Cloud & ML : Assignment 3

Shiv Ratan Sinha – `srs9969, N16386999`

March 19, 2022

Introduction

In this assignment I am performing MNIST number detection on Docker and Singularity.
I used on my laptop and used Singularity on NYU HPC.
I have also compared both the containerization tools in the end.

1. **Docker**
   I used Docker on my laptop and there were few interesting scenarios with Apple M1. Since sharing GPUs with Docker container isn't possible currently (*as per my research*) the training was a lot slower. Nvidia GPUs can be shared inside the containers using nvidia-toolkit but unfortunately, I couldn't leverage that. Steps followed were:

   - Downloaded the Docker application and allocated proper resources in the resources tab of Docker app (4 CPUs, 10Gb memory).
   - Next was to create a Dockerfile and use our MNIST training code inside the container.
   - The Dockerfile uses pytorch base image (*by default pulls latest*) and has the code to copy our model code inside the container.

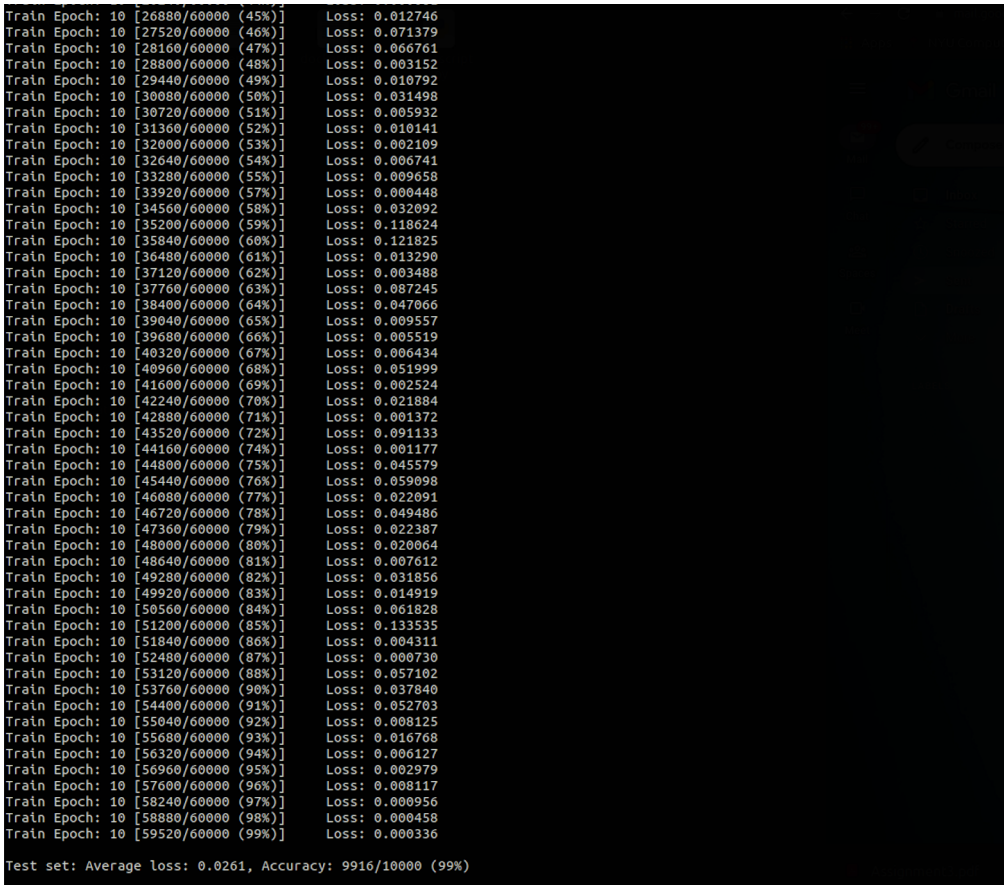
   - I have used the CMD command of Dockerfile which instructs to run the 'given' command as soon as the container is up and also commandline argument (epoch) is passed in the command.

