TonY: An Orchestrator for Distributed Machine Learning Jobs

Jonathan Hung
Agenda

- Background: TensorFlow and YARN
- What is TonY?
- Why use TonY for distributed training?
- Next steps
Productionizing machine learning requires many steps.

The focus of this talk will be model training.
Background

What is TensorFlow?

```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist

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

model = tf.keras.models.Sequential(
    [tf.keras.layers.Flatten(input_shape=(28, 28)),
     tf.keras.layers.Dense(512, activation=tf.nn.relu),
     tf.keras.layers.Dropout(0.2),
     tf.keras.layers.Dense(10, activation=tf.nn.softmax)
    ]
)
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```
What is TensorFlow?

```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist

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

model = tf.keras.models.Sequential([tf.keras.layers.Flatten(input_shape=(28, 28)),
                                   tf.keras.layers.Dense(512, activation=tf.nn.relu),
                                   tf.keras.layers.Dropout(0.2),
                                   tf.keras.layers.Dense(10, activation=tf.nn.softmax)])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Visualisation with TensorBoard
https://learningtensorflow.com/Visualisation/
Background

What is distributed TensorFlow?
How to run distributed TensorFlow?

- Distribute code/data artifacts across multiple machines in distributed job
- Allow tasks in the same distributed job to talk to each other (e.g. tell each worker where all other worker/parameter servers are)
- Ensure your task compute requirements are met before starting distributed job
- Or, have a framework do all of the above for you (Hadoop!)
Background

What is Hadoop?
Background

What is Hadoop?

Distributed File System
Background

What is Hadoop?
Background

How to work with YARN?
Background

How to work with YARN?
Background

How to work with YARN?
What is TonY?
What is TonY?

- Orchestrates running distributed TensorFlow on Hadoop
- Acquires compute resources from Hadoop (memory, CPU, GPU)
- Sets up and launches distributed TensorFlow jobs on Hadoop clusters
- Manages application lifecycle
  - Fault tolerance
  - Job monitoring
TonY Architecture
TonY Architecture

- Entry point for TonY jobs
- Package user’s configurations, user’s model code and submit as YARN application
TonY Architecture

- Job setup and lifecycle management
- Negotiates compute resources from Hadoop
- Sets up container environment
- Launches and monitors containers
TonY Architecture

- Container = Task Executor
- Launches user’s provided python script
- Heartbeats to Application Master for liveness
Why use TonY for distributed training?
Scaling distributed TensorFlow on Hadoop

- Leverage YARN’s fine-grained resource management and multi-tenancy
  - Logical resource isolation via queues
  - Hardware-based physical resource partitioning (CPU, K80, V100)
  - User-based resource limits
Scaling distributed TensorFlow on Hadoop

- Native GPU resource awareness
- Ensures GPU resource isolation and scheduling
Scaling distributed TensorFlow on Hadoop

- One-click TensorBoard access for monitoring training progress
Scaling distributed TensorFlow on Hadoop

- Fault tolerance
- More workers = more failures
Scaling distributed TensorFlow on Hadoop

- Fault tolerance
- More workers = more failures
- First attempt periodically saves model checkpoints to HDFS
Scaling distributed TensorFlow on Hadoop

- Fault tolerance
- More workers = more failures
- First attempt periodically saves model checkpoints to HDFS
- Worker failure -> tear down and restart application
Scaling distributed TensorFlow on Hadoop

- Fault tolerance
- More workers = more failures
- First attempt periodically saves model checkpoints to HDFS
- Worker failure -> tear down and restart application
- Read checkpoints from HDFS, resume from where previous attempt left off
Open Sourced!

- https://github.com/linkedin/TonY

Contributions Welcome!
Next steps

- Dr. Elephant integration
- TonY portal for notebook, job history, cross-execution monitoring
Q & A