Distributed deep learning and why you may not need it

Jakub Sanojca, Mikuláš Zelinka

PyData Warsaw 2018

About us

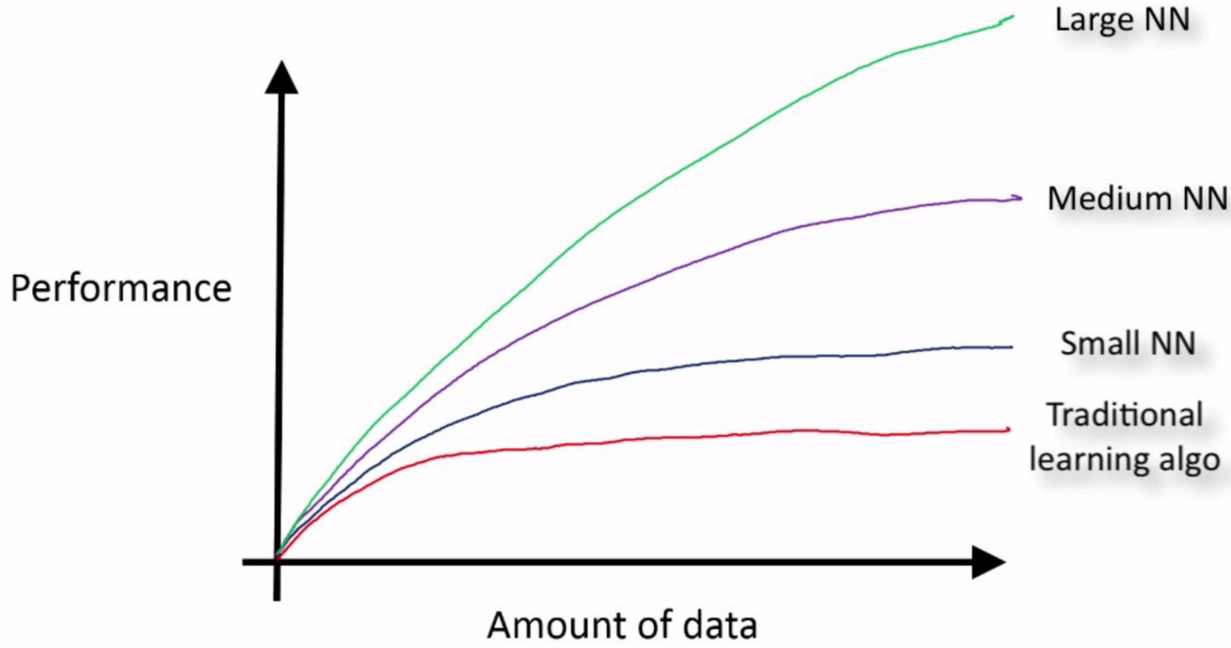
- Research at Avast
  - 400M customers (Avast, AVG, HideMyAss, CCleaner)
  - Tons of ~~damage~~ data

avast

Outline

- Wow such Deep Learning much AI
- Different kinds of distributed learning and parallelism
- Frameworks and approaches

