TensorFlowOnSpark

Scalable TensorFlow Learning on Spark Clusters

Lee Yang, Andrew Feng

Yahoo Big Data ML Platform Team

What is TensorFlowOnSpark?

YAHOO!

Why TensorFlowOnSpark at Yahoo?

- Major contributor to open-source Hadoop ecosystem
  - Originators of Hadoop (2006)
  - An early adopter of Spark (since 2013)
  - Open-sourced CaffeOnSpark (2016)
    - *CaffeOnSpark Update: Recent Enhancements and Use Cases*
    - Wednesday @ 12:20pm by Mridul Jain & Jun Shi
- Large investment in production clusters
  - Tens of clusters
  - Thousands of nodes per cluster
- Massive amounts of data
  - Petabytes of data

Private ML Clusters

Machine-learning at scale

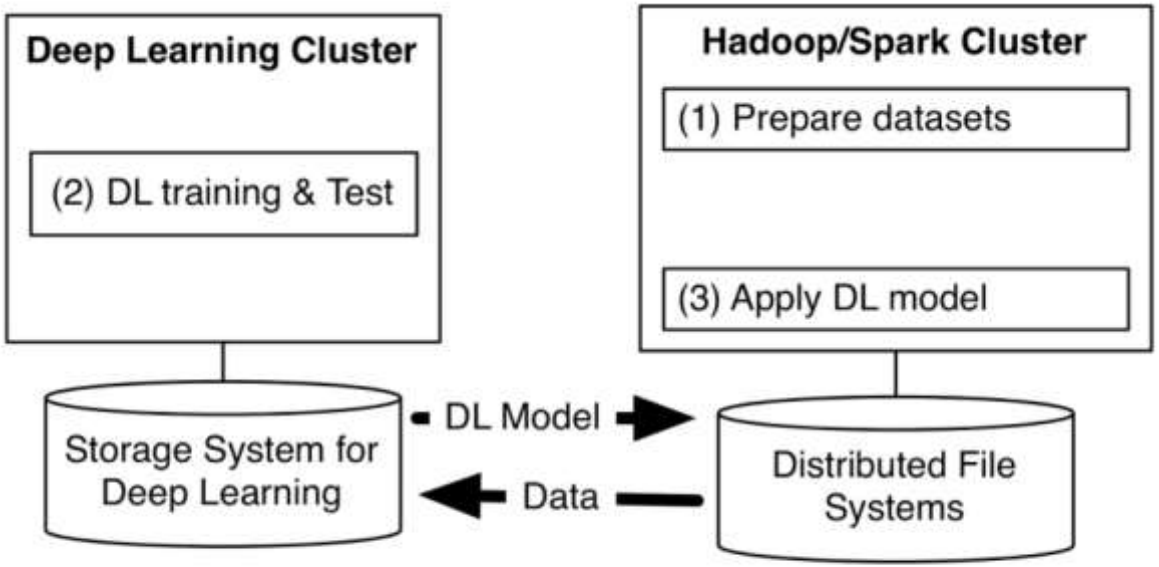


Figure 1: ML Pipeline with multiple programs on separated clusters

YAHOO!

Why TensorFlowOnSpark?

