Distributed Deep Learning with Apache Spark and TensorFlow

Jim Dowling, Logical Clocks AB
The Cargobike Riddle
Dynamic Executors
(release GPUs when training finishes)

Spark & TensorFlow

Spark (Data Prep)

Spark Streaming (Monitoring Models)

Container (GPUs)
ML Pipeline HopsML

Potential Bottlenecks

- TensorFlow for Data Wrangling
- Single GPU
- No LB, Scale-Out

Data Collection → Data Transformation & Verification → Feature Extraction → Experimentation → Training → Test → Serving

- PySpark
- TensorFlow
- Kubernetes

Distributed Storage → HopsFS

Object Stores (S3, GCS), HDFS, Ceph
HopsML Spark/TensorFlow Arch

Dynamic Executors + Blacklisting for Fault Tolerance

Driver

Executor

Executor

Conda Envs

Conda Envs

TensorBoard/Logs

HopsFS

Model Serving
Why Distributed Deep Learning?

DEEP LEARNING ON A GPU IS NOT COOL......

YOU KNOW WHAT'S COOL? DEEP LEARNING ON A 1000 GPUS
All Roads Lead to Distribution

Distributed Deep Learning

Parallel Experiments
Hyper Parameter Optimization
Larger Training Datasets
Auto ML
Elastic Model Serving
Distributed Training
(Commodity) GPU Clusters

Parallel Experiments
Hyper Parameter Optimization
Larger Training Datasets
Auto ML
Elastic Model Serving
Distributed Training
(Commodity) GPU Clusters
Hyperparameter Optimization

(Because DL Theory Sucks!)
Faster Experimentation

GPU Servers

LR: 0.0001
NL: 5
Error: 0.35

LR: 0.0001
NL: 7
Error: 0.34

LR: 0.0005
NL: 5
Error: 0.38

LR: 0.0005
NL: 10
Error: 0.37

LR: 0.001
NL: 5
Error: 0.31

LR: 0.001
NL: 9
Error: 0.36

HyperParameter Optimization

Hyperparameters

TensorFlow Program

Blacklist Executor

LearningRate (LR): [0.0001-0.001]
NumOfLayers (NL): [5-10]
......
Declarative or API Approach?

• Declarative Hyperparameters in external files
  – Vizier/CloudML (yaml)
  – Sagemaker (json)*

• API-Driven
  – Databrick’s MLFlow
  – HopsML

Google CloudML Hyperparameters

scaleTier: CUSTOM
workerCount: 9
parameterServerCount: 3
hyperparameters:
  maxParallelTrials: 1
params:
  - parameterName: hidden1
    type: INTEGER
    minValue: 40
    maxValue: 400
  scaleType: UNIT_LINEAR_SCALE

- parameterName: numRnnCells
  type: DISCRETE
discreteValues:
  - 1
  - 2

- parameterName: rnnCellType
  type: CATEGORICAL
categoricalValues:
  - BasicRNNCell
  - GRUCell
  - LSTMCell

https://cloud.google.com/ml-engine/docs/tensorflow/using-hyperparameter-tuning
def train(learning_rate, dropout):

[TensorFlow Code here]

args_dict = {'learning_rate': [0.001, 0.005, 0.01], 'dropout': [0.5, 0.6]}

experiment.launch(train, args_dict)

Launch 6 Spark Executors
Distributed Training
Model Parallelism

One Big Model on 3 GPUs

GPU3
GPU2
GPU1

Layer100-199
Layer200-299
Layers1-100

Training/Test Data

Predictions

Data Parallelism

(Synchronous Stochastic Gradient Descent (SGD))

Copy of the Model on each GPU

GPU0
GPU1
GPU99

Barrier
Gradients

Aggregate

New Model

32 files

32 files

32 files

Batch Size of 3200

Distributed Filesystem
Data Parallel Distributed Training

GPU Servers

Generalization Error

Training Time
Frameworks for Distributed Training
TF_CONFIG

Bring your own Distribution!

1. Start all processes for P1, P2, G1-G4 yourself

2. Collect all IP addresses in TF_CONFIG along with GPU device IDs.
RingAllReduce (Horovod)

- Bandwidth optimal
- Automatically builds the ring (MPI)
- Supported by HopsML and Databricks’ HorovodEstimator
1. Start all processes for G1-G4 yourself
2. Collect all IP addresses in TF_CONFIG along with GPU device IDs.