One picture explaining the rise of Deep Learning

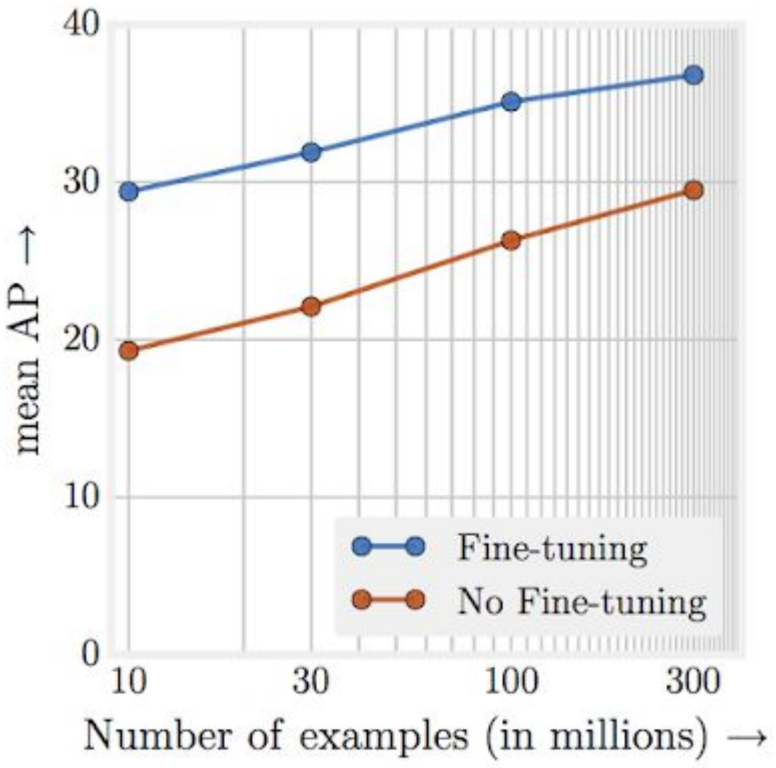


Andrew Ng

Revisiting Unreasonable Effectiveness of Data in Deep Learning Era

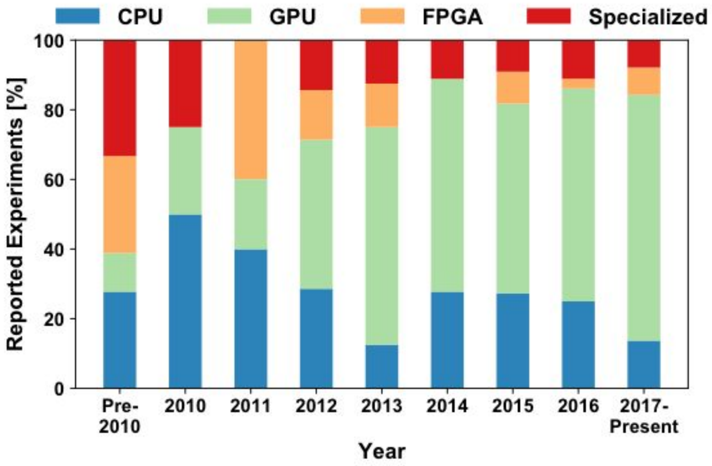
Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta

- COCO object detection:

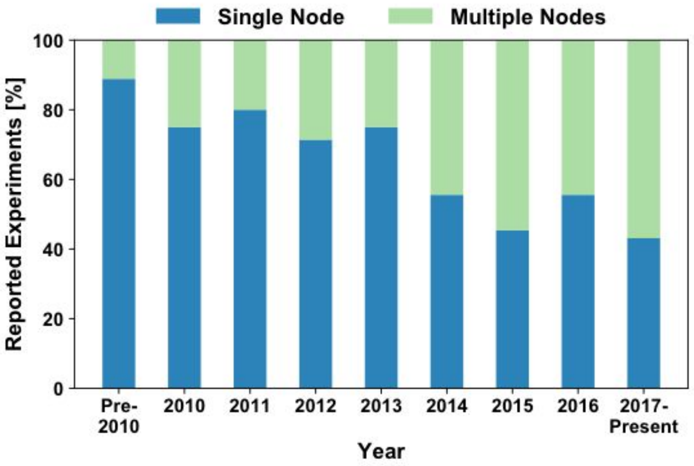


Why go distributed?

Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis 1:7



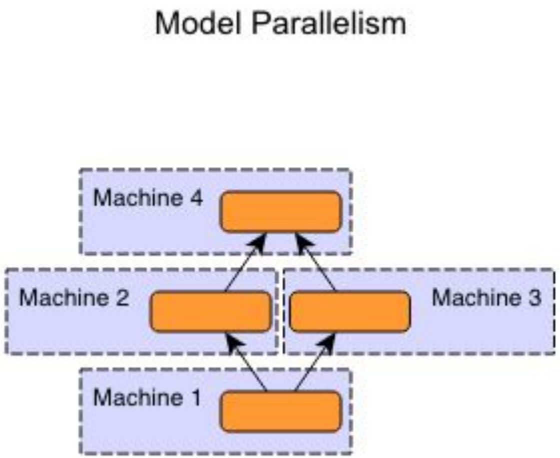
(a) Hardware Architectures

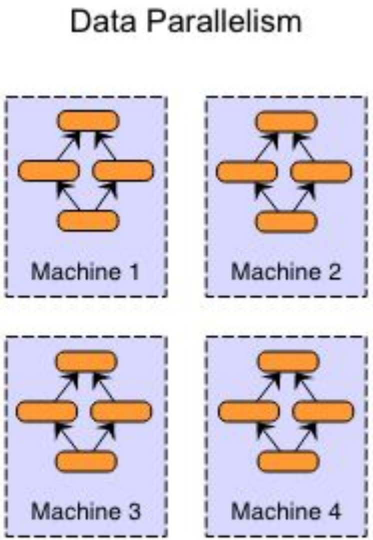


(b) Training with Single vs. Multiple Nodes

avast

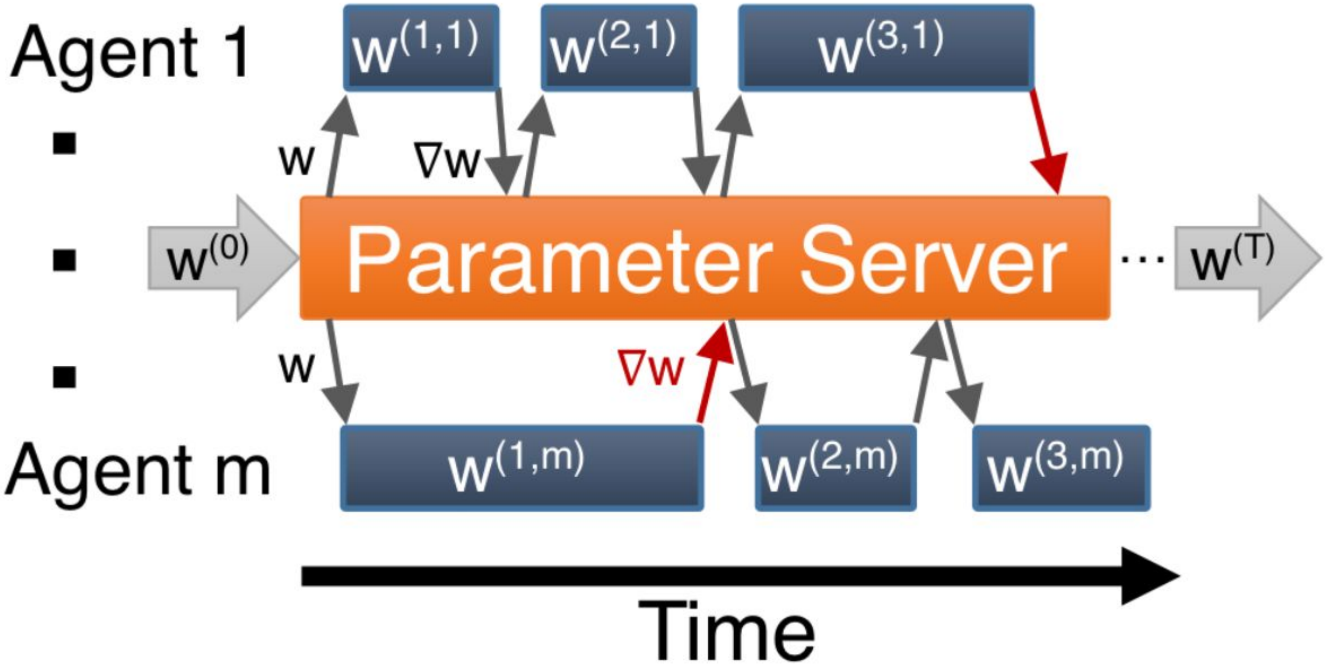
Types of parallelism