Machine-learning at scale

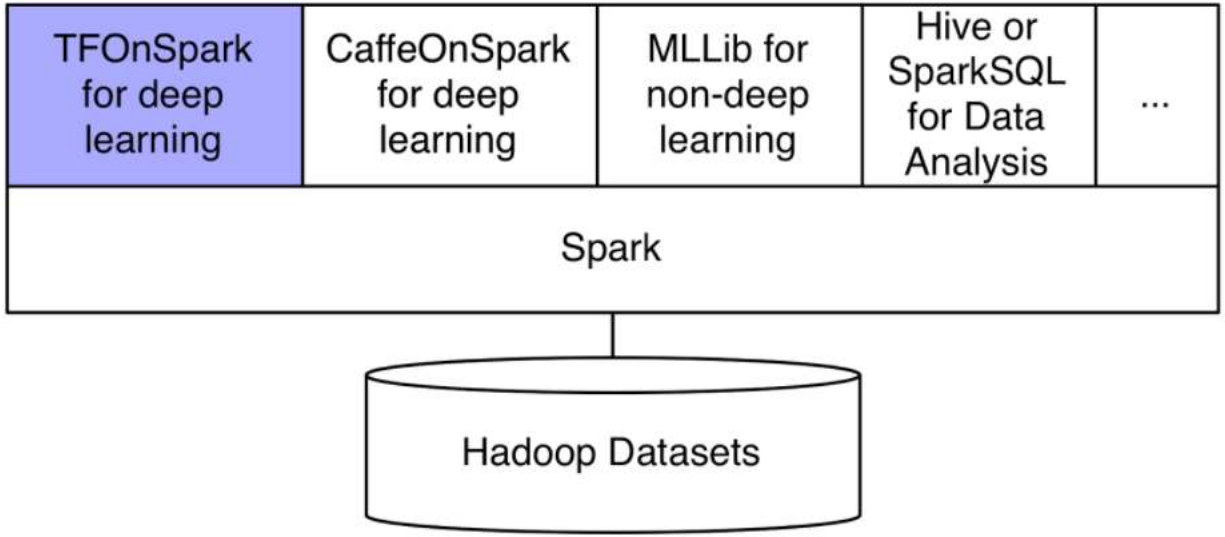


Figure 2: TensorFlowOnSpark for deep learning on Spark clusters

Scaling

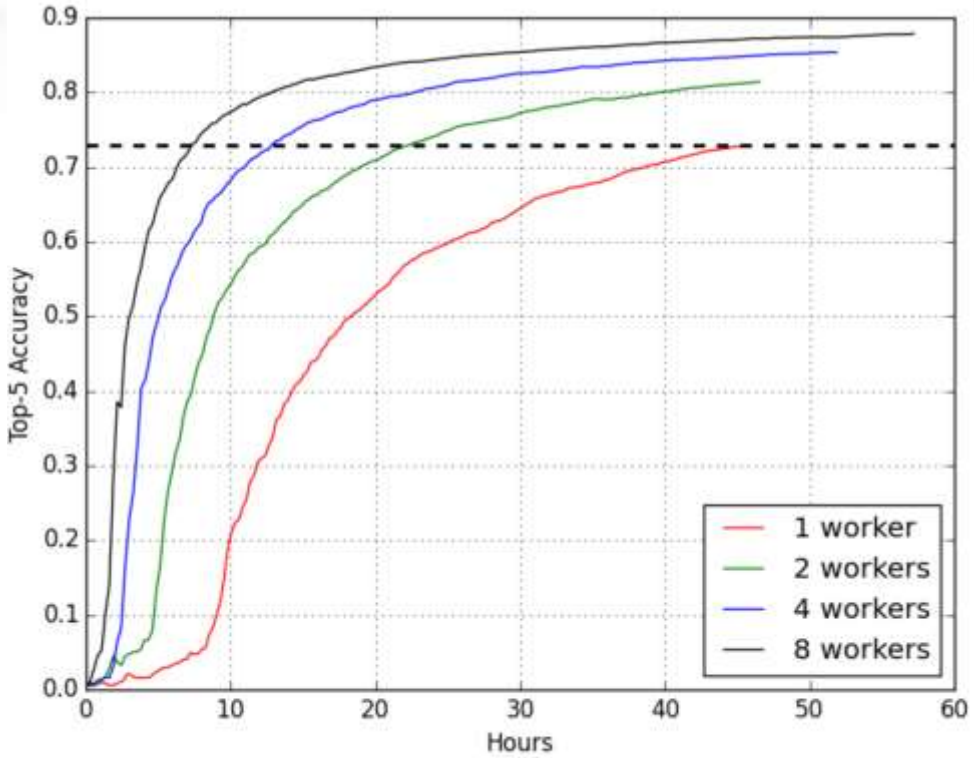
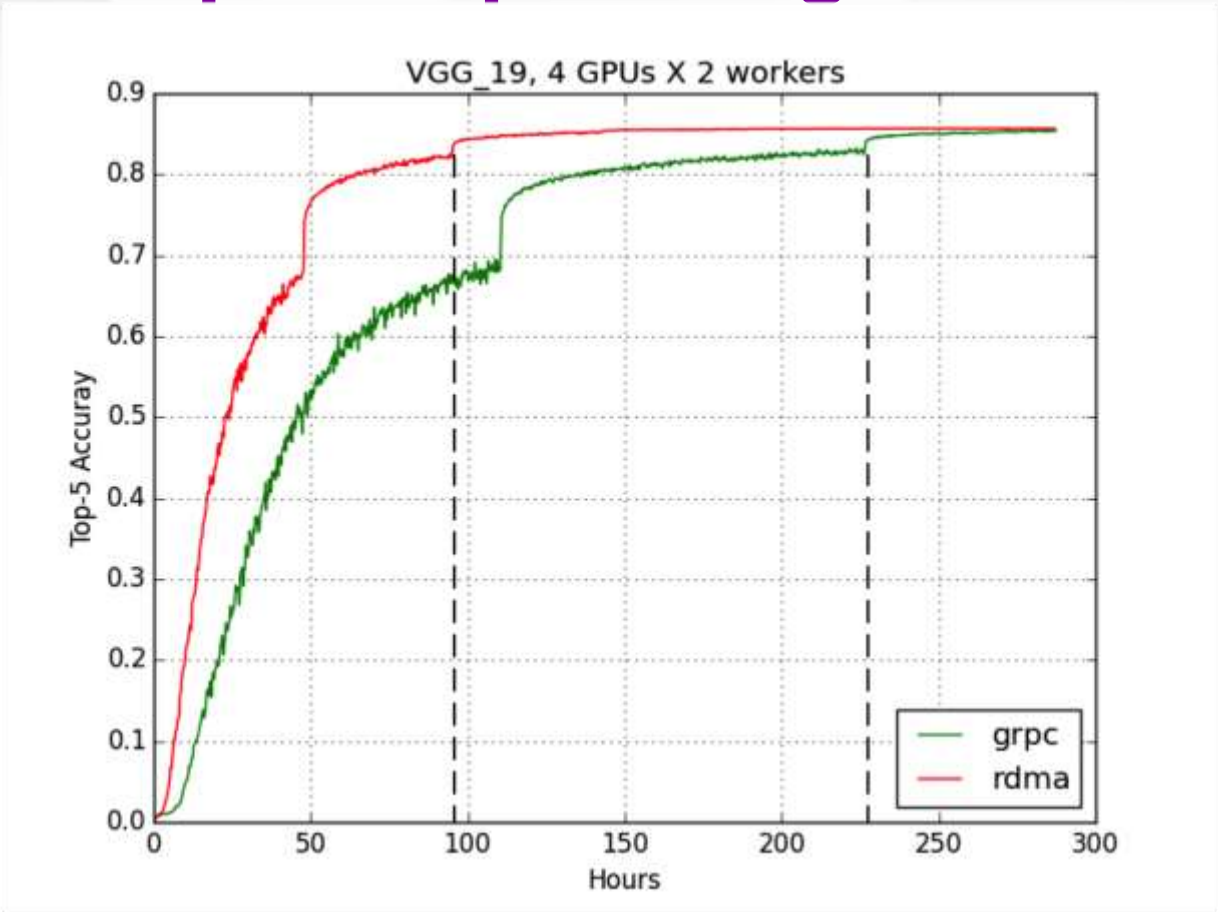


Figure 4: TFoS training of Inception networks

Near-linear scaling

YAHOO!

