Comparing Docker and Firecracker to deploy Machine Learning models

Mohamed Hashim Changrampadi
December 21, 2022

1 Task

- To deploy a Machine Learning (ML) model in Docker
- To deploy same ML model in Firecracker VM
- Compare the performance of the ML inference model in Docker and Firecracker

2 Machine Learning Model

DeepSpeech, a modified open-source implementation of Baidu's deespeech ASR architecture is deployed in Docker and in Firecracker VM. The Deepspeech processes the given audio file and outputs text transcription of the given audio utterances. The English pre-trained model and the corresponding language model (scorer file) is downloaded into the container to perform the inference of test audio data. The performance of the ASR model is analysed by using the audio files saved in the directory test_audios, available in Github.[1]

3 Docker

In this section, a brief introduction to docker is given. The steps to create a docker image is described, along with the details of machine learning model, contents of dockerfile, and configurations to run a docker container.

3.1 About Docker

Docker is an open-source containerization platform, and it is a way to package application along with its dependencies, so it can run independently on any machine. It can also run multiple instances of an application in isolated containers, allowing developers to quickly spin up new environments and push out updates, or allow easy deployment of containerized (isolated) applications on edge devices.

3.2 Creating a Docker Image

The Docker image contains the necessary packages and information for a container to run. The Docker image is built using the contents of the Dockerfile. The Dockerfile is a simple textfile describing the base image, libraries to install, files to copy and executables to run during creation

[1] https://github.com/cmhashim/docker_firecracker_comparison

of container. Base image, 'python:3.9' is the parent image from which the Docker image will be built on. The library deepspeech, pre-trained model, scorer file, test audio files, and python script to run the inference are installed/copied onto the image.

3.3 Running a Docker container

Once the Docker image is created, you can check the list of images that can be used by Docker to run the containers using the command, *sudo docker images*. The Docker container is created using these docker images. There are two ways to run a container, either using the run or create and start commands. To create and start the container, use the command *sudo docker run ds:1.2*. To only create a container, and not starting it , use *sudo docker create ds:1.2*.

The docker container is started by configuring the resources to be used during running the container. For the task, these resources is set as 2 GB memory and 2 CPU cores, and is done as *sudo docker run -m 2GB –cpus 2.0 -it –entrypoint bash –rm ds:1.2*

4 Firecracker

Firecracker is an open-source virtualization technology developed by Amazon Web Services (AWS) to enable serverless computing. It is designed to run containers and micro virtual machines in secure and cost-effective manner. Firecracker allows users to spin up new environments quickly, while providing strong isolation and security. Launching any application is much faster on Firecracker than on Docker.

4.1 Creating a Firecracker VM

The Firecracker VM is created locally using Weave Ignite, an open source Virtual Machine (VM) manager. Ignite supports running OCI-compliant image (Docker image) as firecracker VM. The docker image is saved and imported to be used as ignite images.

4.2 Running a Firecracker VM

The firecracker vm can be run with specific resources defined during the runtime. Same resource constrained areused while creating the Docker containers *sudo ignite run ds:1.2 –cpus 2 –memory 2GB –name my-vm*. This method actually uses Ignite to boot up the imported docker image as the firecracker VM. Once VM is started, you can get into the terminal, *sudo ignite attach my-vm*.

5 Performance Analysis

The Docker container is created from the Docker image and then started when necessary. The Docker container can be stopped and then started again, maintaining the state of the machine. However the Docker container could take up storage space, if it needs to be started again. The Firecracker VM is created and can be started to run inferences. Even the VM can be stopped and

started again to boot into VM. However running (creating and starting) a Firecracker VM is much faster than running (creating a docker image and starting container) Docker container.

5.1 DeepSpeech Inferences

The execution time for different inferences are recorded for comparison. **Single Instance Inference:** a single call to deepspeech to perform transcription of a single audio file. **Multi Instance Inference:** a multiple call to deepspeech to perform transcriptions of several audio files. **Multiple Single Instance Inference:** a multiple call to deepspeech to perform transcription of same audio file. The table lists the performance of inference in Docker and Firecracker. The same python scripts are ran in Docker and Firecracker for the inference. The script inference.py transcribes the given single audio file, and script inference_all.py transcribes all the audio files inside the test_audios directory. Both the script also takes a number as one argument, based on which the inference in ran those number of times.

No.	Task	Details	Docker	Firecracker
1	Import time	import deepspeech	0.411s	0.142s
2	Single instance inference	time deepspeech ...	11.526s	-
	(8555-292519-0009.wav)	model load time	0.012s	0.186s
	(17.965s)	inference time	8.452s	57.445s
3	Single instance inference-1	python inference.py	101.257s	79.801s
	(1221-135766-0010.wav)	model load time	0.012s	0.015s
	(15.050s)	inference time	32.663s	19.246s
	Single instance inference-2	model load time	0.021s	0.014s
	(1221-135766-0010.wav)	inference time	7.294s	6.412s
	Single instance inference	model load time	avg 0.012s	avg 0.015s
	(3rd-10th time)	inference time	avg 7.158s	avg 6.418s
4	Multiple instance inference	python inference_all.py	256.818s	202.987s
	(12 audios)	model load time	avg 0.012s	avg 0.015s
5	Multiple instance inference	python inference_all.py	438.088s	376.980s
	(12 audios, 3 times)	model load time	avg 0.012s	avg 0.015s

The Figure.1 shows the inference time of the audio file 1221-135766-0010.wav of duration 15.050s, deployed inside Docker container and Firecracker VM. It is evident that the inference time in Firecracker VM is comparatively less than inferred inside Docker container, for all the 10 instances. The inference time during the first instance is higher than the other instances.