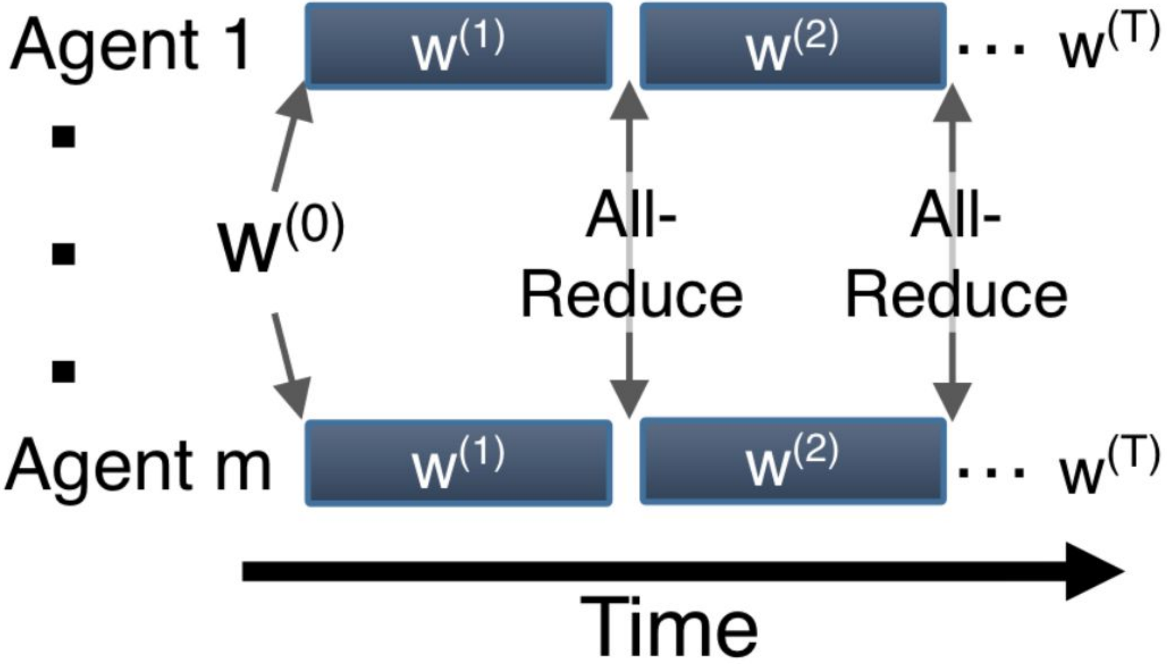
https://blog.skymind.ai/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks/

Asynchronous training



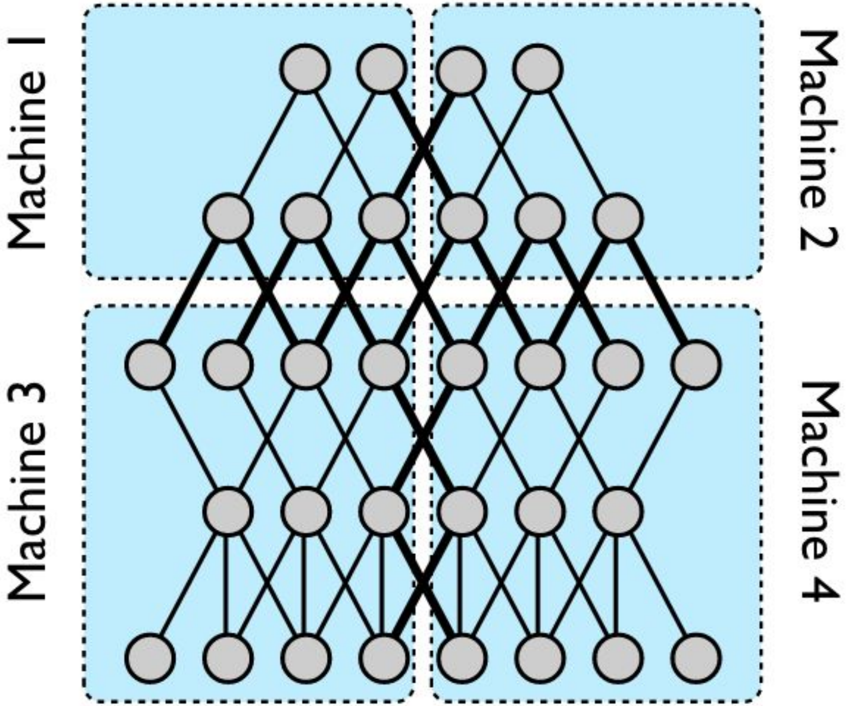
Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. Tal Ben Nun, Torsten Hoefler