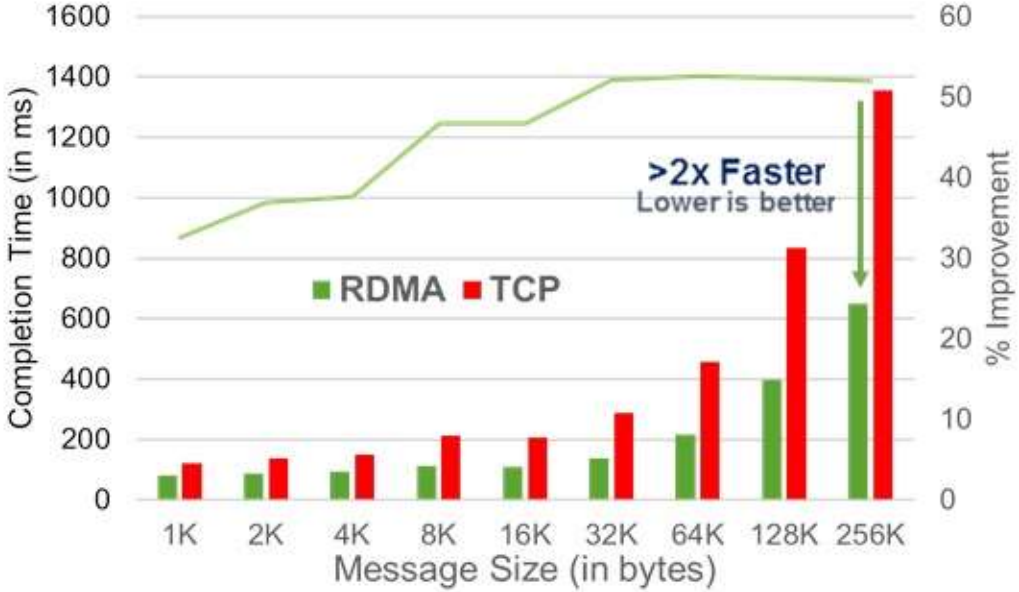
RDMA Speedup over gRPC



2.4X faster

YAHOO!

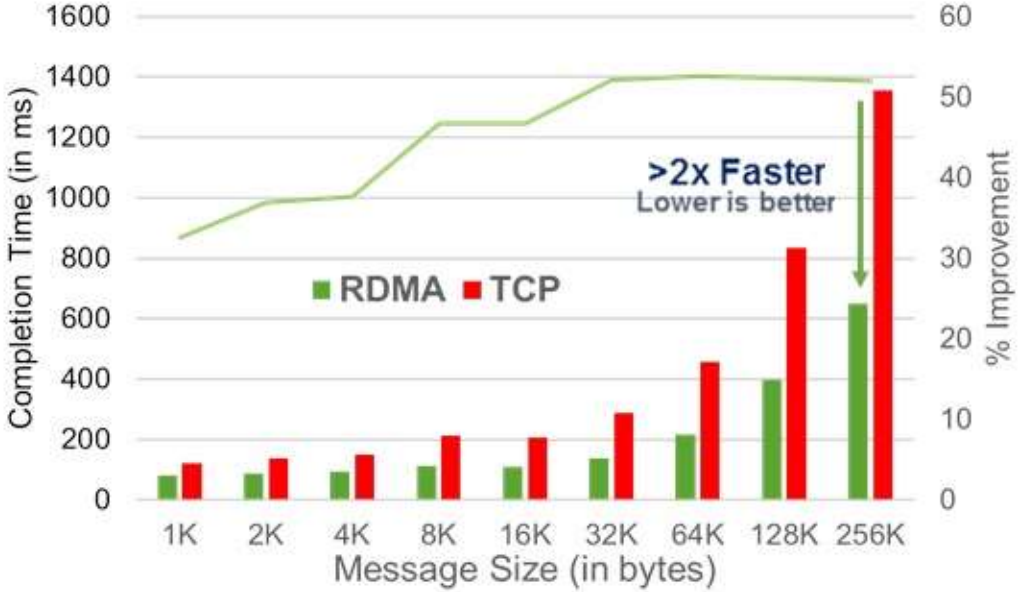
RDMA Speedup over gRPC



http://www.mellanox.com/solutions/machine-learning/tensorflow.php

YAHOO!

RDMA Speedup over gRPC



http://www.mellanox.com/solutions/machine-learning/tensorflow.php

YAHOO!

TensorFlowOnSpark Design Goals

- Scale up existing TF apps with minimal changes
- Support all current TensorFlow functionality
  - Synchronous/asynchronous training
  - Model/data parallelism
  - TensorBoard
- Integrate with existing HDFS data pipelines and ML algorithms
  - ex. Hive, Spark, MLlib

TensorFlowOnSpark

- Pyspark wrapper of TF app code
- Launches distributed TF clusters using Spark executors
- Supports TF data ingestion modes
  - feed_dict – RDD.mapPartitions()
  - queue_runner – direct HDFS access from TF
- Supports TensorBoard during/after training
- Generally agnostic to Spark/TF versions

Supported Environments

- Python 2.7 - 3.x
- Spark 1.6 - 2.x
- TensorFlow 0.12, 1.x
- Hadoop 2.x

YAHOO!