- The an image is built using the *docker build* command and after running it the output is stored in *docker-run.out*.

```
❯ docker build -t mnistdocker .
[+] Building 0.3s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 36B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/pytorch/pytorch:latest
 => [internal] load build context
 => => transferring context: 112B
 => [1/3] FROM docker.io/pytorch/pytorch@sha256:9904a7e081eaca29e3ee46afac87f2879676dd3bf7b5e9b8450454d84e074ef0
 => CACHED [2/3] COPY . /app
 => CACHED [3/3] WORKDIR /app
 => exporting to image
 => => exporting layers
 => => writing image sha256:afb1fa927aa6c24547c20a26e227e948816c036106b214aae24fff296d903e16
 => => naming to docker.io/library/mnistdocker

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

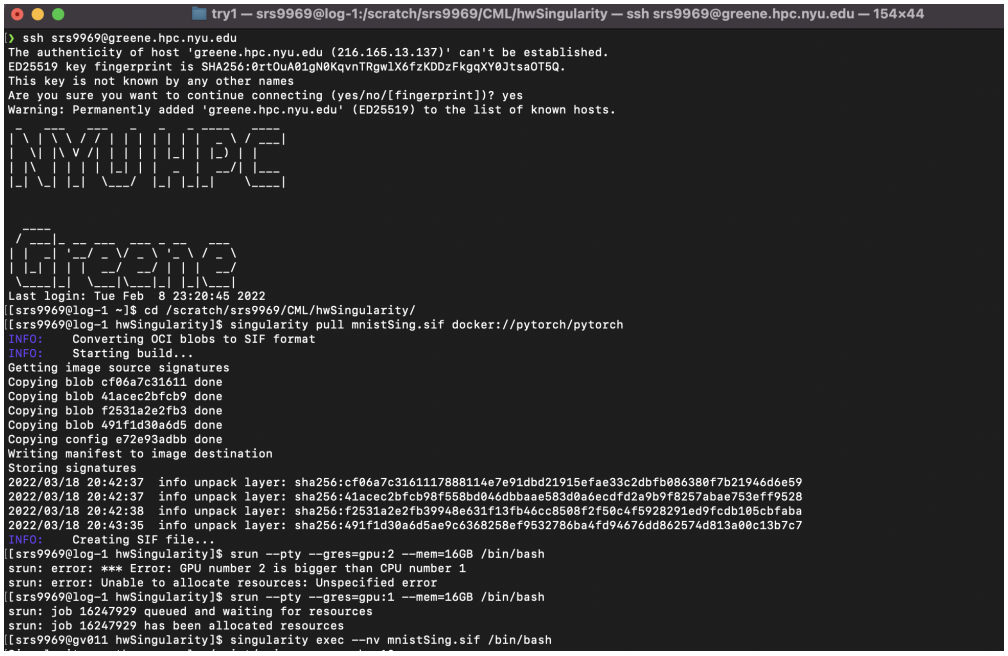
```
    return F.conv2d(input, weight, bias, self.stride,
KeyboardInterrupt
~/De/NYU/sem4_spring/CloudML/hws/hw4-docker/try1 ❯ docker run --platform linux/amd64 mnistdocker 2>&1 | tee docker-run.out
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ../data/MNIST/raw/train-images-idx3-ubyte.gz
9913344it [00:02, 3536689.82it/s]                             Extracting ../data/MNIST/raw/train-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ../data/MNIST/raw/train-labels-idx1-ubyte.gz

29696it [00:00, 3915808.97it/s]
  0%|          | 0/1648877 [00:00<?, ?it/s]Extracting ../data/MNIST/raw/train-labels-idx1-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw/t10k-images-idx3-ubyte.gz
1649664it [00:00, 1741874.06it/s]                             Extracting ../data/MNIST/raw/t10k-images-idx3-ubyte.gz to ../data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz

5120it [00:00, 17544801.05it/s]        Extracting ../data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ../data/MNIST/raw

