What is Analytics Zoo

BigDL

Distributed, High-Performance

Deep Learning Framework

for Apache Spark

https://github.com/intel-analytics/bigdl

ANALYTICS ZOO

Unified Analytics + AI Platform

Distributed TensorFlow, Keras, PyTorch and BigDL on Apache Spark

https://github.com/intel-analytics/analytics-zoo

Accelerating Data Analytics + AI Solutions At Scale

TFPark：TensorFlow in Production on Spark

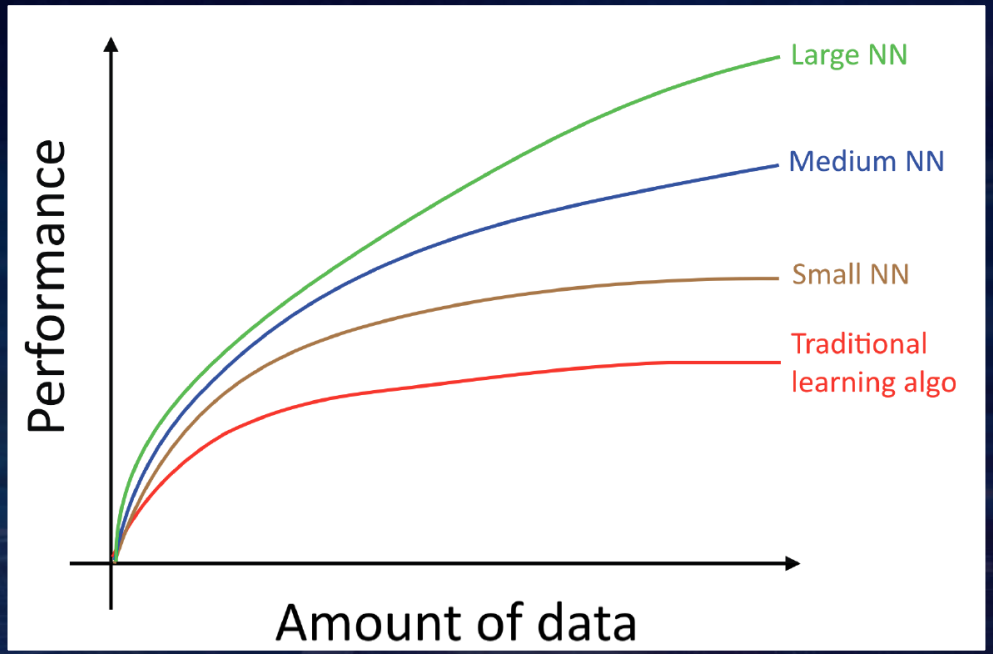
Wang, Yang (yang3.wang@intel.com)
Machine Learning Engineer at Intel

TensorFlow

- One of the most popular open source DL frameworks
- Large community, many published model
- Works on many devices