Architectural Overview

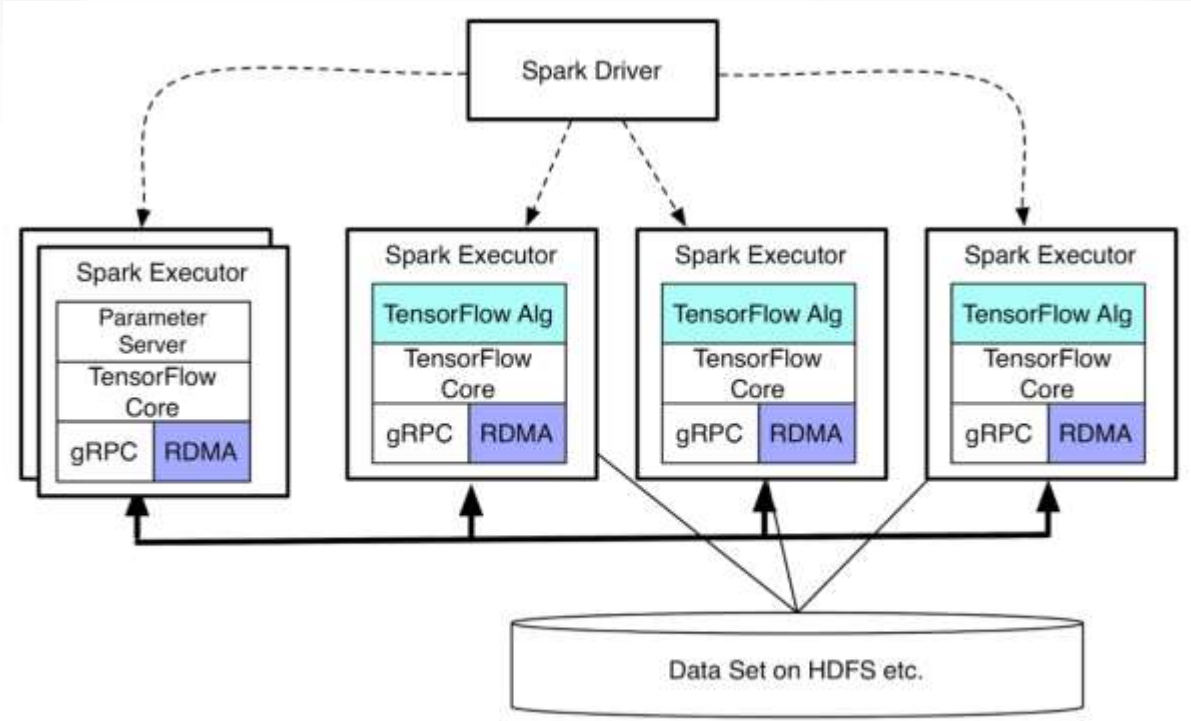


Figure 3: TensorFlowOnSpark system architecture

YAHOO!

TensorFlowOnSpark Basics

1. **Launch** TensorFlow cluster
2. **Feed data** to TensorFlow app
3. **Shutdown** TensorFlow cluster

API Example

```
cluster = TFCluster.run(sc, map_fn, args, num_executors,
num_ps, tensorboard, input_mode)

cluster.train(dataRDD, num_epochs=0)

cluster.inference(dataRDD)

cluster.shutdown()
```

YAHOO!

Conversion Example

```
# diff -w eval_image_classifier.py
20a21,27
> from pyspark.context import SparkContext
> from pyspark.conf import SparkConf
> from tensorflowonspark import TFCluster, TFNode
> import sys
>
> def main_fun(argv, ctx):
27a35,36
>   sys.argv = argv
>
84,85d92
<
< def main(_):
88a96,97
>   cluster_spec, server = TFNode.start_cluster_server(ctx)
>
191c200,204
<   tf.app.run()
---
>   sc = SparkContext(conf=SparkConf().setAppName("eval_image_classifier"))
>   num_executors = int(sc._conf.get("spark.executor.instances"))
>   cluster = TFCluster.run(sc, main_fun, sys.argv, num_executors, 0, False, TFCluster.InputMode.TENSORFLOW)
>   cluster.shutdown()
```

YAHOO!