Train Epoch: 1 [0/60000 (0%)]   Loss: 2.305400
Train Epoch: 1 [640/60000 (1%)] Loss: 1.358980
Train Epoch: 1 [1280/60000 (2%)]        Loss: 0.862895
Train Epoch: 1 [1920/60000 (3%)]        Loss: 0.588242
Train Epoch: 1 [2560/60000 (4%)]        Loss: 0.344540
Train Epoch: 1 [3200/60000 (5%)]        Loss: 0.436769
Train Epoch: 1 [3840/60000 (6%)]        Loss: 0.260234
Train Epoch: 1 [4480/60000 (7%)]        Loss: 0.291517
Train Epoch: 1 [5120/60000 (9%)]        Loss: 0.548768
Train Epoch: 1 [5760/60000 (10%)]       Loss: 0.229064
Train Epoch: 1 [6400/60000 (11%)]       Loss: 0.256919
Train Epoch: 1 [7040/60000 (12%)]       Loss: 0.325944
Train Epoch: 1 [7680/60000 (13%)]       Loss: 0.182787
Train Epoch: 1 [8320/60000 (14%)]       Loss: 0.222929
Train Epoch: 1 [8960/60000 (15%)]       Loss: 0.250860
Train Epoch: 1 [9600/60000 (16%)]       Loss: 0.115535
Train Epoch: 1 [10240/60000 (17%)]      Loss: 0.291817
Train Epoch: 1 [10880/60000 (18%)]      Loss: 0.104538
Train Epoch: 1 [11520/60000 (19%)]      Loss: 0.423641
Train Epoch: 1 [12160/60000 (20%)]      Loss: 0.265125
Train Epoch: 1 [12800/60000 (21%)]      Loss: 0.228689
Train Epoch: 1 [13440/60000 (22%)]      Loss: 0.212150
Train Epoch: 1 [14080/60000 (23%)]      Loss: 0.144838
Train Epoch: 1 [14720/60000 (25%)]      Loss: 0.395401
Train Epoch: 1 [15360/60000 (26%)]      Loss: 0.141356
Train Epoch: 1 [16000/60000 (27%)]      Loss: 0.136433
Train Epoch: 1 [16640/60000 (28%)]      Loss: 0.202155
Train Epoch: 1 [17280/60000 (29%)]      Loss: 0.068605
Train Epoch: 1 [17920/60000 (30%)]      Loss: 0.190426
Train Epoch: 1 [18560/60000 (31%)]      Loss: 0.190940
Train Epoch: 1 [19200/60000 (32%)]      Loss: 0.267266
Train Epoch: 1 [19840/60000 (33%)]      Loss: 0.074468
Train Epoch: 1 [20480/60000 (34%)]      Loss: 0.051595
Train Epoch: 1 [21120/60000 (35%)]      Loss: 0.234107
Train Epoch: 1 [21760/60000 (36%)]      Loss: 0.007340
Train Epoch: 1 [22400/60000 (37%)]      Loss: 0.108690
```

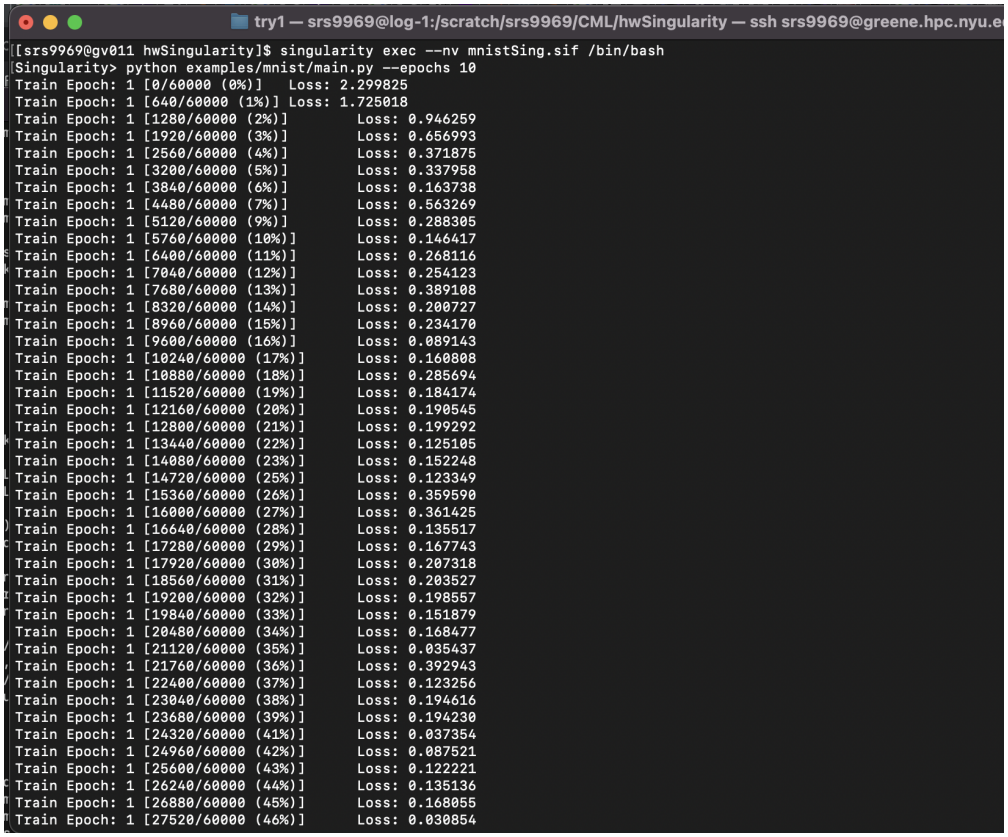


2. **Singularity**
   I used Singularity on NYU HPC. Open-source software, Singularity, builds and runs containers on clusters of high-performance computers. It allows specific applications to run on virtualized environments. It is an industry standard tool to utilize containers in HPC environments.
   Steps followed were:

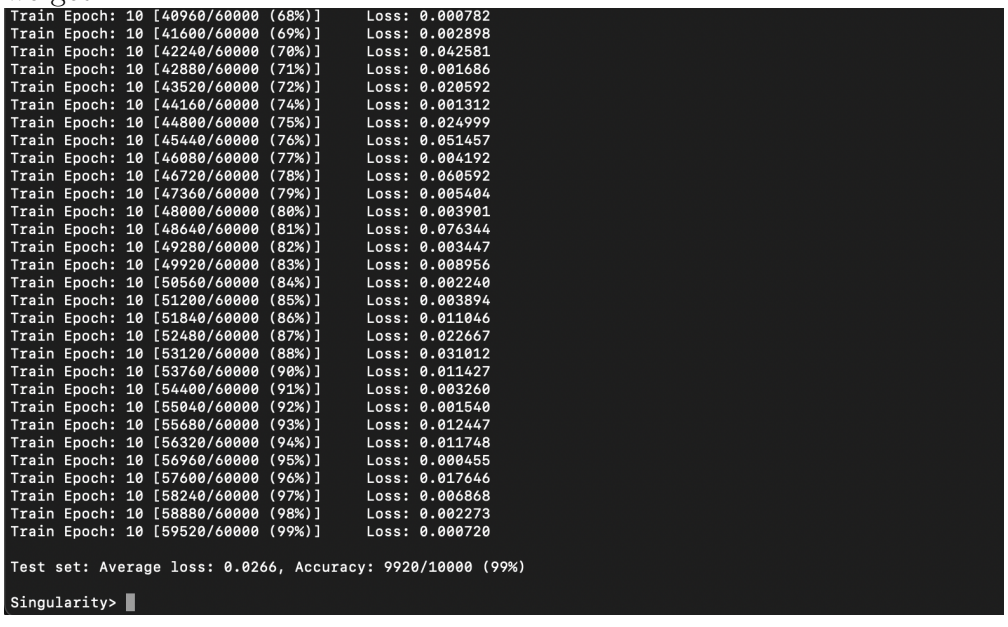