Why Running TensorFlow on Spark

Data Scale Driving Deep Learning Process



Machine Learning System Complexity

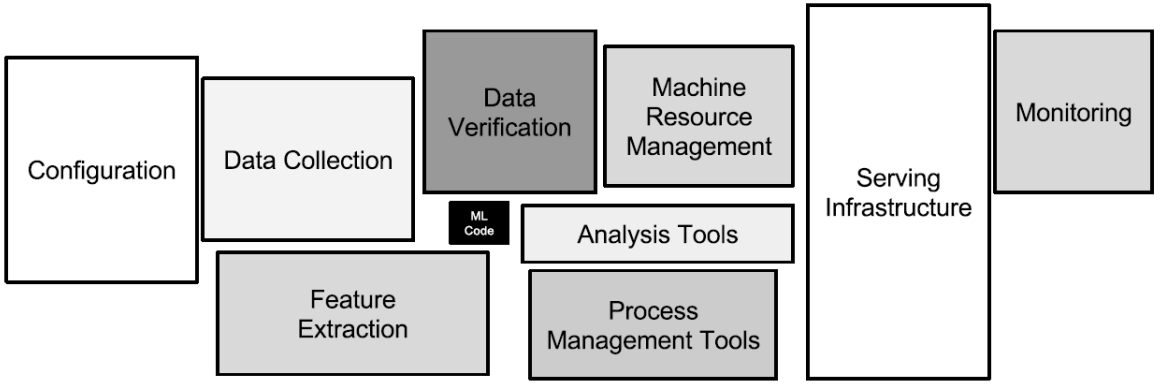
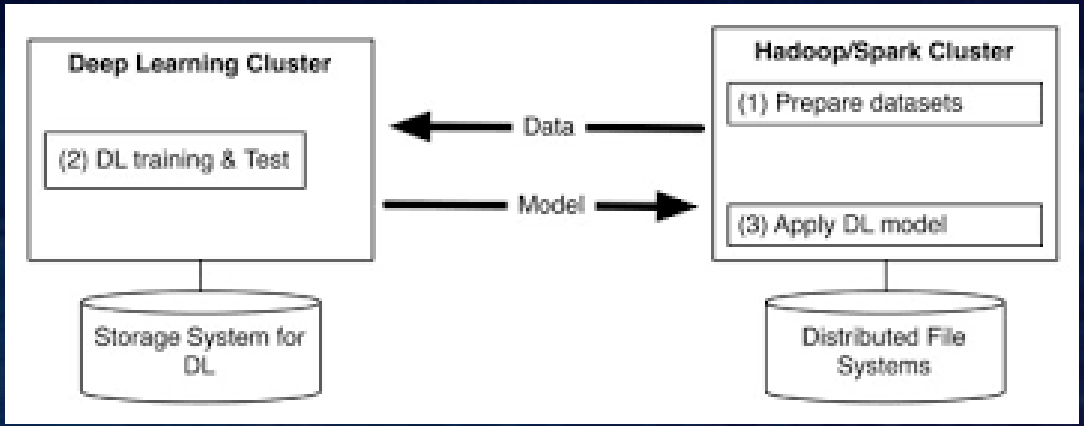