Input Modes

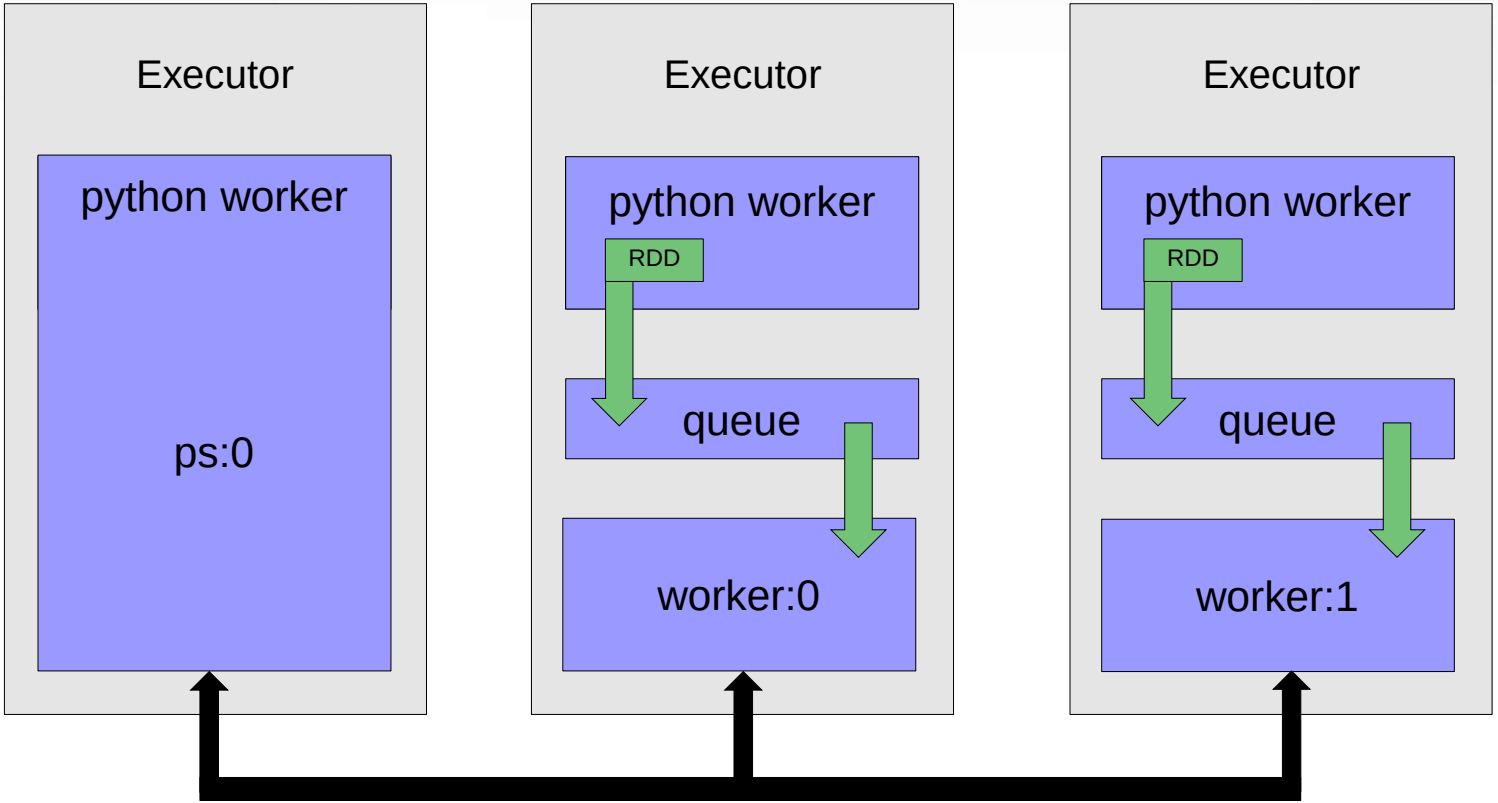
- InputMode.SPARK

  HDFS → RDD.mapPartitions → feed_dict

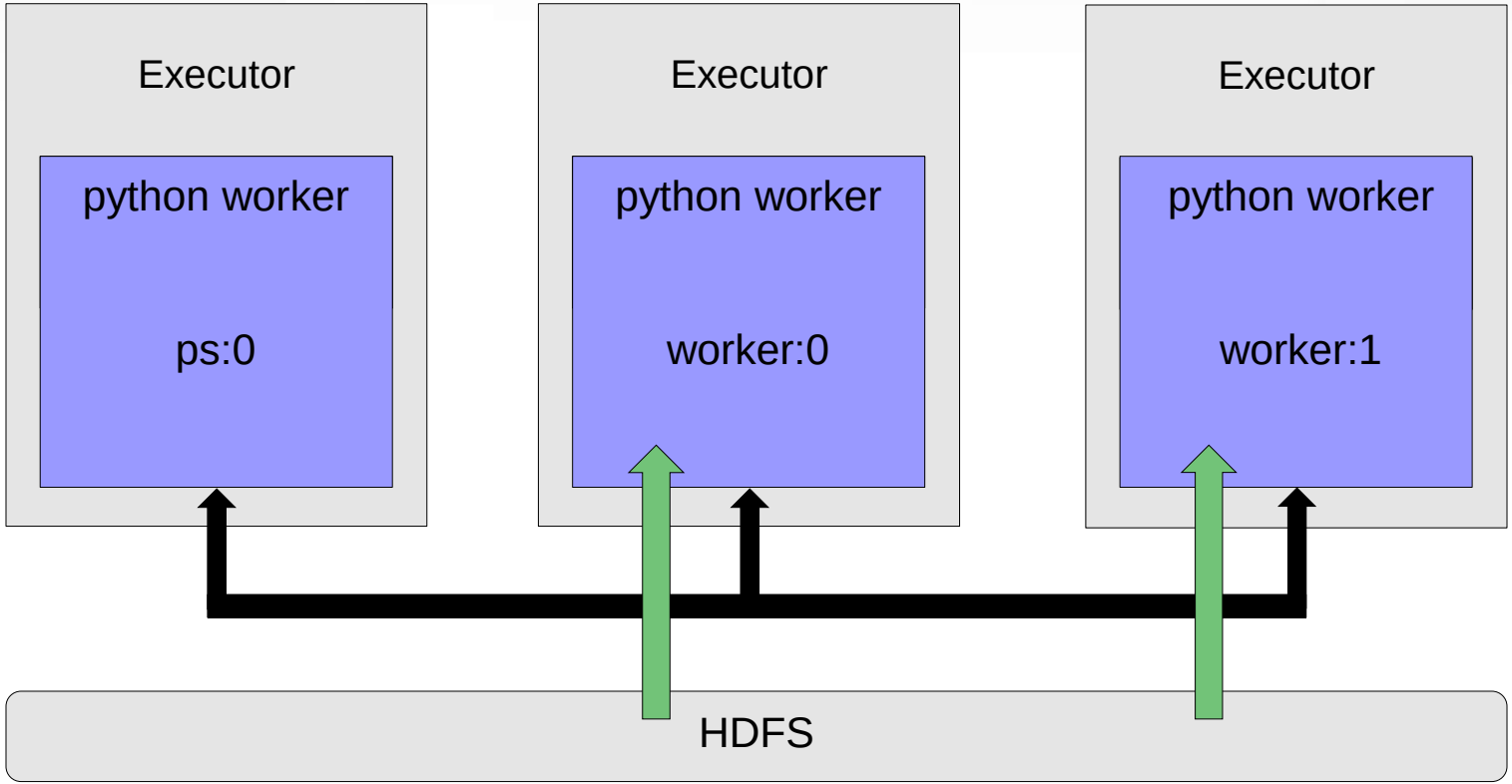
- InputMode.TENSORFLOW

  TFReader + QueueRunner ← HDFS

InputMode.SPARK



InputMode.TENSORFLOW



YAHOO!

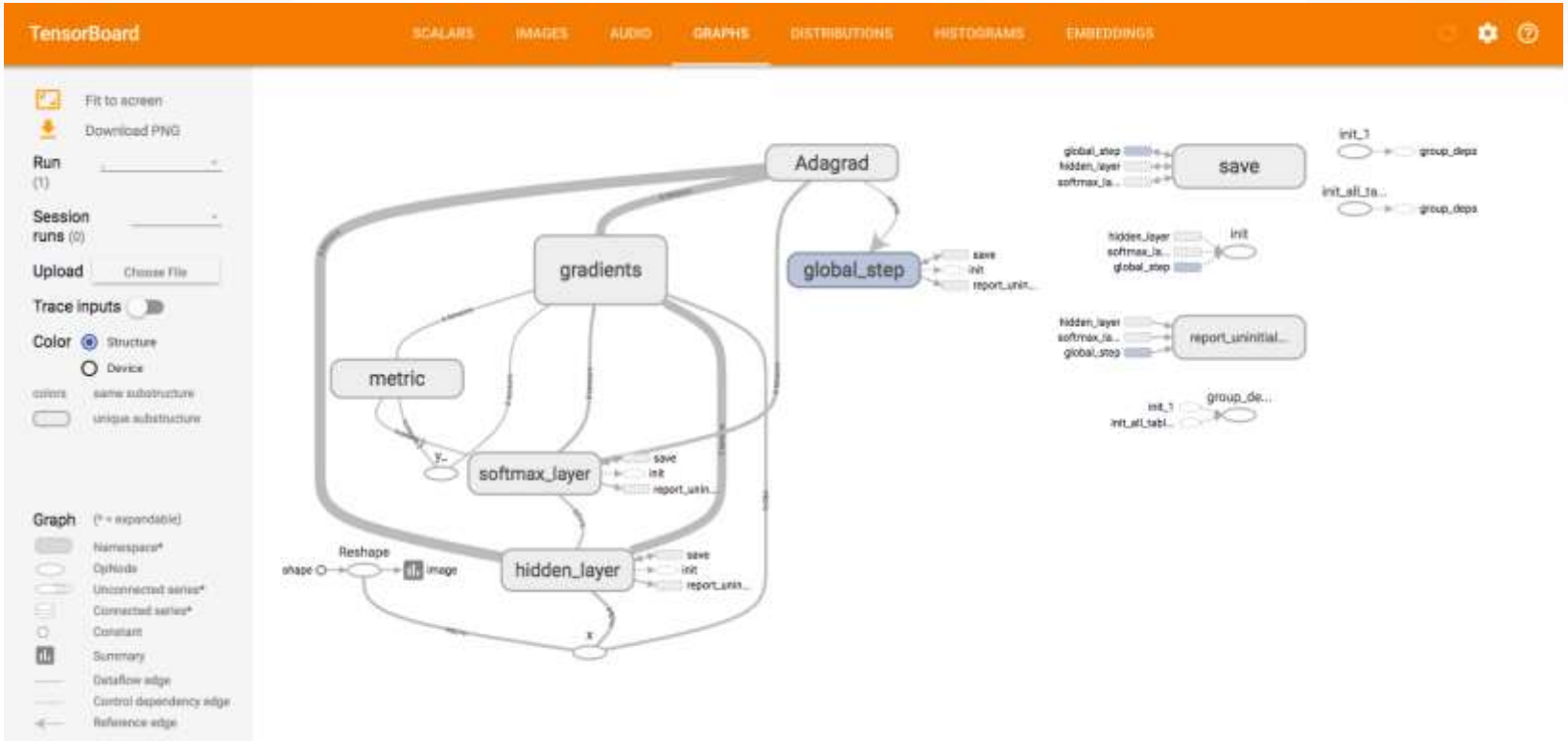
Failure Recovery

- TF Checkpoints written to HDFS
- InputMode.SPARK
  - TF worker runs in background
  - RDD data feeding tasks can be retried
  - However, TF worker failures will be “hidden” from Spark
- InputMode.TENSORFLOW
  - TF worker runs in foreground
  - TF worker failures will be retried as Spark task
  - TF worker restores from checkpoint

