TensorFlowOnSpark:
Scalable TensorFlow Learning on Spark Clusters

Lee Yang, Andy Feng

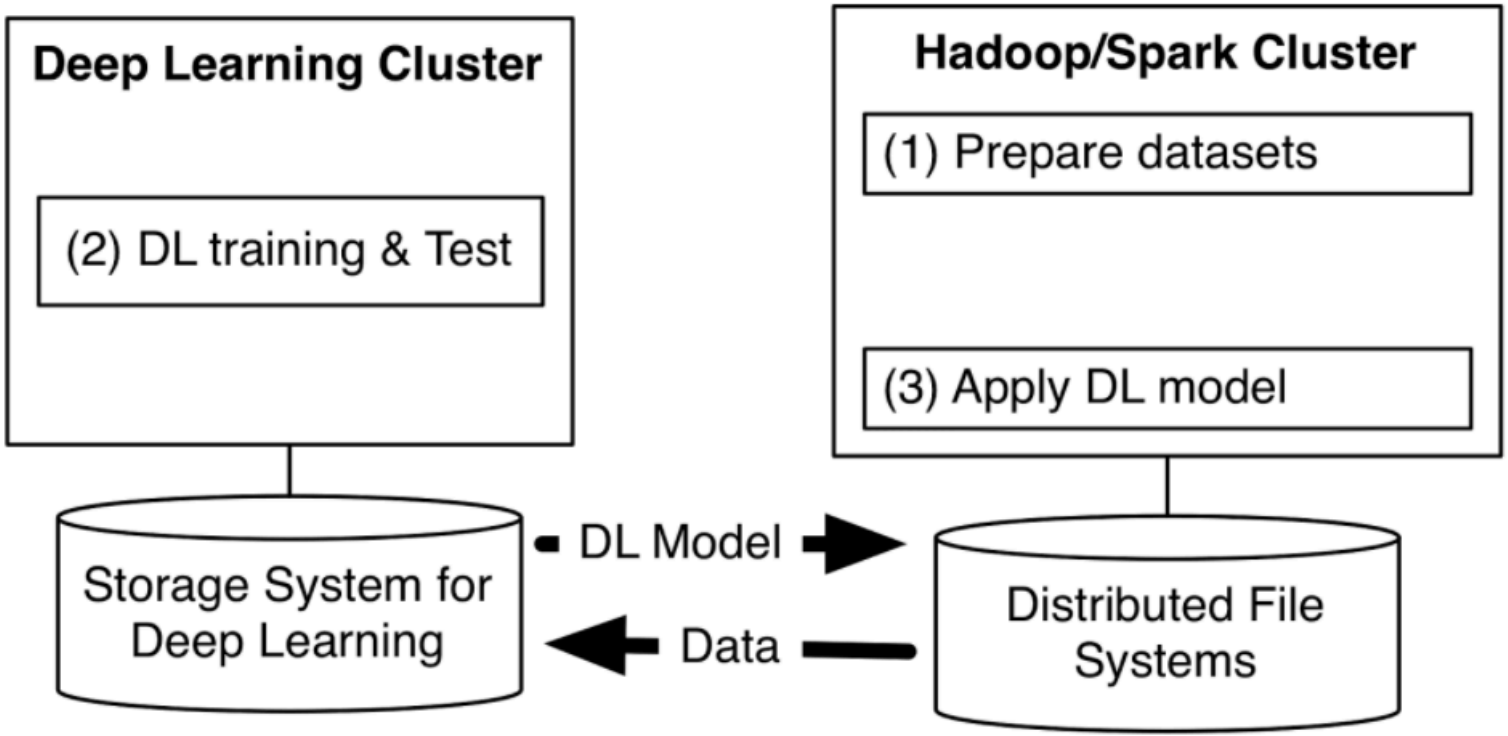
What is TensorFlowOnSpark?