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

Problems of two clusters



TFPark Package in Analytics Zoo

Running TensorFlow model on Spark with minimal code changes at scale

Analytics Zoo: End-to-End DL Pipeline Made Easy for Big Data

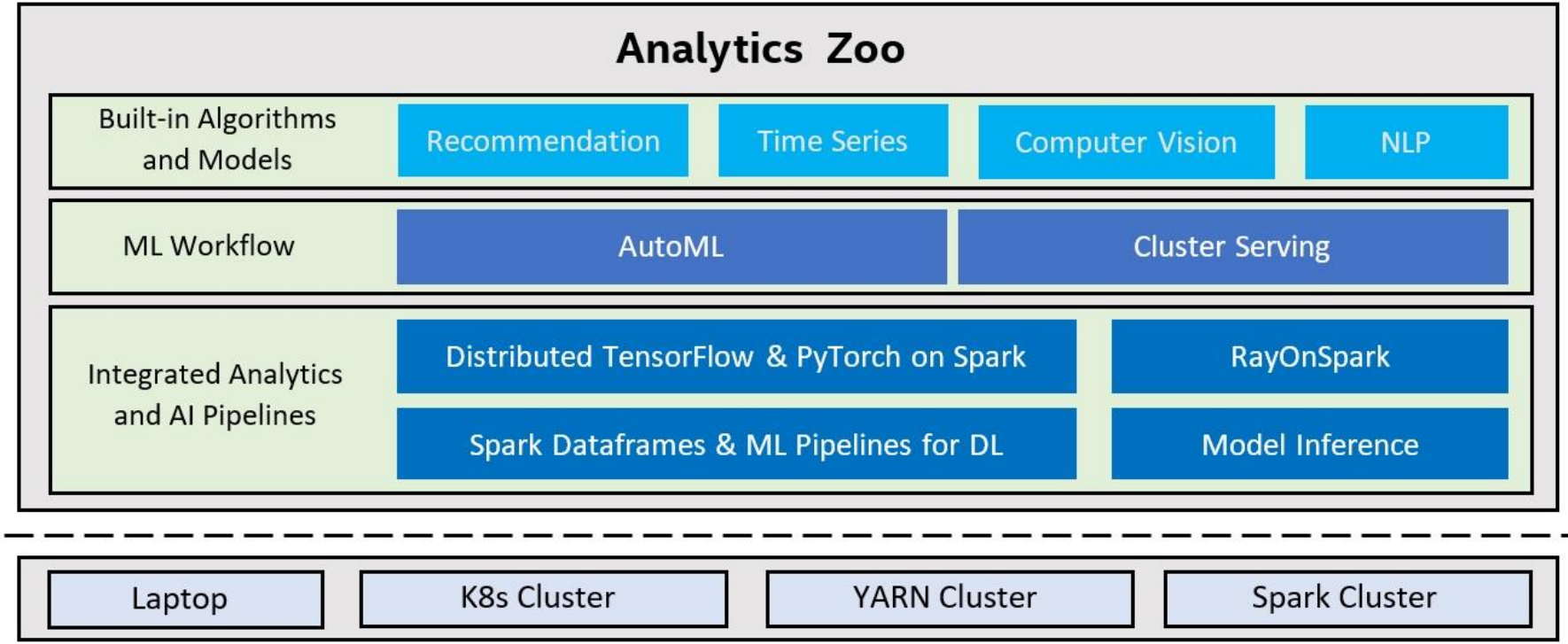
Prototype on laptop using sample data

Experiment on clusters with history data

Deployment with production, distributed big data pipelines

- **"Zero" code change from laptop to distributed cluster**
- **Directly access production data (Hadoop/Hive/HBase) without data copy**
- **Easily prototype the end-to-end pipeline**
- **Seamlessly deployed on production big data clusters**

Analytics Zoo



https://github.com/intel-analytics/analytics-zoo

TFPark Package in Analytics Zoo

- Seamless Integrate TensorFlow with Spark & BigDL
- Better performance with custom built TensorFlow with MKL
- Easy to use distributed API, little changes required for your TF models
- “Zero” code change from laptop to cluster

Comparison with Distributed TF Solutions

Local TensorFLow（v1.x）

```
import tensorflow as tf
import tensorflow_datasets as tfds

global_batch_size = 256
global_step = tf.train.get_or_create_global_step()

# Dataset
ds = tfds.load("mnist", split="train").map(map_func)
dataset = ds.repeat().shuffle(1000).batch(batch_size)
iter = dataset.make_one_shot_iterator()
images, labels = iter.get_next()

# Model
loss = create_model(images, labels)

optimizer = tf.train.GradientDescentOptimizer(.0001)
opt = optimizer.minimize(loss, global_step=global_step)

# Session
stop_hook = tf.train.StopAtStepHook(last_step=2000)
hooks = [stop_hook]
sess = tf.train.MonitoredTrainingSession(checkpoint_dir='.\checkpoint_dir', hooks=hooks)
while not sess.should_stop():
    _, r, gs = sess.run([opt, loss, global_step])
    print("step {}, loss {}".format(gs, r))
```

Built-in Distributed TF (v1)

```
import tensorflow as tf
import tensorflow_datasets as tfds

ps_hosts = FLAGS.ps_hosts.split(",")
worker_hosts = FLAGS.worker_hosts.split(",")
num_task = len(worker_hosts)
global_batch_size = 256

# Cluster
cluster = tf.train.ClusterSpec({"ps": ps_hosts, "worker": worker_hosts})
if FLAGS.job_name == 'ps':  # checks if parameter server
    server = tf.train.Server(cluster, job_name="ps", task_index=FLAGS.task_index)
    server.join()
else:  # it must be a worker server
    is_chief = (FLAGS.task_index == 0)  # checks if this is the chief node
    server = tf.train.Server(cluster, job_name="worker", task_index=FLAGS.task_index)

# Dataset
ds = tfds.load("mnist", split="train").map(map_func)
dataset = ds.shard(num_task, FLAGS.task_index).
            repeat().shuffle(1000).
            batch(global_batch_size // num_task)
images, labels = dataset.make_one_shot_iterator().get_next()
```

```
# Model
worker_device = "/job:%s/task:%d/cpu:0" % (FLAGS.job_name, FLAGS.task_index)
with tf.device(tf.train.replica_device_setter(ps_tasks=1, worker_device=worker_device)):

    global_step = tf.train.get_or_create_global_step()
    loss = create_model(images, labels)
    optimizer = tf.train.GradientDescentOptimizer(.0001)
    optimizer = tf.train.SyncReplicasOptimizer(optimizer,
                        replicas_to_aggregate=num_task, total_num_replicas=num_task)
    opt = optimizer.minimize(loss, global_step=global_step)

# Session
sync_replicas_hook = optimizer.make_session_run_hook(is_chief)
stop_hook = tf.train.StopAtStepHook(last_step=2000)
hooks = [sync_replicas_hook, stop_hook]

checkpoint_dir = './checkpoints' if is_chief else None
sess = tf.train.MonitoredTrainingSession(checkpoint_dir=checkpoint_dir,
master=server.target,
          is_chief=is_chief, hooks=hooks,  stop_grace_period_secs=10)

while not sess.should_stop():
    _, r, gs = sess.run([opt, loss, global_step])
```