Failure Recovery

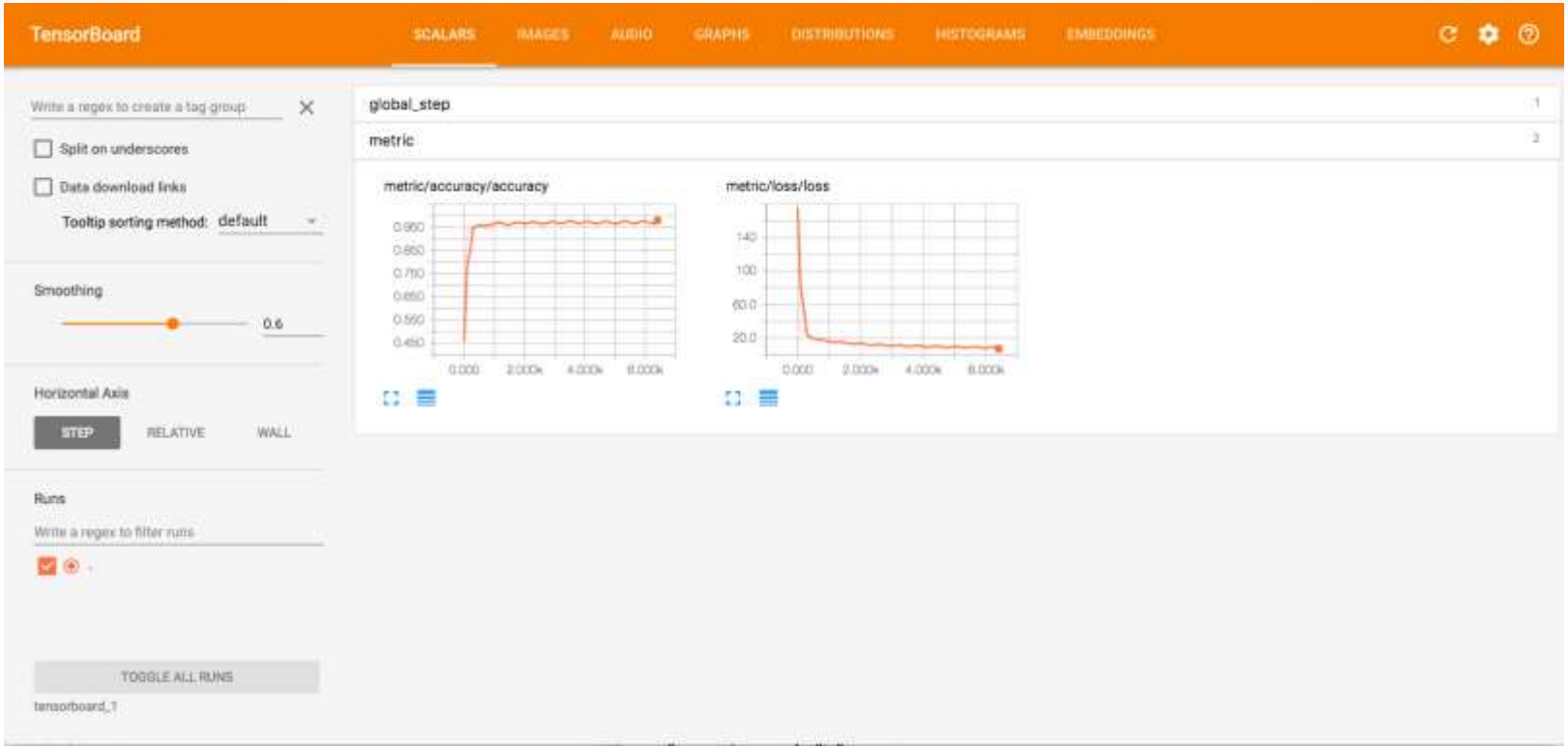
- Executor failures are problematic
  - e.g. pre-emption
  - TF cluster_spec is statically-defined at startup
  - YARN does not re-allocate on same node
  - Even if possible, port may no longer be available.
- Need dynamic cluster membership
  - Exploring options w/ TensorFlow team