Synchronous training with allreduce

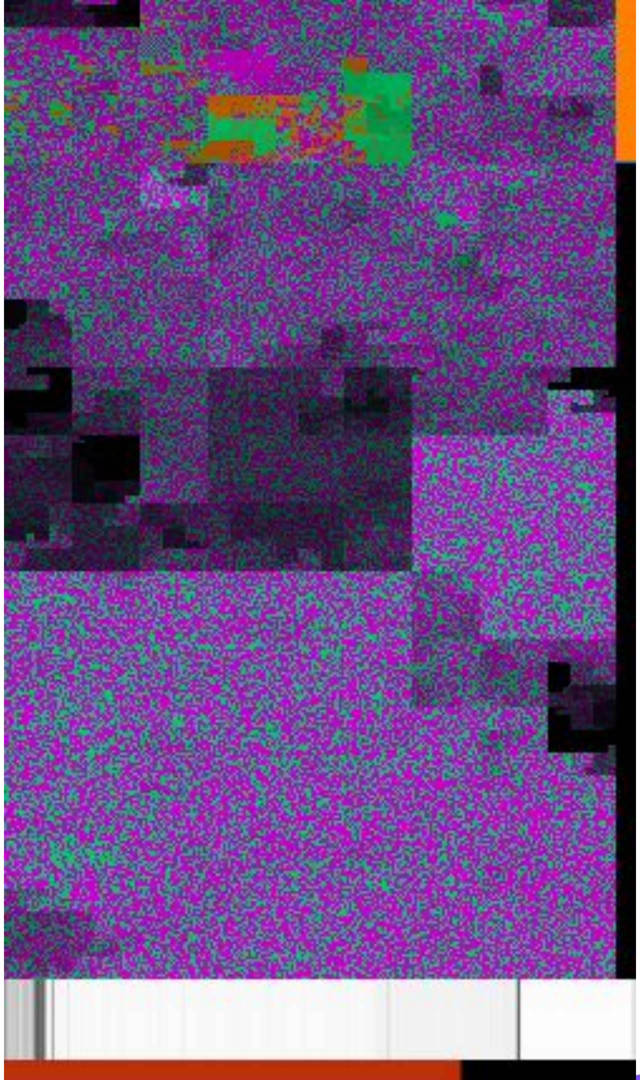


Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis. Tal Ben Nun, Torsten Hoefler

Model parallelism



Large Scale Distributed Deep Networks. Jeff Dean et al.



Distributed Tensorflow (asynchronous)

```
# Create a cluster from the parameter server and worker hosts.
cluster = tf.train.ClusterSpec({"ps": ps_hosts, "worker": worker_hosts})

# Create and start a server for the local task.
server = tf.train.Server(cluster,
                         job_name=FLAGS.job_name,
                         task_index=FLAGS.task_index)

if FLAGS.job_name == "ps":
    server.join()
    elif FLAGS.job_name == "worker":

        # Assigns ops to the local worker by default.
        with tf.device(tf.train.replica_device_setter(
            worker_device="/job:worker/task:%d" % FLAGS.task_index,
            cluster=cluster)):

            # Build model...

# The MonitoredTrainingSession takes care of session initialization,
# restoring from a checkpoint, saving to a checkpoint, and closing when done
# or an error occurs.
with tf.train.MonitoredTrainingSession(master=server.target,
                                       is_chief=(FLAGS.task_index == 0),
                                       checkpoint_dir="/tmp/train_logs",
                                       hooks=hooks) as mon_sess:
    # Train your model
```

avast

Tensorflow

```
run_config = tf.estimator.RunConfig()

classifier = tf.estimator.Estimator(
    model_fn=model_function, # desctiption of your model - loss, updates etc.
    model_dir=model_dir, # directory where you want to store your model
    config=run_config) # how often you need summaries, how often to checkpoint etc.

classifier.train(input_fn=input_function)
```

Distributed Tensorflow (synchronous, single node)

```
distribution = tf.contrib.distribute.MirroredStrategy() # multi-GPU distribution

run_config = tf.estimator.RunConfig(train_distribute=distribution)

classifier = tf.estimator.Estimator(
    model_fn=model_function, # desctiption of your model - loss, updates etc.
    model_dir=model_dir, # directory where you want to store your model
    config=run_config) # how often you need summaries, how often to checkpoint etc.

classifier.train(input_fn=input_function)
```

Horovod

```
import tensorflow as tf
import horovod.tensorflow as hvd

# Initialize Horovod
hvd.init()

# Pin GPU to be used to process local rank (one GPU per process)
config = tf.ConfigProto()
config.gpu_options.visible_device_list = str(hvd.local_rank())

# Build model...

# Add Horovod Distributed Optimizer
opt = hvd.DistributedOptimizer(opt)

# The MonitoredTrainingSession takes care of session initialization,
# restoring from a checkpoint, saving to a checkpoint, and closing when done
# or an error occurs.
with tf.train.MonitoredTrainingSession(checkpoint_dir=checkpoint_dir,
                                       config=config,
                                       hooks=hooks) as mon_sess:
    # Perform synchronous training.
```