Horovod

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

hvd.init()

global_batch_size = 256
global_step = tf.train.get_or_create_global_step()

ds = tfds.load("mnist", split="train").map(map_func)
dataset = ds.shard(hvd.size(), hvd.rank()).
            repeat().shuffle(1000).
            batch(global_batch_size // num_task)
images, labels = dataset.make_one_shot_iterator().get_next()
loss = create_model(images, labels)

optimizer = tf.train.GradientDescentOptimizer(.0001)
optimizer = hvd.DistributedOptimizer(optimizer, op=hvd.Average)
opt = optimizer.minimize(loss, global_step=global_step)

hooks = [
    hvd.BroadcastGlobalVariablesHook(0),
    tf.train.StopAtStepHook(last_step=2000 // hvd.size()),
]

checkpoint_dir = './checkpoints' if hvd.rank() == 0 else None
with tf.train.MonitoredTrainingSession(checkpoint_dir=checkpoint_dir,hooks=hooks) as mon_sess:
    while not mon_sess.should_stop():
        _, r, gs = mon_sess.run([opt, loss, global_step])
```

Analytics Zoo (TFPark)

```
import tensorflow as tf
import tensorflow_datasets as tfds
from zoo.tfpark import TFDataset, TFOptimizer
from zoo import init_nncontext

sc = init_nncontext()

global_batch_size = 256

ds = tfds.load("mnist", split="train").map(map_func)
ds = ds.shuffle(1000)
dataset = TFDataset.from_tf_data_dataset(ds, batch_size=global_batch_size)
images, labels = dataset.tensors

loss = create_model(images, labels)
optimizer = TFOptimizer.from_loss(loss, SGD(1e-3),
                                  model_dir="./checkpoints")
optimizer.optimize(end_trigger=MaxIteration(200000))
```

Code Change Summary

	Modifications to Single Node Program
Built-in Distributed TF	1. Specify *ps_hosts* and *worker_hosts*, and create *tf.train.ClusterSpec object* 2. Customize *tf.train.Server* object for each node 3. Shard data according to worker node *number* and *index* using *tf.data.dataset* 4. Assign operations to devices 5. Replace regular Optimizer with *SyncReplicasOptimizer* 6. Create *tf.train.MonitoredTrainingSession*
Horovod	1. Run *hvd.init()* 2. Shard data according to *hvd.size()* and *hvd.rank()* using *tf.data.dataset* 3. Replace regular optimizer with *hvd.DistributedOptimizer* 4. Add *hvd.BroadcastGlobalVariablesHook(0)* to broad rank 0's variables to other nodes 5. Checking *hdv.rank()* and only make checkpoint and Tensorboard log if *rank* is 0
Analytics Zoo (TFPark)	1. Run *init_nncontext()* 2. Create *TFDataset* from *tf.data.dataset* or *RDD* 3. Create *TFOptimizer* and start training

Deployment

	Installation	Modification to Cluster	Run the program
Built-in Distributed TF	pip install tensorflow	Install on every node	Copy program to every node, set environment variables and start program on each node
Horovod	• pip install horovod • install openMPI	• Install on every node • Driver need to ssh to each node without prompt	Using *horovodrun* command (on the driver machine using ssh to each worker without prompt)
Analytics Zoo (TFPark)	pipe install analytics-zoo	Only install on driver	Using spark-submit command to the underlying Haoop, Spark or K8s cluster

Other TFPark Features

TensorFlow High-Level API （Estimator)