- github.com/yahoo/TensorFlowOnSpark

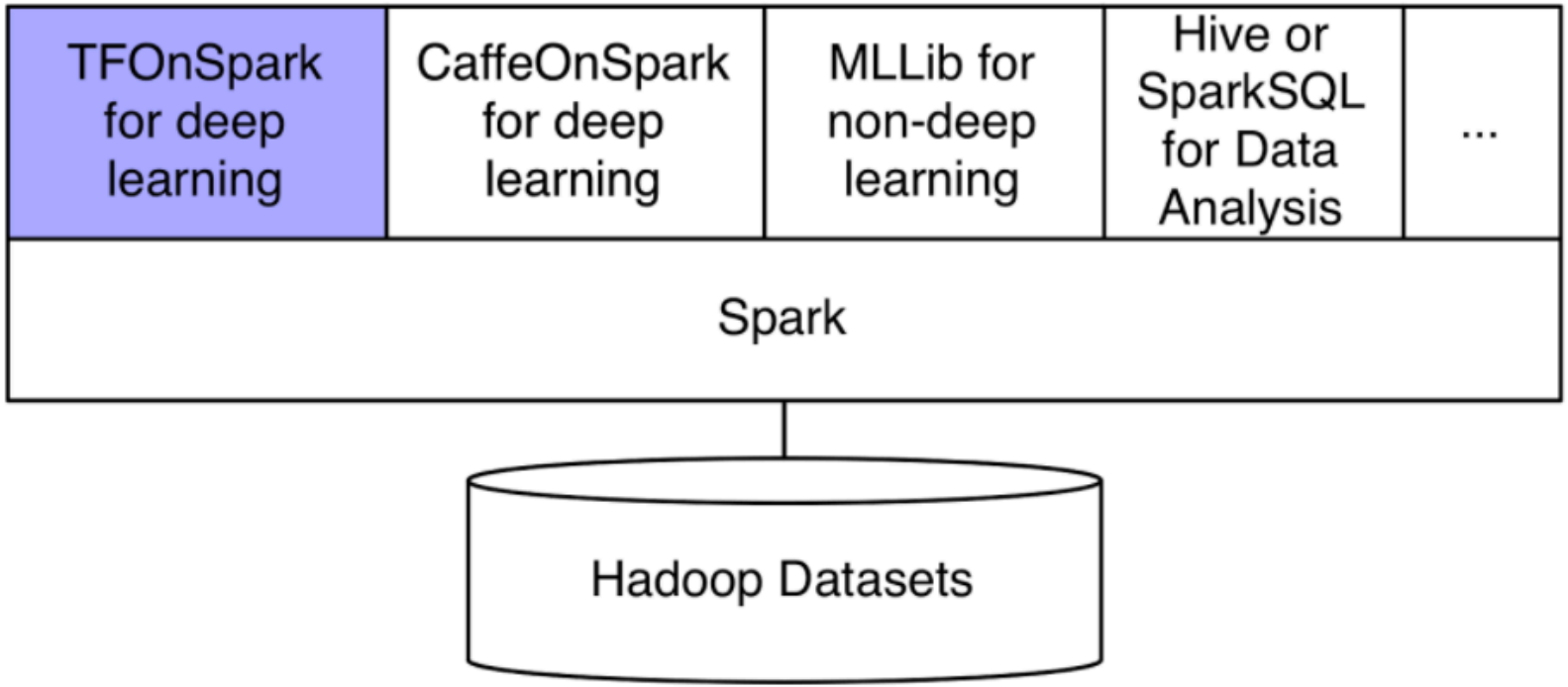
Why TensorFlowOnSpark at Yahoo?

- Major contributor to open-source Hadoop ecosystem
  - Originators of Hadoop (2006)
  - An early adopter of Spark (since 2013)
  - Open-sourced CaffeOnSpark (2016)
- Large investment in production clusters
  - Tens of clusters
  - Thousands of nodes per cluster
- Massive amounts of data
  - Petabytes of data