   - Firstly I accessed NYU High performance computing Greene cluster and created required directories in the /scratch.



- Above I created a base immutable image in the Singularity Image File (SIF) format. This format helps ensures reproducible and verifiable images. I used pytorch image from Docker registry as the base image. Then I ran a parallel job on the NYU HPC cluster (managed by Slurm) using the srun command where I also gave the resources I wanted to use (for eg gpu : 1)



- In above and below image you can see the model being trained and the accuracy we get.



3. **Observation**

   - Running on laptop using Docker was slower than using on HPC Singularity - obviosuly as couldn't leverage the GPUs and secondly I used Slurm on HPC.

4. **Singularity v/s Docker**

   - Security:
     Docker images are not secure because they provide a means to gain root access to the system they are running on. However, Singularity is a secure alternative to Docker.
   - Orchestration:
     Docker provides Docker swarm and Kubernetes to manage docker containers whereas Singularity does not support orchestration yet. It runs as a job and works well wil Slurm.
   - Namespace Sharing:
     Docker, by default shares as little as possible. The network space, user space are all isolated by default but can be made to share via commands. Singularity, on the other hand enables most of it by default (*can be controlled*).
   - Volume sharing:
     By default host volumes are not shared inside the container but can be altered. Whereas, in Singularity, it runs as a user process and has access to all of hosts' filesystem and devices attached. Singularity easily makes use of GPUs, high speed networks, parallel filesystems on a cluster or server by default.
   - Portability:
     Both are equally good, as in one can use sif format images from Singularity or use Docker hub - colletion of iamges to upload/download various images.
   - Parallel programming:
     There's nothing like MPI being supported in Docker containers, whereas Singularity supports MPI.

   Conclusion: Both are useful in their own way, for eg: microservices architecture v/s batch-processing. Just that using Singularity provides more security than Docker which is mainly used for isolation.

References

1. NYU singularity
2. nextplatform
3. Docker official doc

4. Docker wikipedia
5. Singularity wiki
6. Singularity official site
7. pytorch mnist code