In the Figure.2, 12 different audio files of varying duration is transcribed using deepspeech, and its inference time is plotted. For most of the audio files, the inference time inside Firecracker VM is less.

To conclude, the inference of the ML models usually make use of large pre-trained models stored in cloud. To be able to perform inference on edge devices completely without using cloud resources, these pre-trained model has to be stored in local edge devices for low latency inferences. This scenario is imitated and inference was done locally without network connectivity. Due to the resource contraints of the host machine, containers with ML frameworks like torch

Figure 1: Comparison of inference time of a audio file in 10 instances

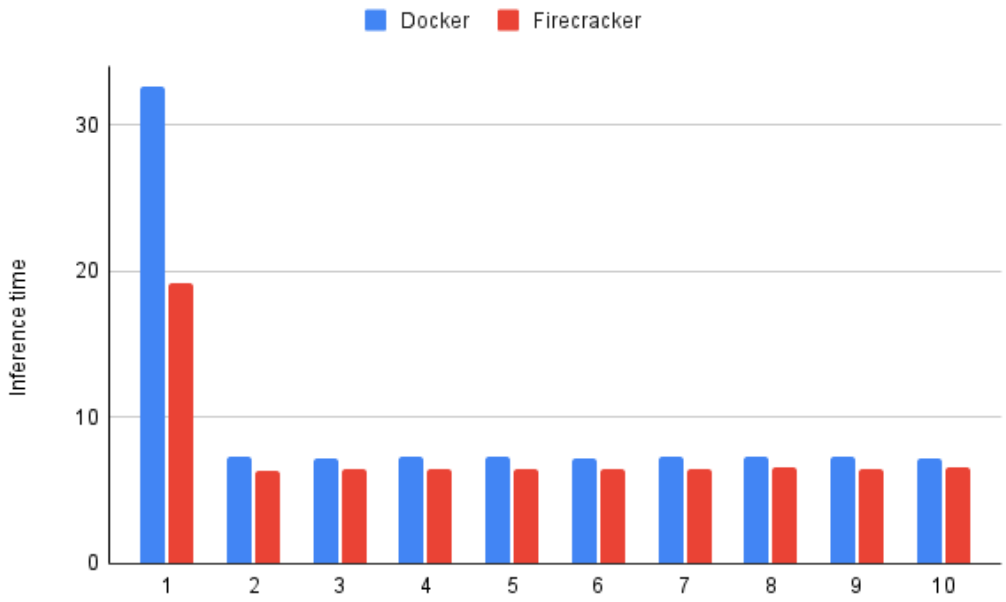
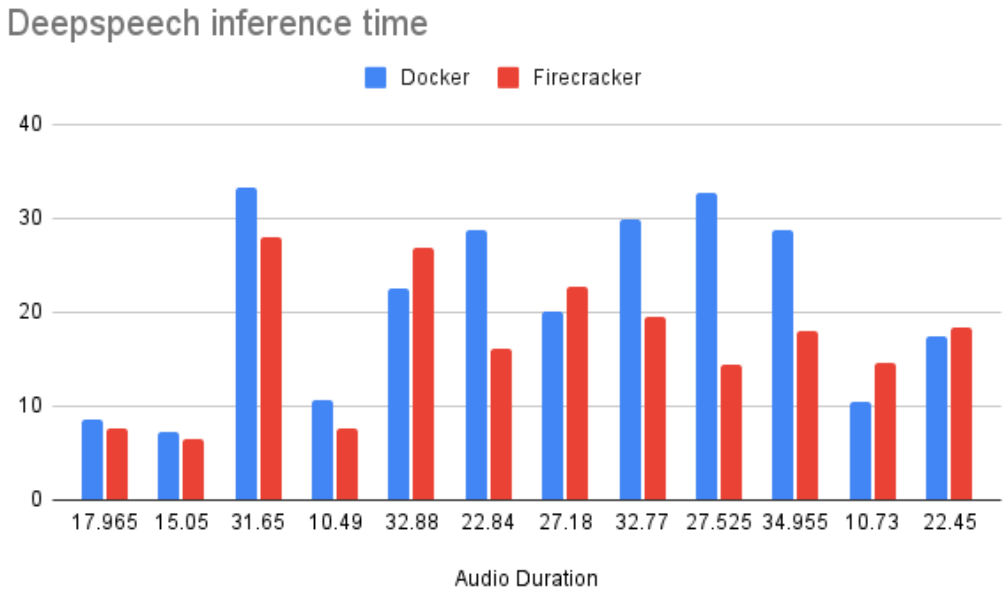


Figure 2: Comparison of inference time of 12 different audio files



were not created. For Deepspeech inference, both Docker and Firecracker VM performs reasonably well, even though Firecracker VM is better than the other. However, deepspeech inferences in Firecracker VM are executed in less time than in the Docker container.

6 Whats Next ?

- Run different ML frameworks to compare performance effectively.
- Deploy ML model with web application in Docker and firecracker VM in AWS EC2 instance.
- Run multiple applications or overload to benchmark the bottleneck of Docker and Firecracker VM.
- Use Prometheus to monitor the resource utilization and visualization using Grafana can be done.
- Deploy a ML model on Nvidia Jetson and compare the performances.