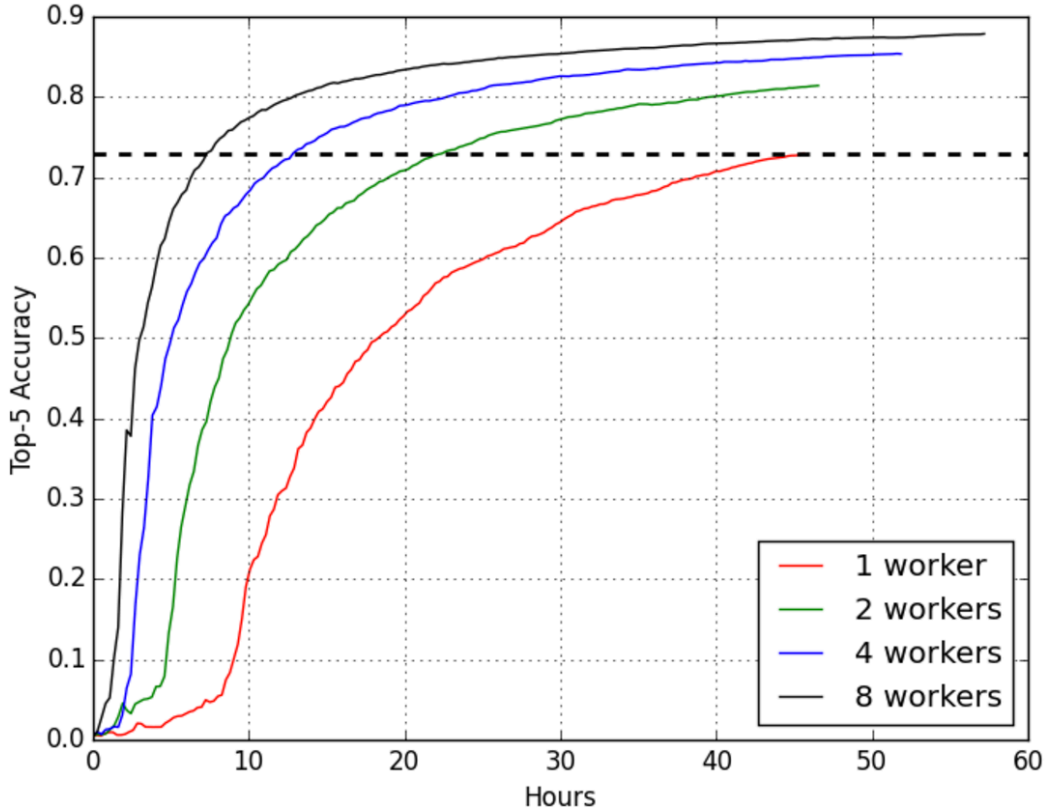
Separated DL Cluster



Why TensorFlowOnSpark?