```
linear_est = tf.estimator.LinearClassifier(feature_columns=feature_columns,
                                           optimizer=ZooOptimizer(tf.train.FtrlOptimizer(0.2)),
                                           model_dir="/tmp/estimator/linear")
zoo_est = TFEstimator(linear_est)
train_input_fn = make_input_fn(dftrain, y_train,
                               mode=tf.estimator.ModeKeys.TRAIN,
                               batch_size=32)
zoo_est.train(train_input_fn, steps=200)
```

https://github.com/intel-analytics/analytics-zoo/blob/master/pyzoo/zoo/examples/tensorflow/tfpark/estimator/pre-made-estimator.py

intel Software

TensorFlow High-Level API （Keras）

```
model = create_keras_model()
model.compile(optimizer='rmsprop',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
keras_model = KerasModel(model)
keras_model.fit(x, y,
                epochs=max_epoch,
                batch_size=320,
                distributed=True)
```

https://github.com/intel-analytics/analytics-zoo/blob/master/pyzoo/zoo/examples/tensorflow/tfpark/keras/keras_ndarray.py

GAN Model Support

```
opt = GANEstimator(
    generator_fn=conditional_generator,
    discriminator_fn=conditional_discriminator,
    generator_loss_fn=wasserstein_generator_loss,
    discriminator_loss_fn=wasserstein_discriminator_loss,
    generator_optimizer=ZooOptimizer(tf.train.AdamOptimizer(1e-5, 0.5)),
    discriminator_optimizer=ZooOptimizer(tf.train.AdamOptimizer(1e-4, 0.5)),
    model_dir=MODEL_DIR,
    session_config=tf.ConfigProto()
)