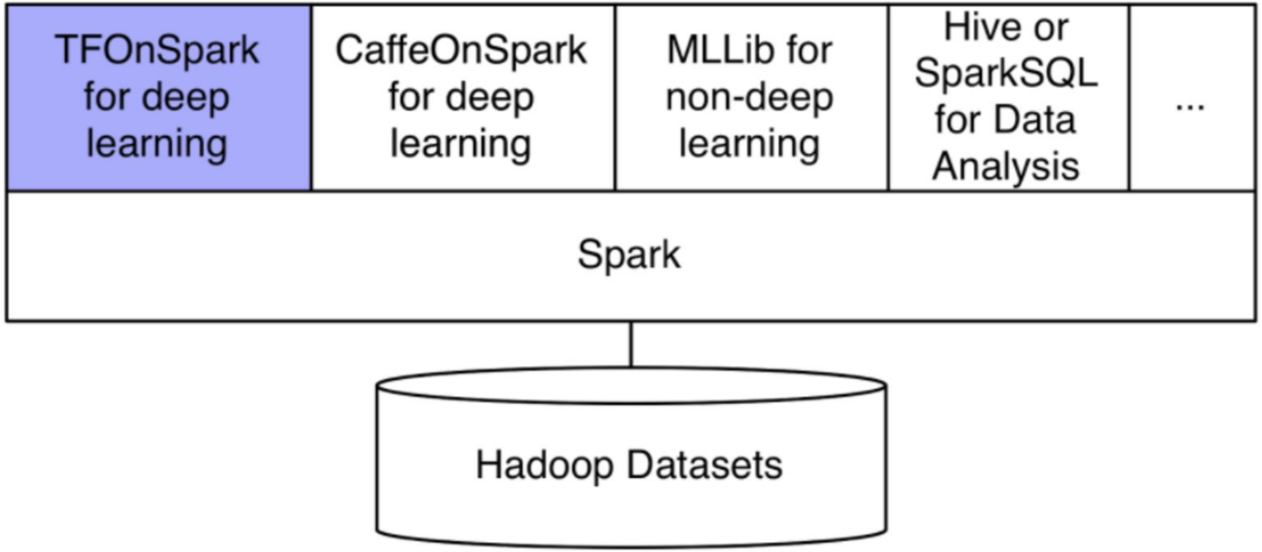
https://github.com/uber/horovod

Horovod

```python
import tensorflow as tf
import horovod.tensorflow as hvd

# Initialize Horovod
hvd.init()

# Pin GPU to be used to process local rank (one GPU per process)
config = tf.ConfigProto()
config.gpu_options.visible_device_list = str(hvd.local_rank())

# Build model...

# Add Horovod Distributed Optimizer
opt = hvd.DistributedOptimizer(opt)

# The MonitoredTrainingSession takes care of session initialization,
# restoring from a checkpoint, saving to a checkpoint, and closing when done
# or an error occurs.
with tf.train.MonitoredTrainingSession(checkpoint_dir=checkpoint_dir,
                                       config=config,
                                       hooks=hooks) as mon_sess:
    # Perform synchronous training.
```

```
$ mpirun -np 16 \
    -H server1:4,server2:4,server3:4,server4:4 \
    -bind-to none -map-by slot \
    -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH \
    -mca pml ob1 -mca btl ^openib \
    python train.py
```

