

Cloud & ML : Assignment 3

Shiv Ratan Sinha – srs9969, N16386999

March 19, 2022

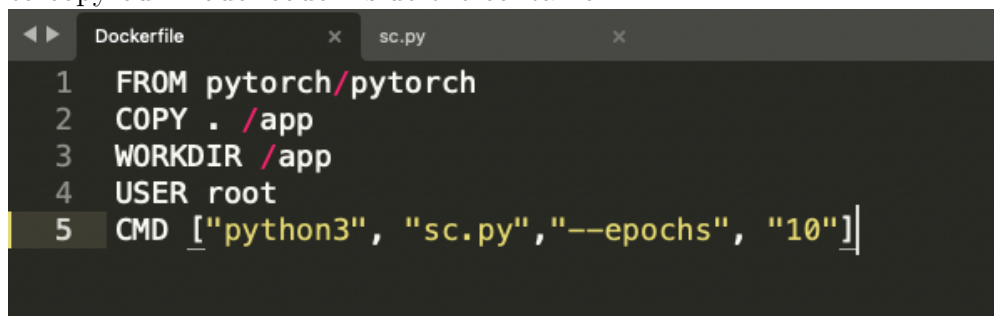
Introduction

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.



```
1 FROM pytorch/pytorch
2 COPY . /app
3 WORKDIR /app
4 USER root
5 CMD ["python3", "sc.py", "--epochs", "10"]
```

- 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: 368
=> [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, 3915800.97it/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, 17544001.05it/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 (8%)] 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
```

```

Train Epoch: 10 [26880/60000 (45%)] Loss: 0.012746
Train Epoch: 10 [27520/60000 (46%)] Loss: 0.071379
Train Epoch: 10 [28160/60000 (47%)] Loss: 0.066761
Train Epoch: 10 [28800/60000 (48%)] Loss: 0.003152
Train Epoch: 10 [29440/60000 (49%)] Loss: 0.010792
Train Epoch: 10 [30080/60000 (50%)] Loss: 0.031498
Train Epoch: 10 [30720/60000 (51%)] Loss: 0.005932
Train Epoch: 10 [31360/60000 (52%)] Loss: 0.010141
Train Epoch: 10 [32000/60000 (53%)] Loss: 0.002109
Train Epoch: 10 [32640/60000 (54%)] Loss: 0.006741
Train Epoch: 10 [33280/60000 (55%)] Loss: 0.009658
Train Epoch: 10 [33920/60000 (57%)] Loss: 0.000448
Train Epoch: 10 [34560/60000 (58%)] Loss: 0.032092
Train Epoch: 10 [35200/60000 (59%)] Loss: 0.118624
Train Epoch: 10 [35840/60000 (60%)] Loss: 0.121825
Train Epoch: 10 [36480/60000 (61%)] Loss: 0.013290
Train Epoch: 10 [37120/60000 (62%)] Loss: 0.003488
Train Epoch: 10 [37760/60000 (63%)] Loss: 0.087245
Train Epoch: 10 [38400/60000 (64%)] Loss: 0.047066
Train Epoch: 10 [39040/60000 (65%)] Loss: 0.009557
Train Epoch: 10 [39680/60000 (66%)] Loss: 0.005519
Train Epoch: 10 [40320/60000 (67%)] Loss: 0.006434
Train Epoch: 10 [40960/60000 (68%)] Loss: 0.051999
Train Epoch: 10 [41600/60000 (69%)] Loss: 0.002524
Train Epoch: 10 [42240/60000 (70%)] Loss: 0.021884
Train Epoch: 10 [42880/60000 (71%)] Loss: 0.001372
Train Epoch: 10 [43520/60000 (72%)] Loss: 0.091133
Train Epoch: 10 [44160/60000 (74%)] Loss: 0.001177
Train Epoch: 10 [44800/60000 (75%)] Loss: 0.045579
Train Epoch: 10 [45440/60000 (76%)] Loss: 0.059098
Train Epoch: 10 [46080/60000 (77%)] Loss: 0.022091
Train Epoch: 10 [46720/60000 (78%)] Loss: 0.049486
Train Epoch: 10 [47360/60000 (79%)] Loss: 0.022387
Train Epoch: 10 [48000/60000 (80%)] Loss: 0.020064
Train Epoch: 10 [48640/60000 (81%)] Loss: 0.007612
Train Epoch: 10 [49280/60000 (82%)] Loss: 0.031856
Train Epoch: 10 [49920/60000 (83%)] Loss: 0.014919
Train Epoch: 10 [50560/60000 (84%)] Loss: 0.061828
Train Epoch: 10 [51200/60000 (85%)] Loss: 0.133535
Train Epoch: 10 [51840/60000 (86%)] Loss: 0.004311
Train Epoch: 10 [52480/60000 (87%)] Loss: 0.000730
Train Epoch: 10 [53120/60000 (88%)] Loss: 0.057102
Train Epoch: 10 [53760/60000 (90%)] Loss: 0.037840
Train Epoch: 10 [54400/60000 (91%)] Loss: 0.052703
Train Epoch: 10 [55040/60000 (92%)] Loss: 0.008125
Train Epoch: 10 [55680/60000 (93%)] Loss: 0.016768
Train Epoch: 10 [56320/60000 (94%)] Loss: 0.006127
Train Epoch: 10 [56960/60000 (95%)] Loss: 0.002979
Train Epoch: 10 [57600/60000 (96%)] Loss: 0.008117
Train Epoch: 10 [58240/60000 (97%)] Loss: 0.000956
Train Epoch: 10 [58880/60000 (98%)] Loss: 0.000458
Train Epoch: 10 [59520/60000 (99%)] Loss: 0.000336