Available from TensorFlow 1.11
HopsML CollectiveAllReduceStrategy

- Uses Spark/YARN to add distribution to TensorFlow’s CollectiveAllReduceStrategy
  - Automatically builds the ring (Spark/YARN)

https://github.com/logicalclocks/hops-util-py
## Collective AllReduce vs Horovod Benchmark

<table>
<thead>
<tr>
<th>TensorFlow:</th>
<th>1.11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model:</td>
<td>Inception v1</td>
</tr>
<tr>
<td>Dataset:</td>
<td>imagenet (synthetic)</td>
</tr>
<tr>
<td>Batch size:</td>
<td>256 global, 32.0 per device</td>
</tr>
<tr>
<td>Num batches:</td>
<td>100</td>
</tr>
<tr>
<td>Optimizer:</td>
<td>Momentum</td>
</tr>
<tr>
<td>Num GPUs:</td>
<td>8</td>
</tr>
<tr>
<td>AllReduce:</td>
<td>collective</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step</th>
<th>Img/sec</th>
<th>total_loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>images/sec: 2972.4 +/- 0.0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>images/sec: 3008.9 +/- 8.9</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>images/sec: 2998.6 +/- 4.3</td>
<td></td>
</tr>
</tbody>
</table>

Total images/sec: **2993.52**

---

<table>
<thead>
<tr>
<th>TensorFlow:</th>
<th>1.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model:</td>
<td>Inception v1</td>
</tr>
<tr>
<td>Dataset:</td>
<td>imagenet (synthetic)</td>
</tr>
<tr>
<td>Batch size:</td>
<td>256 global, 32.0 per device</td>
</tr>
<tr>
<td>Num batches:</td>
<td>100</td>
</tr>
<tr>
<td>Optimizer:</td>
<td>Momentum</td>
</tr>
<tr>
<td>Num GPUs:</td>
<td>8</td>
</tr>
<tr>
<td>AllReduce:</td>
<td>horovod</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step</th>
<th>Img/sec</th>
<th>total_loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>images/sec: 2816.6 +/- 0.0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>images/sec: 2808.0 +/- 10.8</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>images/sec: 2806.9 +/- 3.9</td>
<td></td>
</tr>
</tbody>
</table>

Total images/sec: **2803.69**

---

https://groups.google.com/a/tensorflow.org/forum/#i!topic/discuss/7T05tNV08Us
CollectiveAllReduce vs Horovod Benchmark

TensorFlow: 1.11
Model: VGG19
Dataset: imagenet (synthetic)
Batch size: 256 global, 32.0 per device
Num batches: 100
Optimizer: Momentum
Num GPUs: 8
AllReduce: collective

<table>
<thead>
<tr>
<th>Step</th>
<th>Img/sec</th>
<th>total_loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>images/sec: 634.4 +/- 0.0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>images/sec: 635.2 +/- 0.8</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>images/sec: 635.0 +/- 0.5</td>
<td></td>
</tr>
</tbody>
</table>

-------------------------------------------------------------
total images/sec: 634.80

TensorFlow: 1.7
Model: VGG19
Dataset: imagenet (synthetic)
Batch size: 256 global, 32.0 per device
Num batches: 100
Optimizer: Momentum
Num GPUs: 8
AllReduce: horovod

<table>
<thead>
<tr>
<th>Step</th>
<th>Img/sec</th>
<th>total_loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>images/sec: 583.01 +/- 0.0</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>images/sec: 582.22 +/- 0.1</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>images/sec: 583.61 +/- 0.2</td>
<td></td>
</tr>
</tbody>
</table>

-------------------------------------------------------------
total images/sec: 583.61

https://groups.google.com/a/tensorflow.org/forum/#!topic/discuss/7T05tNV08Us
Reduction in LoC for Dist Training