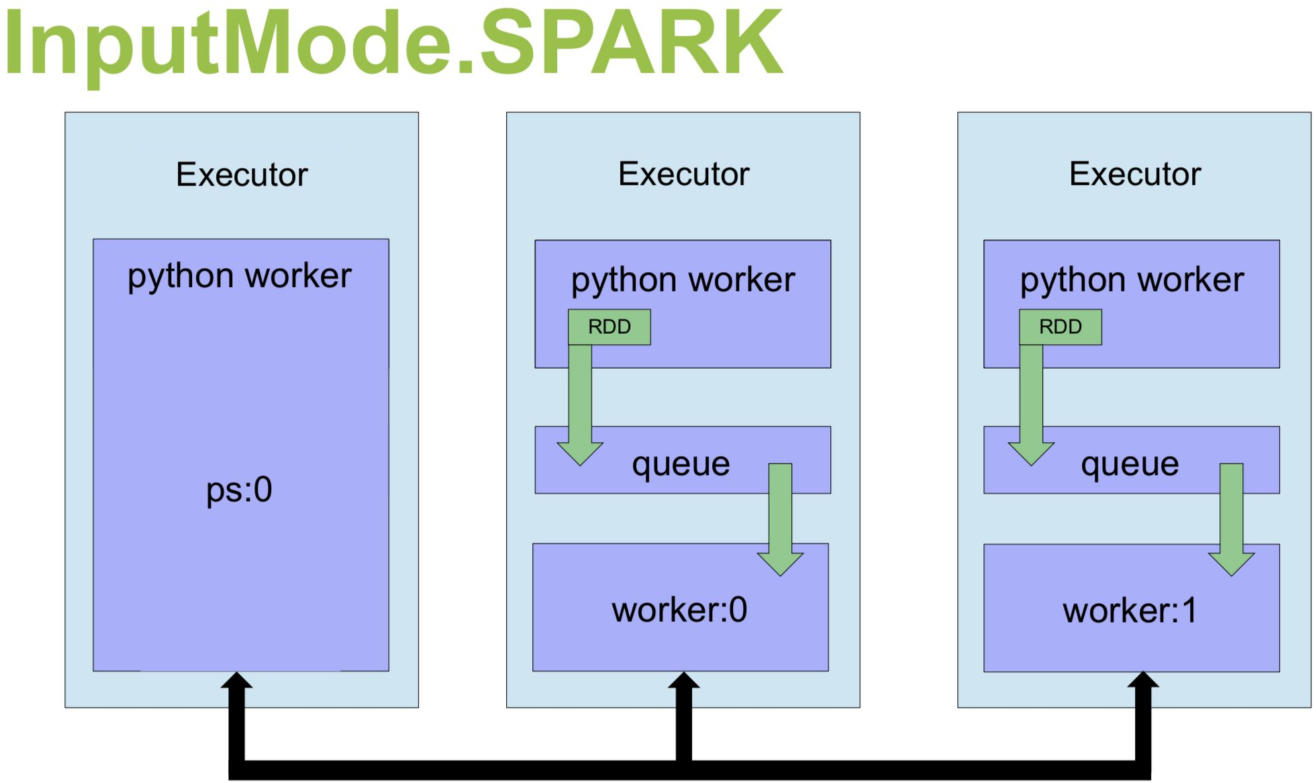
Tensorflow on Spark



https://github.com/yahoo/TensorFlowOnSpark

TensorFlowOnSpark: Scalable Tensorflow Learning on Spark Clusters. Lee Yang, Andrew Feng, Yahoo Big Data ML Platform