Test set: Average loss: 0.0261, Accuracy: 9916/10000 (99%)

```

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.

```
try1 — srs9969@log-1:/scratch/srs9969/CML/hwSingularity — ssh srs9969@greene.hpc.nyu.edu — 154x44
> ssh srs9969@greene.hpc.nyu.edu
The authenticity of host 'greene.hpc.nyu.edu (216.165.13.137)' can't be established.
ED25519 key fingerprint is SHA256:0rt0UA0igN0KqvnTRgwIX6fzKDD2FkgqXY0jtsaOT5Q.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'greene.hpc.nyu.edu' (ED25519) to the list of known hosts.

NYU
CML

Last login: Tue Feb 8 23:28:45 2022
[srs9969@log-1 ~]$ cd /scratch/srs9969/CML/hwSingularity/
[srs9969@log-1 hwSingularity]$ singularity pull mnistSing.sif docker://pytorch/pytorch
INFO: Converting OCI blobs to SIF format
INFO: Starting build...
Getting image source signatures
Copying blob cf06a7c31611 done
Copying blob 41acec2bfc9 done
Copying blob f2531a2e2fb3 done
Copying blob 491fd30a6d5 done
Copying config e72e93adbb done
Writing manifest to image destination
Storing signatures
2022/03/18 20:42:37 info unpack layer: sha256:cf06a7c3161117888114e7e91dbd21915efae33c2dbfb086380f7b21946d6e59
2022/03/18 20:42:37 info unpack layer: sha256:41acec2bfc98f558bd046dbbaae583d0a6ecd9d2a9b9f8257abae753eff9528
2022/03/18 20:42:38 info unpack layer: sha256:f2531a2e2fb39948e631f13fb46cc8508f2f50c4f5928291ed9fcd9105cbfaba
2022/03/18 20:43:35 info unpack layer: sha256:491fd30a6d5ae9c6368258ef9532786ba4fd94676dd862574d813a00c13b7c7
INFO: Creating SIF file...
[srs9969@log-1 hwSingularity]$ srun --pty --gres=gpu:2 --mem=16GB /bin/bash
srun: error: *** Error: GPU number 2 is bigger than CPU number 1
srun: error: Unable to allocate resources: Unspecified error
[srs9969@log-1 hwSingularity]$ srun --pty --gres=gpu:1 --mem=16GB /bin/bash
srun: job 16247929 queued and waiting for resources
srun: job 16247929 has been allocated resources
[srs9969@gv011 hwSingularity]$ singularity exec --nv mnistSing.sif /bin/bash
```

- 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)

