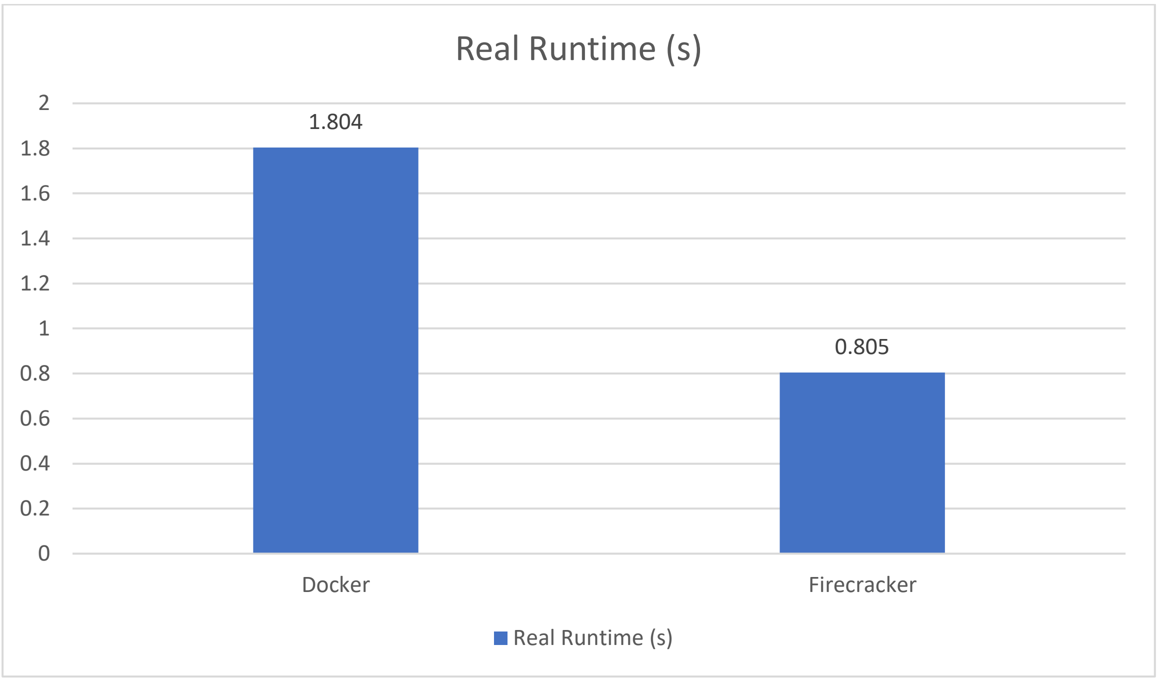
The DC was created from the parent image of ubuntu:22.04. and the necessary libraries were also installed during the creation.

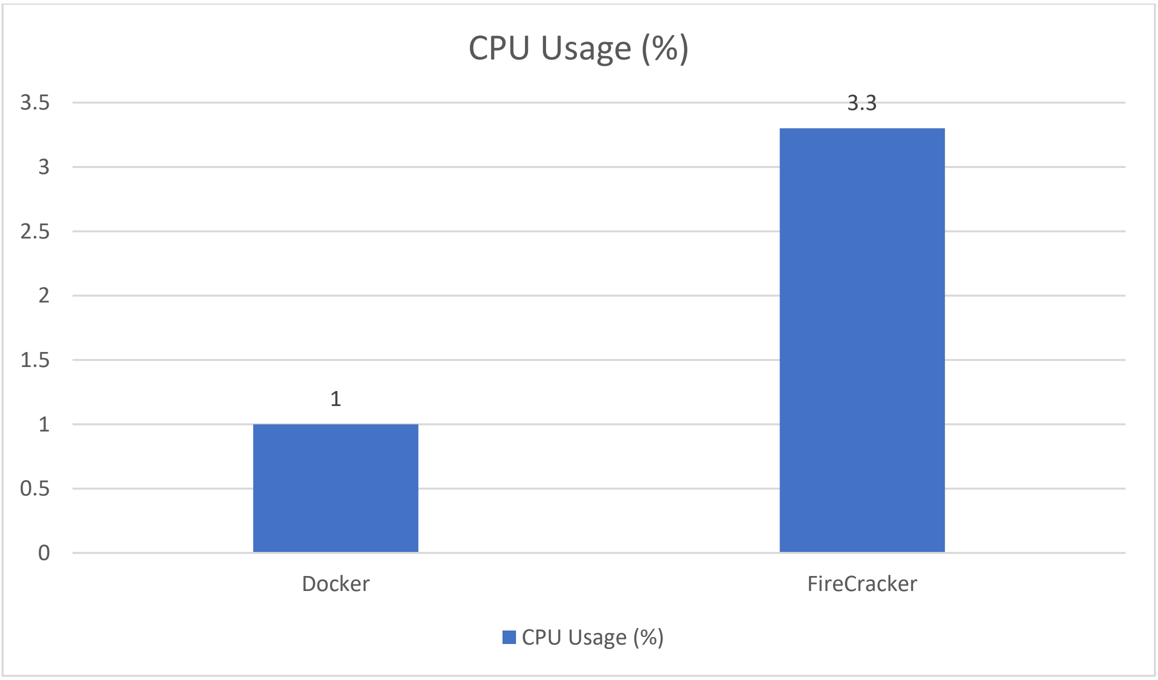
Comparison of FC and DC by Running the Same ML inference Model

In this section the results of benchmarking and profiling the model on both FC and DC will be presented and discussed.

Methodology of benchmarking and profiling the performance on the ML model on FC and DC

In order to benchmark the performance of the ML model, “time” module was used. This is a built-in module in order to assess the running time of the inference model. Another method also was used in order to assess the CPU performance during execution of the inference model by checking the PID of the process “os.getpid()”.





Conclusion

As it can be observed from the charts, Firecracker performs significantly faster than the Docker. Also, the boot time of the Firecracker VM is really fast, of course it depends on the resource configuration as well as the OS that it is running on.

The configuration for both the Docker and the Firecracker was 2 CPUs.