<table>
<thead>
<tr>
<th>Released</th>
<th>Framework</th>
<th>Lines of Code in Hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>March 2016</td>
<td>Distributed TensorFlow</td>
<td>~1000</td>
</tr>
<tr>
<td>Feb 2017</td>
<td>TensorFlowOnSpark*</td>
<td>~900</td>
</tr>
<tr>
<td>Jan 2018</td>
<td>Horovod (Keras)*</td>
<td>~130</td>
</tr>
<tr>
<td>June 2018</td>
<td>Databricks’ HorovodEstimator</td>
<td>~100</td>
</tr>
<tr>
<td>Sep 2018</td>
<td>HopsML (Keras/CollectiveAllReduce)*</td>
<td>~100</td>
</tr>
</tbody>
</table>

*https://github.com/logicalclocks/hops-examples
**https://docs.azuredatabricks.net/_static/notebooks/horovod-estimator.html
def distributed_training():
    def input_fn():  # return dataset
        model = …
        optimizer = …
        model.compile(…)
        rc = tf.estimator.RunConfig('CollectiveAllReduceStrategy')
        keras_estimator = tf.keras.estimator.model_to_estimator(….)
        tf.estimator.train_and_evaluate(keras_estimator, input_fn)

    experiment.allreduce(distributed_training)
HopsML CollectiveAllReduceStrategy

• Scale to 10s or 100s of GPUs on Hops
• Generate Tensorboard Logs in HopsFS
• Checkpoint to HopsFS
• Save a trained model to HopsFS
• Experiment History
  – Reproducible training
def distributed_training():
    from hops import tensorboard
    model_dir = tensorboard.logdir()
    def input_fn(): # return dataset
        model = ...
        optimizer = ...
        model.compile(...)
        rc = tf.estimator.RunConfig('CollectiveAllReduceStrategy')
        keras_estimator = keras->model_to_estimator(model_dir)
        tf.estimator.train_and_evaluate(keras_estimator, input_fn)
    experiment.allreduce(distributed_training)
def distributed_training():
    from hops import devices
    def input_fn(): # return dataset
        model = ...
        optimizer = ...
        model.compile(...)
        est->RunConfig(num_gpus_per_worker=devices.get_num_gpus())
        keras_estimator = keras->model_to_estimator(...)
        tf.estimator.train_and_evaluate(keras_estimator, input_fn)

eperiment.allreduce(distributed_training)
def distributed_training():
    def input_fn(): # return dataset
        model = ...
        optimizer = ...
        model.compile(...)
        rc = tf.estimator.RunConfig(‘CollectiveAllReduceStrategy’)
        keras_estimator = keras->model_to_estimator(...)
        tf.estimator.train_and_evaluate(keras_estimator, input_fn)

notebook = hdfs.project_path()+’/Jupyter/Experiment/inc.ipynb’
experiment.allreduce(distributed_training, name=’inception’,
    description=’A inception example with hidden layers’,
    versioned_resources=[notebook])
Experiment Versioning/History/Reproduce
The Data Layer
FEED_DICT is single threaded (Python GIL)
TensorFlow Dataset API does not support DFs

- Petastorm (Uber) for Parquet->TensorFlow training
- What about Datafiles (.csv, images, txt)?
HopsFS

- HDFS derivative with Distributed Metadata
  - 16X HDFS throughput.
  - Winner IEEE Scale Prize 2017
- Integrates NVMe disks transparently*
  - Store small files (replicated) on NVMe hardware

*Size Matters: Improving the Performance of Small Files in Hadoop, Middleware 2018. Niazi et al
Model Serving on Kubernetes
Kubernetes Model Serving

• Elastic scaling for model serving

• Supports:
  • Fault tolerance
  • Rolling release new models
  • Autoscaling
Model Monitoring with Spark Streaming

• Log model inference requests/results to Kafka
• Spark monitors model performance and input data
• When to retrain?
  — If you look at the input data and use covariant shift to see when it deviates significantly from the data that was used to train the model on.
Orchestrating HopsML Workflows

Data Collection
Data Transformation & Verification
Feature Extraction
Experimentation
Training
Test
Serving

Airflow (Hopsworks Operator)
Summary

• The future of Deep Learning is Distributed
  https://www.oreilly.com/ideas/distributed-tensorflow

• Hops is a new Data Platform with first-class support for Python / Deep Learning / ML / Data Governance / GPUs

  hopshadoop
  logicalclocks