for i in range(20):
    opt.train(input_fn, MaxIteration(1000))
    eval()
```

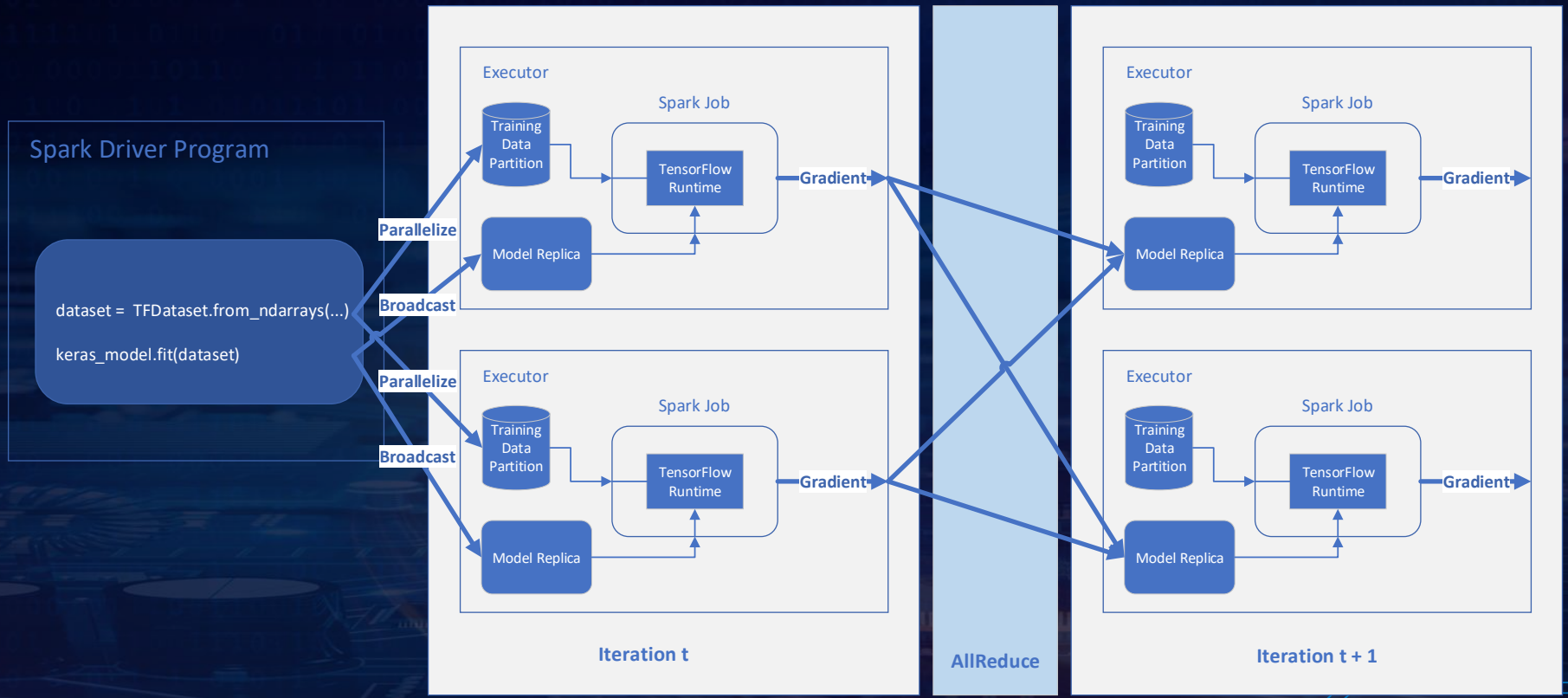
https://github.com/intel-analytics/analytics-zoo/tree/master/pyzoo/zoo/examples/tensorflow/tfpark/gan

Connecting to Various Data Sources (TFDataset)

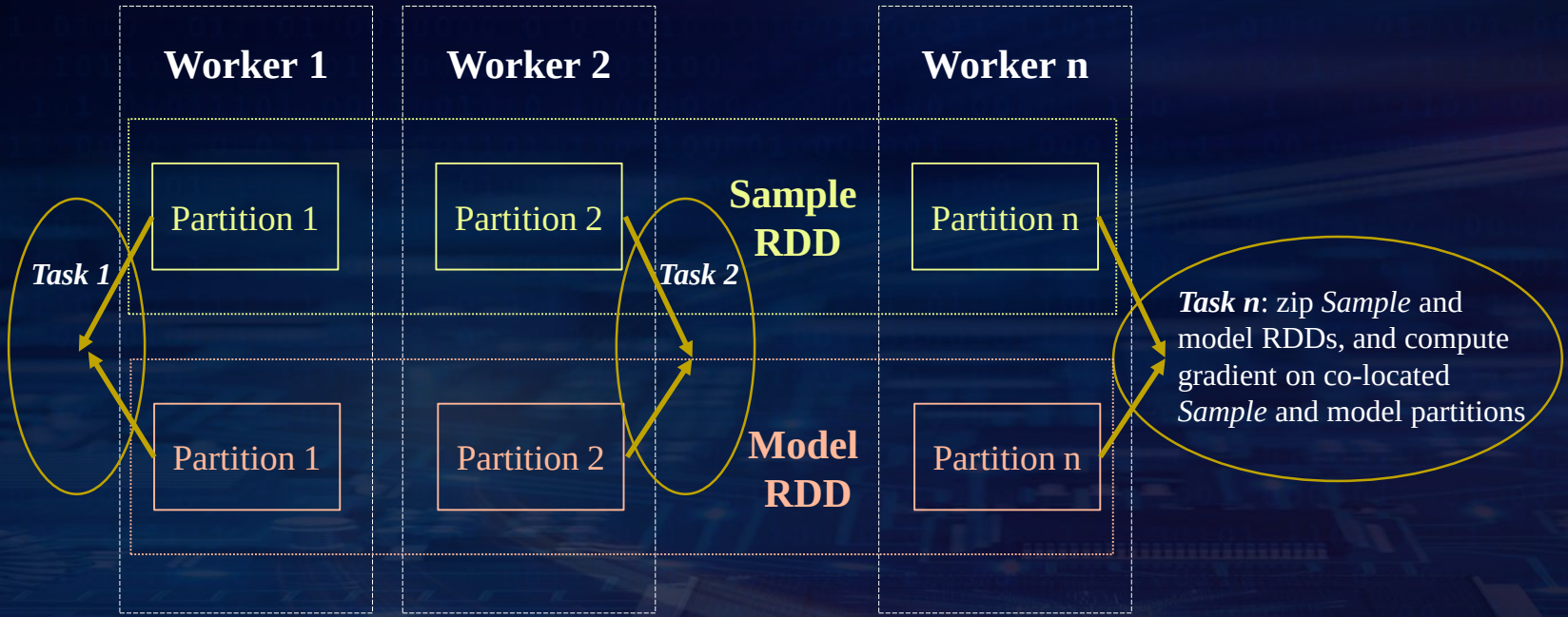
- Driver in memory data, such as ***numpy.ndarray***
- Pipeline written as ***tensorflow.data.Dataset***