Scaling as Dedicated TF Clusters



Near-linear scaling

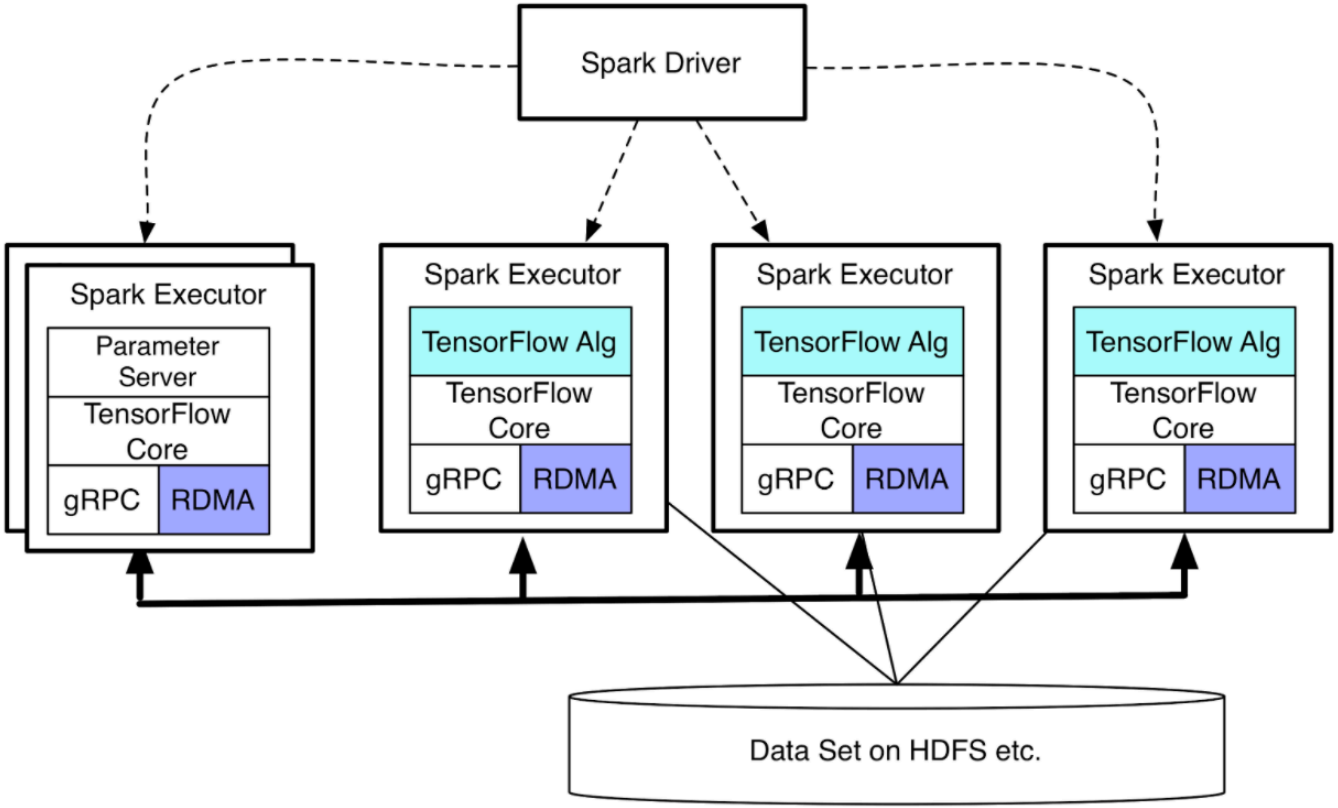
TensorFlowOnSpark Design Goals

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

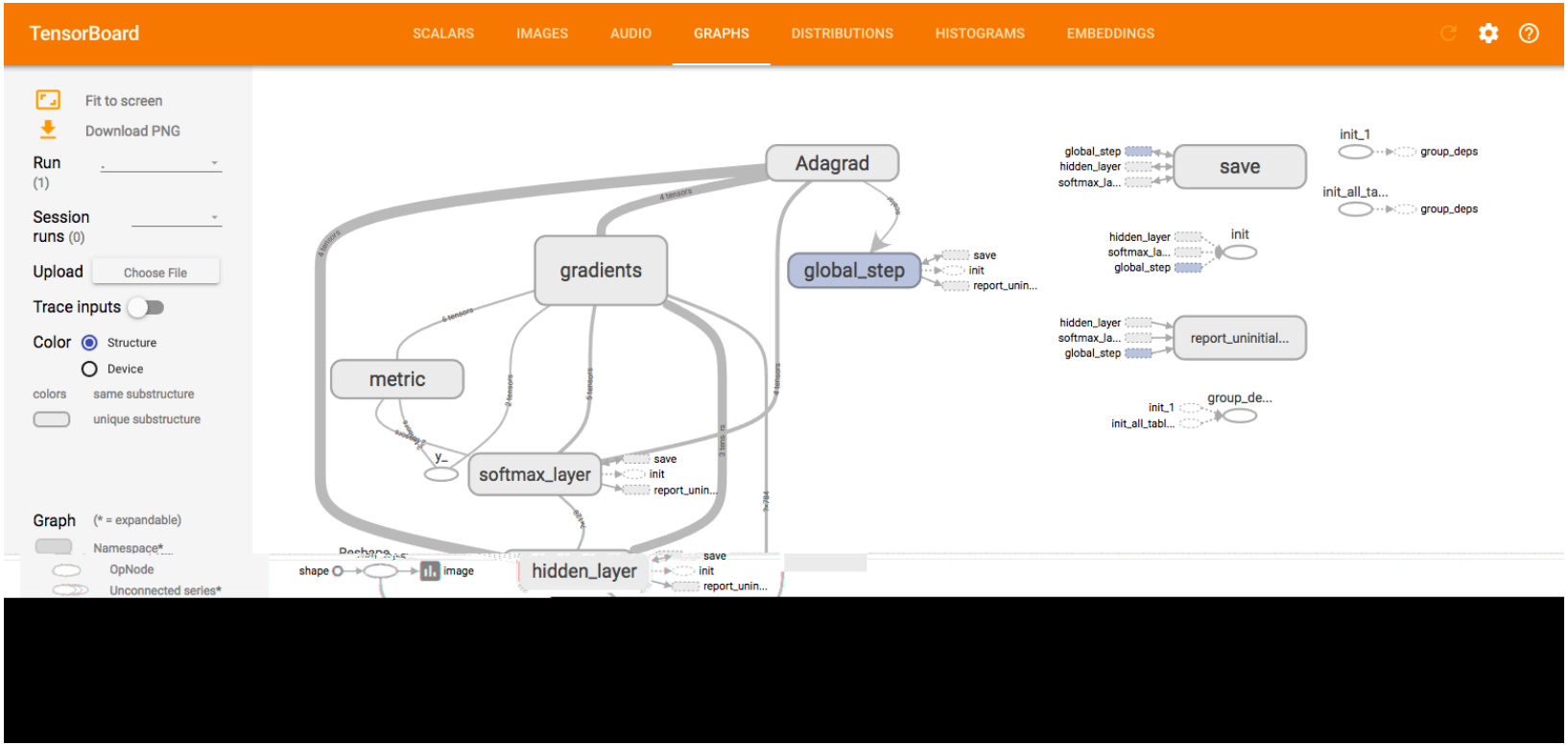
TensorFlowOnSpark

- Pyspark wrapper of TF app code
- Launches distributed TF clusters using Spark
- 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

Architectural Overview



TensorBoard



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()
```

Pipeline API*: Training

```
model = TFEstimator(map_fn, tf_args)
    .setInputMapping({“image”: “placeholder_X”,
                      “label”: “placeholder_Y”})
    .setClusterSize(10).setNumPS(1)
    .setModelDir(“my_model_checkpoints”)
    .setExportDir(“my_saved_model_dir”)
    .setEpochs(10)
    .setInputMode(InputMode.SPARK) //or InputMode.TENSORFLOW
    .fit(train_df)
```

- * Jointly developed with Spark Deep-Learning Team at DataBricks

Pipeline API: Inference

Prediction

```
preds = model.setInputMapping({“image”: placeholder_X})
    .setOutputMapping({“prediction”: “col_out”})
    .transform(input_df)
```

Featurizer

```
Features = model.setInputMapping({“image”: placeholder_X})
    .setOutputMapping({“Relu”: “col_out2”})
    .transform(input_df)
```

Pipeline API: Inference from …

TF checkpoints

```
model = model.setModelDir(model_dir)
```

TF saved models: ***using exported signature or not***