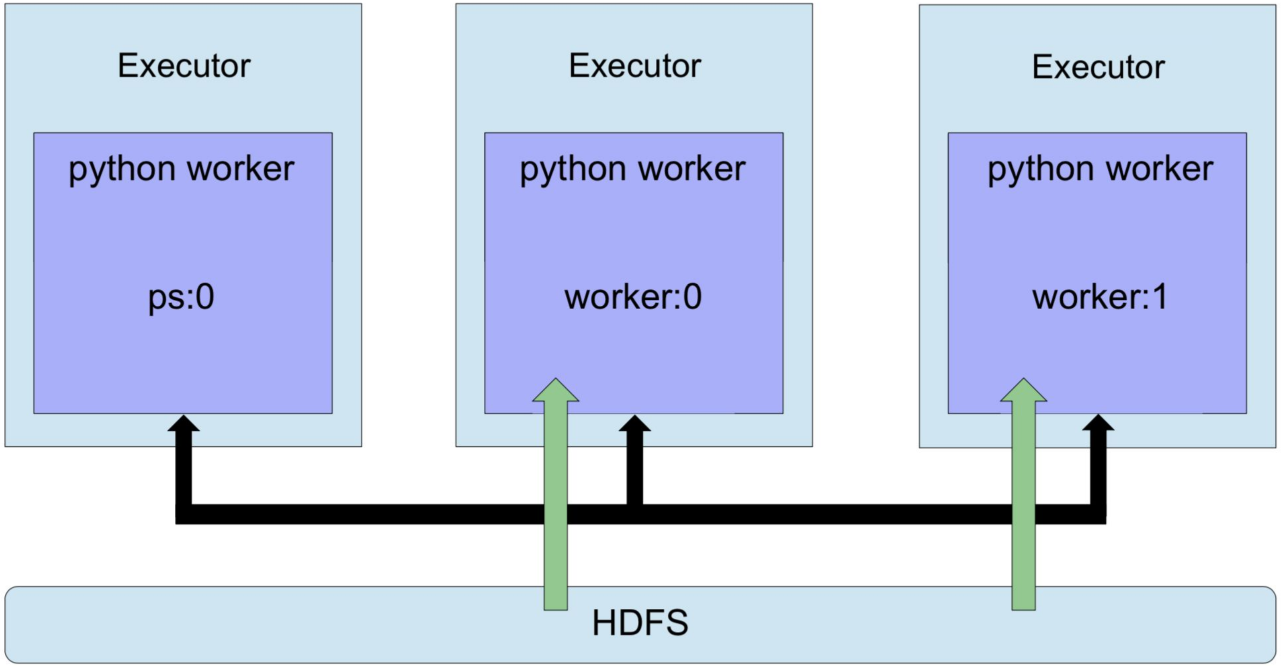
Tensorflow on Spark



TensorFlowOnSpark: Scalable Tensorflow Learning on Spark Clusters. Lee Yang, Andrew Feng, Yahoo Big Data ML Platform

Tensorflow on Spark

InputMode.TENSORFLOW



Tensorflow on Spark

```
from tensorflowonspark import TFCluster, TFNode

# start the cluster
cluster = TFCluster.run(sc, map_fn, args, num_executors, num_ps, tensorboard, input_mode)

# either pass data for training
cluster.train(dataRDD, num_epochs=0)

# or pass data for scoring
cluster.inference(dataRDD)

# shutdown the cluster
cluster.shutdown()
```

Tensorflow on Spark

```
def map_fun(args, ctx):

    import tensorflow as tf

    # necessary variables
    logdir = ctx.absolute_path(args.model)
    hooks = [tf.train.StopAtStepHook(last_step=100000)]
    batch_size = args.batch_size

    worker_num = ctx.worker_num
    job_name = ctx.job_name
    task_index = ctx.task_index

    # Get TF cluster and server instances
    cluster, server = ctx.start_cluster_server(1, args.rdma)

    if job_name == "ps":
        print('PARAMETER SERVER')
        server.join()
    elif job_name == "worker":

        # Assigns ops to workers - taken care of by Spark
        with tf.device(tf.train.replica_device_setter(
                worker_device="/job:worker/task:%d" % task_index,
                cluster=cluster)):

            [...]

    # start your MonitoredTrainingSession
    with tf.train.MonitoredTrainingSession(master=server.target,
                                           is_chief=(task_index == 0),
                                           checkpoint_dir=logdir,
                                           hooks=hooks) as mon_sess:

        # train or score your data
        [...]
```

Comparison and our benchmarks

synchronous Tensorflow	asynchronous Tensorflow	Horovod	TFonS
easiest to use	not hard to use, but requires code change	requires code change	requires most code change
only one machine	multi-device, multi-machine	multi-device, multi-machine	multi-device, multi-machine
speed scales almost linearily	scales almost linearity	slower than expected, most probably because of higher network requirements	slowest because of a lot of overhead, can be run on existing infrastructure

Why you may not need it

- Hardware improvements
- Code simplicity
- Simpler models are enough / data efficiency
- Experimentation

Thank you!

For questions, feel free to reach out to us at:

jakub.sanojca@avast.com
mikulas.zelinka@avast.com

Resources:

- https://arxiv.org/pdf/1802.09941.pdf / https://www.youtube.com/watch?v=xtxxLWZznBI
- https://arxiv.org/pdf/1610.02527.pdf
- https://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.pdf
- https://medium.com/@esaliya/model-parallelism-in-deep-learning-is-not-what-you-think-94d2f81e82ed
- https://blog.skymind.ai/distributed-deep-learning-part-1-an-introduction-to-distributed-training-of-neural-networks/
- https://www.youtube.com/watch?v=SphfeTl70MI
- https://www.tensorflow.org/deploy/distributed
- https://arxiv.org/pdf/1802.05799.pdf / https://github.com/uber/horovod
- https://github.com/yahoo/TensorFlowOnSpark
- https://github.com/tensorflow/ecosystem/tree/master/spark/spark-tensorflow-connector

avast