- File data such as text file or image files on both local file system/ hdfs/ s3
- Data represented as ***Spark RDDs*** or ***Spark Dataframes***
- Any other data sources that can be read using Spark, such as Hbase,Hive, Kafka, or Relational Database

TFPark Implementation

Architecture Overview

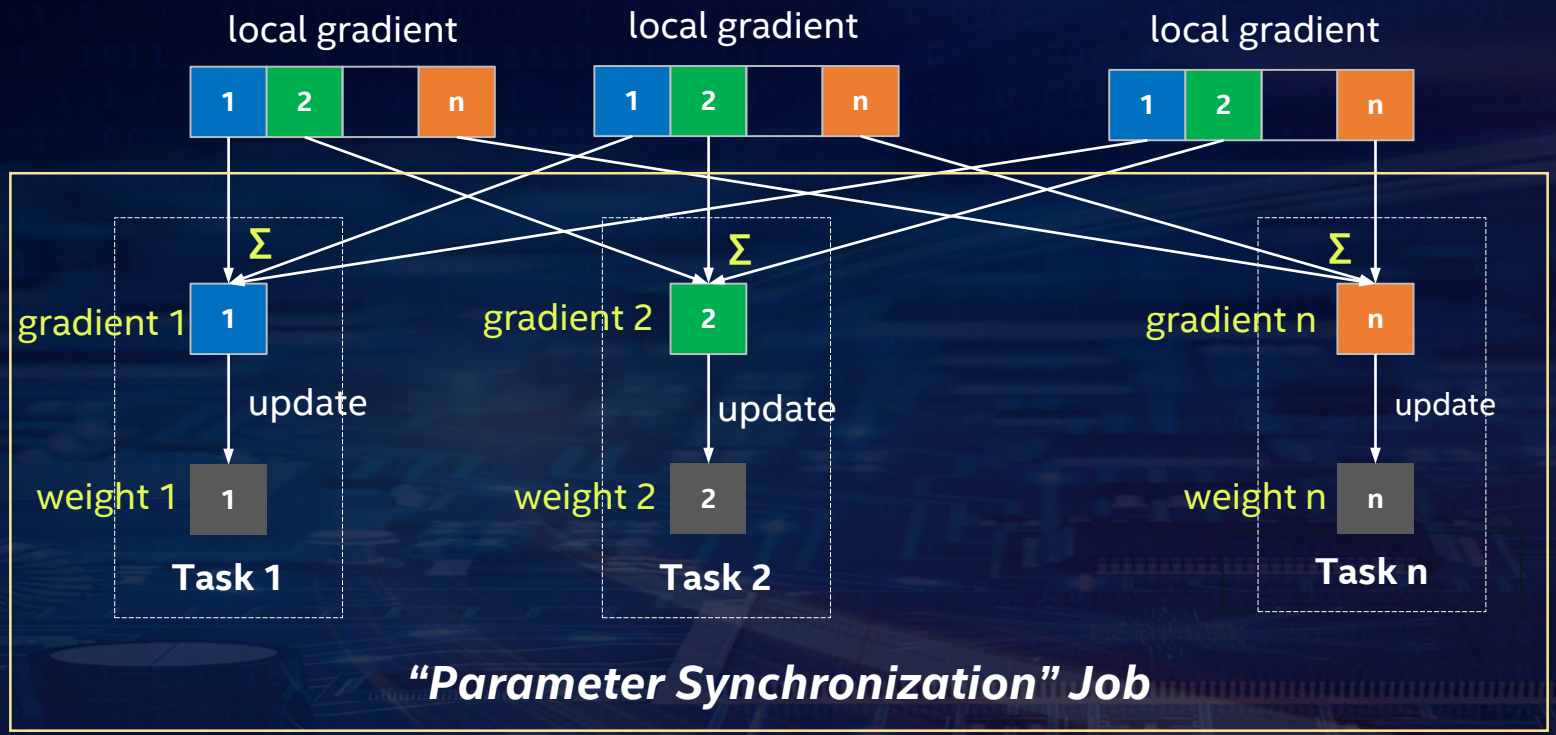


Synchronous Data Parallelism



"Model Forward-Backward" Job

All-Reduce Parameter Synchronization



For each Iteration