```
model = model.setExportDir(export_dir)
    .setTagSet(tag_constants.SERVING)
    .setSignatureDefKey(DEFAULT_SERVING_SIGNATURE_DEF_KEY)
```

Input Modes

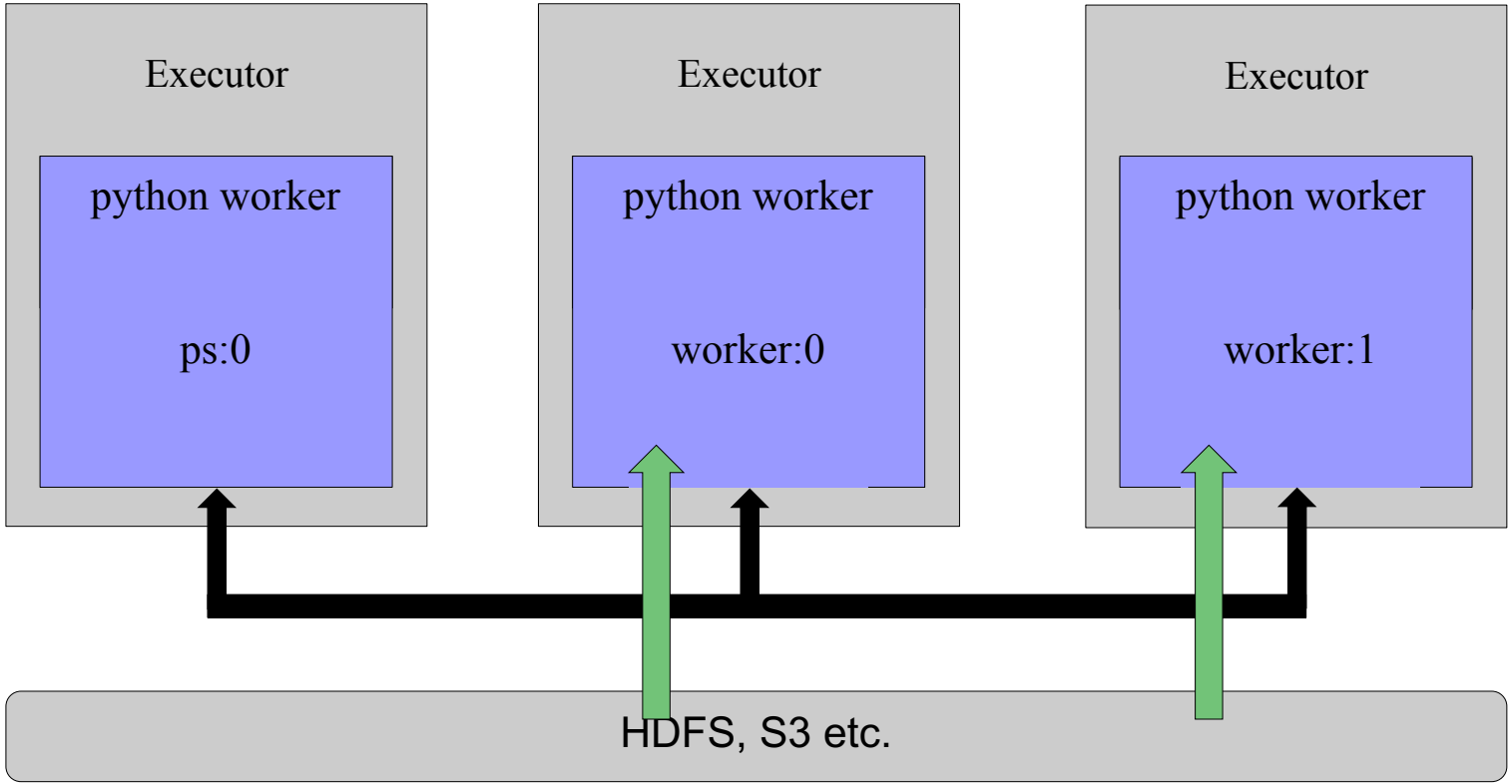
- InputMode.TENSORFLOW

  TFReader + QueueRunner ← HDFS, S3 etc

- InputMode.SPARK

  HDFS → RDD.mapPartitions → feed_dict

InputMode.TENSORFLOW

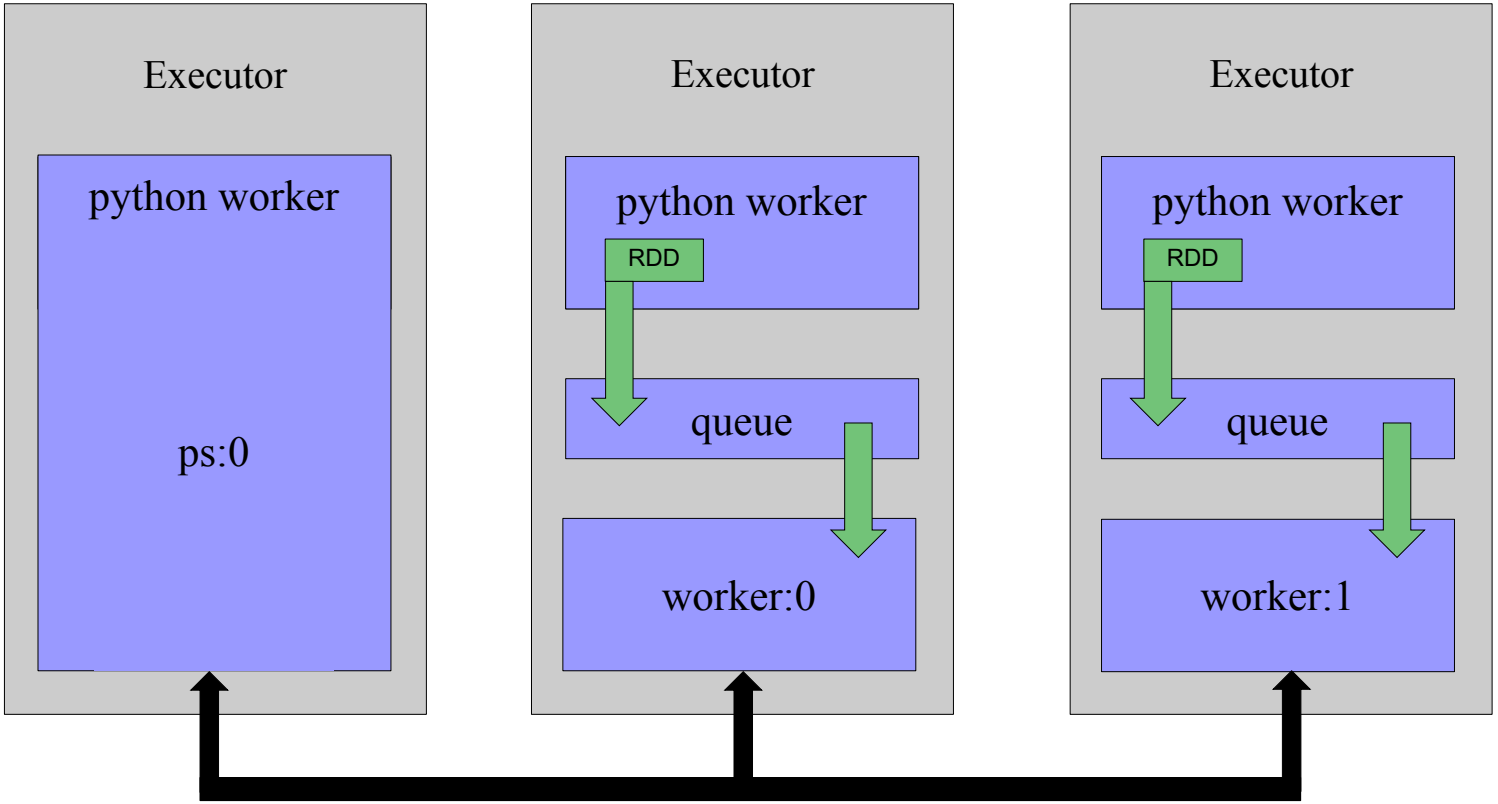


Conversion: Use TensorFlow Input

https://github.com/yahoo/TensorFlowOnSpark/blob/master/examples/mnist/tf/mnist_dist.py

```
def map_fun(args, ctx):
    …
    worker_num = ctx.worker_num
    job_name = ctx.job_name
    task_index = ctx.task_index
    …
    cluster, server = ctx.start_cluster_server(args.num_gpu, args.rdma)
    …
```

InputMode.SPARK



Conversion: Use Spark Input

https://github.com/yahoo/TensorFlowOnSpark/blob/master/examples/mnist/spark/mnist_dist.py

```
def map_fun(args, ctx):
  ...
    worker_num = ctx.worker_num
    job_name = ctx.job_name
    task_index = ctx.task_index
    cluster_spec = ctx.cluster_spec
  ...
    cluster, server = ctx.start_cluster_server(args.num_gpus, args.rdma)
  ...
    tf_feed = ctx.get_data_feed(args.mode == "train")
    while ...:
        batch_xs, batch_ys = my_data_conversion(tf_feed.next_batch(batch_size))
        feed = {x: batch_xs, y_: batch_ys}
        ...
        labels, preds, acc = sess.run([label, prediction, accuracy], feed_dict=feed)
```

Failure Recovery

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

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

Thanks!

And our open-source contributors!

Questions?

https://github.com/yahoo/TensorFlowOnSpark