```
try1 — srs9969@log-1:/scratch/srs9969/CML/hwSingularity — ssh srs9969@greene.hpc.nyu.e
[srs9969@q011 hwSingularity]$ singularity exec --nv mnistSing.sif /bin/bash
Singularity> python examples/mnist/main.py --epochs 10
Train Epoch: 1 [0/60000 (0%)] Loss: 2.299825
Train Epoch: 1 [640/60000 (1%)] Loss: 1.725018
Train Epoch: 1 [1280/60000 (2%)] Loss: 0.946259
Train Epoch: 1 [1920/60000 (3%)] Loss: 0.656993
Train Epoch: 1 [2560/60000 (4%)] Loss: 0.371875
Train Epoch: 1 [3200/60000 (5%)] Loss: 0.337958
Train Epoch: 1 [3840/60000 (6%)] Loss: 0.163738
Train Epoch: 1 [4480/60000 (7%)] Loss: 0.563269
Train Epoch: 1 [5120/60000 (9%)] Loss: 0.288305
Train Epoch: 1 [5760/60000 (10%)] Loss: 0.146417
Train Epoch: 1 [6400/60000 (11%)] Loss: 0.268116
Train Epoch: 1 [7040/60000 (12%)] Loss: 0.254123
Train Epoch: 1 [7680/60000 (13%)] Loss: 0.389108
Train Epoch: 1 [8320/60000 (14%)] Loss: 0.200727
Train Epoch: 1 [8960/60000 (15%)] Loss: 0.234170
Train Epoch: 1 [9600/60000 (16%)] Loss: 0.089143
Train Epoch: 1 [10240/60000 (17%)] Loss: 0.160808
Train Epoch: 1 [10880/60000 (18%)] Loss: 0.285694
Train Epoch: 1 [11520/60000 (19%)] Loss: 0.184174
Train Epoch: 1 [12160/60000 (20%)] Loss: 0.190545
Train Epoch: 1 [12800/60000 (21%)] Loss: 0.199292
Train Epoch: 1 [13440/60000 (22%)] Loss: 0.125105
Train Epoch: 1 [14080/60000 (23%)] Loss: 0.152248
Train Epoch: 1 [14720/60000 (25%)] Loss: 0.123349
Train Epoch: 1 [15360/60000 (26%)] Loss: 0.359590
Train Epoch: 1 [16000/60000 (27%)] Loss: 0.361425
Train Epoch: 1 [16640/60000 (28%)] Loss: 0.135517
Train Epoch: 1 [17280/60000 (29%)] Loss: 0.167743
Train Epoch: 1 [17920/60000 (30%)] Loss: 0.207318
Train Epoch: 1 [18560/60000 (31%)] Loss: 0.203527
Train Epoch: 1 [19200/60000 (32%)] Loss: 0.198557
Train Epoch: 1 [19840/60000 (33%)] Loss: 0.151879
Train Epoch: 1 [20480/60000 (34%)] Loss: 0.168477
Train Epoch: 1 [21120/60000 (35%)] Loss: 0.035437
Train Epoch: 1 [21760/60000 (36%)] Loss: 0.392943
Train Epoch: 1 [22400/60000 (37%)] Loss: 0.123256
Train Epoch: 1 [23040/60000 (38%)] Loss: 0.194616
Train Epoch: 1 [23680/60000 (39%)] Loss: 0.194230
Train Epoch: 1 [24320/60000 (41%)] Loss: 0.037354
Train Epoch: 1 [24960/60000 (42%)] Loss: 0.087521
Train Epoch: 1 [25600/60000 (43%)] Loss: 0.122221
Train Epoch: 1 [26240/60000 (44%)] Loss: 0.135136
Train Epoch: 1 [26880/60000 (45%)] Loss: 0.168055
Train Epoch: 1 [27520/60000 (46%)] Loss: 0.030854
```

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

```
Train Epoch: 10 [40960/60000 (68%)] Loss: 0.000782
Train Epoch: 10 [41600/60000 (69%)] Loss: 0.002898
Train Epoch: 10 [42240/60000 (70%)] Loss: 0.042581
Train Epoch: 10 [42880/60000 (71%)] Loss: 0.001686
Train Epoch: 10 [43520/60000 (72%)] Loss: 0.020592
Train Epoch: 10 [44160/60000 (74%)] Loss: 0.001312
Train Epoch: 10 [44800/60000 (75%)] Loss: 0.024999
Train Epoch: 10 [45440/60000 (76%)] Loss: 0.051457
Train Epoch: 10 [46080/60000 (77%)] Loss: 0.004192
Train Epoch: 10 [46720/60000 (78%)] Loss: 0.060592
Train Epoch: 10 [47360/60000 (79%)] Loss: 0.005404
Train Epoch: 10 [48000/60000 (80%)] Loss: 0.003901
Train Epoch: 10 [48640/60000 (81%)] Loss: 0.076344
Train Epoch: 10 [49280/60000 (82%)] Loss: 0.003447
Train Epoch: 10 [49920/60000 (83%)] Loss: 0.008956
Train Epoch: 10 [50560/60000 (84%)] Loss: 0.002240
Train Epoch: 10 [51200/60000 (85%)] Loss: 0.003894
Train Epoch: 10 [51840/60000 (86%)] Loss: 0.011046
Train Epoch: 10 [52480/60000 (87%)] Loss: 0.022667
Train Epoch: 10 [53120/60000 (88%)] Loss: 0.031012
Train Epoch: 10 [53760/60000 (90%)] Loss: 0.011427
Train Epoch: 10 [54400/60000 (91%)] Loss: 0.003260
Train Epoch: 10 [55040/60000 (92%)] Loss: 0.001540
Train Epoch: 10 [55680/60000 (93%)] Loss: 0.012447
Train Epoch: 10 [56320/60000 (94%)] Loss: 0.011748
Train Epoch: 10 [56960/60000 (95%)] Loss: 0.000455
Train Epoch: 10 [57600/60000 (96%)] Loss: 0.017646
Train Epoch: 10 [58240/60000 (97%)] Loss: 0.006868
Train Epoch: 10 [58880/60000 (98%)] Loss: 0.002273
Train Epoch: 10 [59520/60000 (99%)] Loss: 0.000720

Test set: Average loss: 0.0266, Accuracy: 9920/10000 (99%)

Singularity> █
```

3. Observation

- Running on laptop using Docker was slower than using on HPC Singularity - obviously 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 with 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 - collection of images 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](#)