TensorBoard



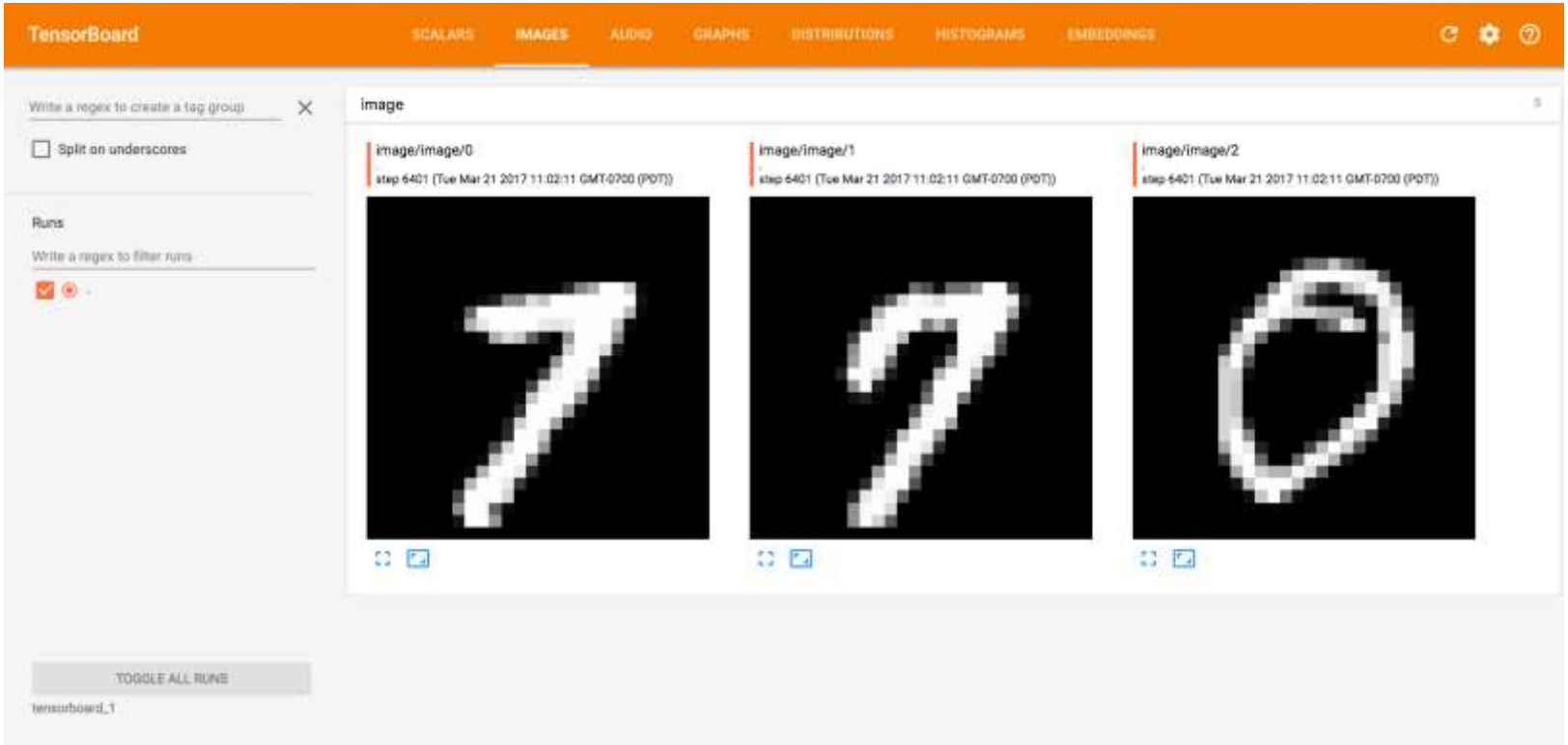
YAHOO!

TensorBoard



YAHOO!

TensorBoard



YAHOO!

TensorFlow App Development

Experimentation Phase

- Single-node
- Small scale data
- TensorFlow APIs

```
tf.Graph
tf.Session
tf.InteractiveSession
```

YAHOO!

TensorFlow App Development

Scaling Phase

- Multi-node
- Medium scale data (local disk)
- Distributed TensorFlow APIs

```
tf.train.ClusterSpec
tf.train.Server
tf.train.Saver
tf.train.Supervisor
```

YAHOO!

TensorFlow App Development

Production Phase

- Cluster deployment
- Upstream data pipeline
- Model training w/ TensorFlowOnSpark APIs

```
TFCluster.run
TFNode.start_cluster_server
TFCluster.shutdown
```

- Production inference w/ TensorFlow Serving

Example Usage

https://github.com/yahoo/TensorFlowOnSpark/tree/master/examples

Common Gotchas

- Single node/task per executor
- HDFS access (native libs/env)
- Why doesn’t algorithm X scale linearly?

What's New?

- Community contributions
  - CDH compatibility
  - TFNode.DataFeed
  - Bug fixes
- RDMA merged into TensorFlow repository
- Registration server
- Spark streaming
- Pip packaging

YAHOO!

Spark Streaming

```
from pyspark.streaming import StreamingContext
ssc = StreamingContext(sc, 10)
images = sc.textFile(args.images).map(lambda ln: parse(ln))
stream = ssc.textFileStream(args.images)
imageRDD = stream.map(lambda ln: parse(ln))
cluster = TFCluster.run(sc, map_fun, args,…)
predictionRDD = cluster.inference(imageRDD)
predictionRDD.saveAsTextFile(args.output)
predictionRDD.saveAsTextFiles(args.output)
ssc.start()
cluster.shutdown(ssc)
```

YAHOO!

Pip packaging

```
pip install tensorflowonspark
${SPARK_HOME}/bin/spark-submit \
    --master ${MASTER} \
    --py-files ${TFoS_HOME}/examples/mnist/spark/mnist_dist.py \
    --archives ${TFoS_HOME}/tfspark.zip \
    ${TFoS_HOME}/examples/mnist/spark/mnist_spark.py \
    --cluster_size ${SPARK_WORKER_INSTANCES} \
    --images examples/mnist/csv/train/images \
    --labels examples/mnist/csv/train/labels \
    --format csv \
    --mode train \
    --model mnist_model
```

YAHOO!

Demo

Next Steps

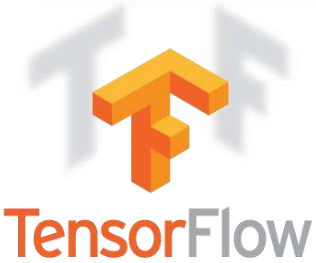
- TF/Keras Layers
- Failure recovery w/ dynamic cluster management (e.g. registration server)

Summary

TFoS brings deep learning to big-data clusters

- TensorFlow: 0.12 -1.x
- Spark: 1.6-2.x
- Cluster manager: YARN, Standalone, Mesos
- EC2 image provided
- RDMA in TensorFlow

Thanks!



And our open-source contributors!

Questions?

https://github.com/yahoo/TensorFlowOnSpark