Create ec2 instances:
To be able to use the system I first needed to launch some ec2 instances

```
aws ec2 run-instances --image-id ami-0ff8a91507f77f867 --count 1 --instance-type m5.large --key-name p0KeyPair
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-0ff8a91507f77f867",
            "InstanceId": "i-007cafb754913a075",
            "InstanceType": "m5.large",
            "KeyName": "p0KeyPair",
            "LaunchTime": "2023-05-15T00:06:15+00:00",
            "Monitoring": {
                "State": "disabled"
            },
            "Placement": {
                "AvailabilityZone": "us-east-1d",
                "GroupName": "",
                "Tenancy": "default"
            },
            "PrivateDnsName": "ip-172-31-0-129.ec2.internal",
            "PrivateIpAddress": "172.31.0.129",
            "ProductCodes": [],
            "PublicDnsName": "",
            "State": {
```

Ssh into the instance:

```
ssh -i downloads/p0KeyPair.pem ec2-user@3.221.159.2
The authenticity of host '3.221.159.2 (3.221.159.2)' can't be established.
ED25519 key fingerprint is SHA256:i1HEsFi1zF7dmD3UbQbfrYfUQyqrcZ0/UekvSoRNrO0.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.221.159.2' (ED25519) to the list of known hosts.
       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|
https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
38 package(s) needed for security, out of 61 available
Run "sudo yum update" to apply all
```

Install TensorFlow:

```
pip install tensorflow
Collecting tensorflow
  Using cached
https://files.pythonhosted.org/packages/d3/59/d88fe8c58ffb66aca21d03c0e290cd68327cc133591130c674985e98a482/tens
orflow-1.14.0-cp27-cp27mu-manylinux1_x86_64.whl
Collecting six>=1.10.0 (from tensorflow)
  Using cached
https://files.pythonhosted.org/packages/d9/5a/e7c31adbe875f2abbb91bd84cf2dc52d792b5a01506781dbcf25c91daf11/six-
1.16.0-py2.py3-none-any.whl
Collecting keras-applications>=1.0.6 (from tensorflow)
  Using cached
https://files.pythonhosted.org/packages/21/56/4bcec5a8d9503a87e58e814c4e32ac2b32c37c685672c30bc8c54c6e478a/Ker
as_Applications-1.0.8.tar.gz
Collecting grpcio>=1.8.6 (from tensorflow)
```

```
  Using cached
https://files.pythonhosted.org/packages/bd/81/6c704c002a992b9d6466c739e3e7687e0bb2365d8cd63d7fc8e95d502cb6/grpcio-1.41.1.tar.gz
    Complete output from command python setup.py egg_info:
    Traceback (most recent call last):
      File "<string>", line 1, in <module>
      File "/tmp/pip-build-K5XF_V/grpcio/setup.py", line 256, in <module>
        if check_linker_need_libatomic():
      File "/tmp/pip-build-K5XF_V/grpcio/setup.py", line 206, in check_linker_need_libatomic
        stderr=PIPE)
      File "/usr/lib64/python2.7/subprocess.py", line 394, in __init__
        errread, errwrite)
      File "/usr/lib64/python2.7
```

Code for cnn training on single machine:

```
import tensorflow as tf
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train /= 255.0
x_test /= 255.0

model = tf.keras.models.Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(optimizer='sgd', loss='categorical_crossentropy',
metrics=['accuracy'])
history = model.fit(x_train.reshape(-1, 28, 28, 1), y_train, epochs=10,
batch_size=32, validation_split=0.1)
```

Code for distributed cnn training:

```
import tensorflow as tf
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

mnist = tf.keras.datasets.mnist
strategy = tf.distribute.MirroredStrategy()
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train /= 255.0
x_test /= 255.0

with strategy.scope():
    model = tf.keras.models.Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
        MaxPooling2D((2, 2)),
```

```
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(64, activation='relu'),
        Dense(10, activation='softmax')
    ])

    model.compile(optimizer='sgd', loss='categorical_crossentropy',
metrics=['accuracy'])
    strategy = tf.distribute.MirroredStrategy()

history = model.fit(train_data, epochs=num_epochs)
```

Monitoring and testing:

The models perform similarly with slight advantage to the distributed system. Since the number of layers is not as large the accuracy of each model is fairly similar at 90-98% on average. On the ec2 console the memory and CPU usage of the distributed model were divided between instances which made the training need less computational power from the system, unlike the first model.

References

Convolutional Neural Network (CNN): https://www.tensorflow.org/tutorials/images/cnn

AWS Documentation: https://aws.amazon.com/documentation/

TensorFlow: https://www.tensorflow.org/

TensorFlow tf.keras: https://www.tensorflow.org/api_docs/python/tf/keras

PyTorch: https://pytorch.org/

EC2 CLI: https://docs.aws.amazon.com/cli/latest/reference/ec2/

SCP (Secure Copy) Documentation: https://linux.die.net/man/1/scp