Machine Learning over Real-Time Streaming Data with TensorFlow

Yong Tang, MobileIron
SIG IO Lead, TensorFlow
GitHub: yongtang
Streaming Data

- IoT sensors
- GPS positions
- Web transactions
- Social media updates
Event Driven Microservices
Distributed Streaming Framework

Structured Commit Log

Partition and Offset Management
Partition 0
0 1 2 3 4 5 6 7 8 9 1 1 1 2

Partition 1
0 1 2 3 4 5 6 7 8 9

Partition 2
0 1 2 3 4 5 6 7 8 9 1 1 1 2

Old
New

Writes
Server 1
- P0
- P3

Server 2
- P1
- P2

Consumer Group A
- C1
- C2

Consumer Group B
- C3
- C4
- C5
- C6
ML as Microservices
Infrastructure Solution

Convert to TFRecord File

(Distributed) File System

Service

(Distributed) File System

Write Back to Kafka/etc.
Solution

Distributed Persistent Storage

+ Distributed ML Data Pipeline
Distributed Persistent Storage

Server 1
- P0
- P3

Server 2
- P1
- P2

Consumer Group A
- C1
- C2

Consumer Group B
- C3
- C4
- C5
- C6
TensorFlow

Distributed ML Data Pipeline
Kafka Support in TensorFlow

<table>
<thead>
<tr>
<th>Part of tf.contrib in TF 1.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part of tensorflow-io for TF 2.0:</td>
</tr>
<tr>
<td>KafkaDataset (for Read)</td>
</tr>
<tr>
<td>KafkaOutputSequence (for Write)</td>
</tr>
</tbody>
</table>
Kafka Support in TensorFlow

Package: tensorflow-io 0.9.0
Implemented in C++
Dependency: tensorflow only
Python and R bindings
Kafka Support in TensorFlow 2.0

$ pip install tensorflow-io
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(128, activation='relu'),
                                     tf.keras.layers.Dropout(0.2),
                                     tf.keras.layers.Dense(10, activation='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)
import tensorflow as tf

import tensorflow_io.kafka as kafka_io

def func_x(self, x):
    # Decode image to (28, 28)
    x = tf.io.decode_raw(x, out_type=tf.uint8)
    x = tf.reshape(x, [28, 28])
    # Convert to float32 for tf.keras
    x = tf.image.convert_image_dtype(x, tf.float32)
    return x

def func_y(self, y):
    # Decode label to ()
    x = tf.io.decode_raw(x, out_type=tf.uint8)
    x = tf.reshape(x, [])
    return x

d_train_x = kafka_io.KafkaDataset(['x_train:0'], group='x', eof=True).map(func_x)

d_train_y = kafka_io.KafkaDataset(['y_train:0'], group='y', eof=True).map(func_y)
model = tf.keras.models.Sequential([  
    tf.keras.layers.Flatten(input_shape=(28, 28)),  
    tf.keras.layers.Dense(128, activation='relu'),  
    tf.keras.layers.Dropout(0.2),  
    tf.keras.layers.Dense(10, activation='softmax')  
])

model.compile(optimizer='adam',  
    loss='sparse_categorical_crossentropy',  
    metrics=['accuracy'])

d_train = tf.data.Dataset.zip(  
    (d_train_x, d_train_y)).batch(1)

model.fit(d_train, epochs=5)
# continue with inference and write

# build keras callback for Kafka producer

class OutputCallback(tf.keras.callbacks.Callback):

    def __init__(self, batch_size, topic, servers):
        self._sequence = kafka_io.KafkaOutputSequence(
            topic=topic, servers=servers)
        self._batch_size = batch_size

    def on_predict_batch_end(self, batch, logs=None):
        ...
        self._sequence.setitem(index, class_names)

# <= inference with callback for streaming output

model.predict(
    dataset,
    callbacks=[OutputCallback(32, 'topic', 'localhost')])
# setup zookeeper

docker run -d \
  --net=host \
  -e ZOOKEEPER_CLIENT_PORT=2181 \
  -e ZOOKEEPER_TICK_TIME=2000 \
  -e ZOOKEEPER_SYNC_LIMIT=2 \
  confluentinc/cp-zookeeper:5.3.1

# setup kafka

docker run -d \
  --net=host \
  -e KAFKA_ZOOKEEPER_CONNECT=localhost:2181 \
  -e KAFKA_ADVERTISED_LISTENERS=PLAINTEXT://localhost:9092 \
  -e KAFKA_BROKER_ID=2 \
  -e KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR=1 \
  confluentinc/cp-kafka:5.3.1
# push mnist data into kafka,
# not truly needed, for completeness only.

```python
import numpy as np
import tensorflow as tf
import confluent_kafka as kafka

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
producer = kafka.Producer({'bootstrap.servers': 'localhost:9092'})

count = 0
for (x, y) in zip(x_train, y_train):
    producer.poll(0)
    producer.produce('x_train', x.tobytes())
    producer.produce('y_train', y.tobytes())
    count += 1
print("count(x, y): ", count)

producer.flush()
```
# push mnist data into kafka,
# not truly needed, for completeness only.

```python
import numpy as np
import tensorflow as tf
import confluent_kafka as kafka

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
producer = kafka.Producer({'bootstrap.servers': 'localhost:9092'})

count = 0
for (x, y) in zip(x_train, y_train):
    producer.poll(0)
    producer.produce('x_train', x.tobytes())
    producer.produce('y_train', y.tobytes())
    count += 1
print("count(x, y): ", count)

producer.flush()
```
Inference vs Training

Inference

Training
Schema with Avro
TensorFlow + Apache Kafka
TensorFlow + Apache Kafka

MQTT => Kafka => TensorFlow I/O => TensorFlow
SIG IO

TensorFlow
SIG IO

TensorFlow

Special Interest Group under TensorFlow
I/O, Data streaming, File formats
Apache Kafka, Apache Parquet/Arrow, Apache Ignite
Google Cloud BigQuery, Azure File Storage
WebP, TIFF, OpenEXR, HDR, FFmpeg

......
TensorFlow

https://github.com/tensorflow/io
THANK YOU