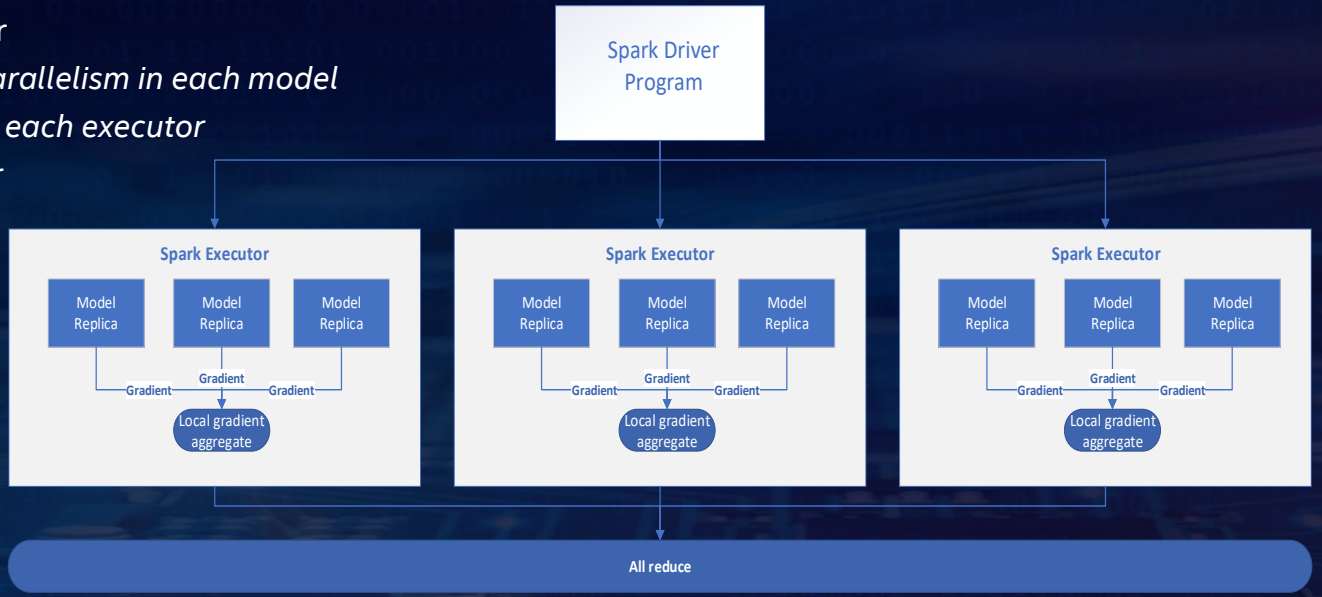
- Job 1: every partition input data and label, output loss and gradients, using TensorFlow

- Job 2: all-reduce parameter sync, using BigDL

Optimize CPU Utilization

Tunable parameter

- *Inter/inter_op_parallelism in each model*
- *Model number in each executor*
- *Executor Number*



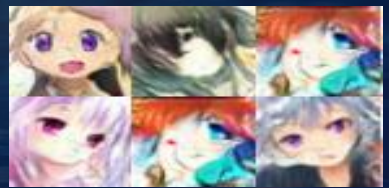
Use Case

Cartoon Face Generation (SJTU Course Project)

Epoch 1

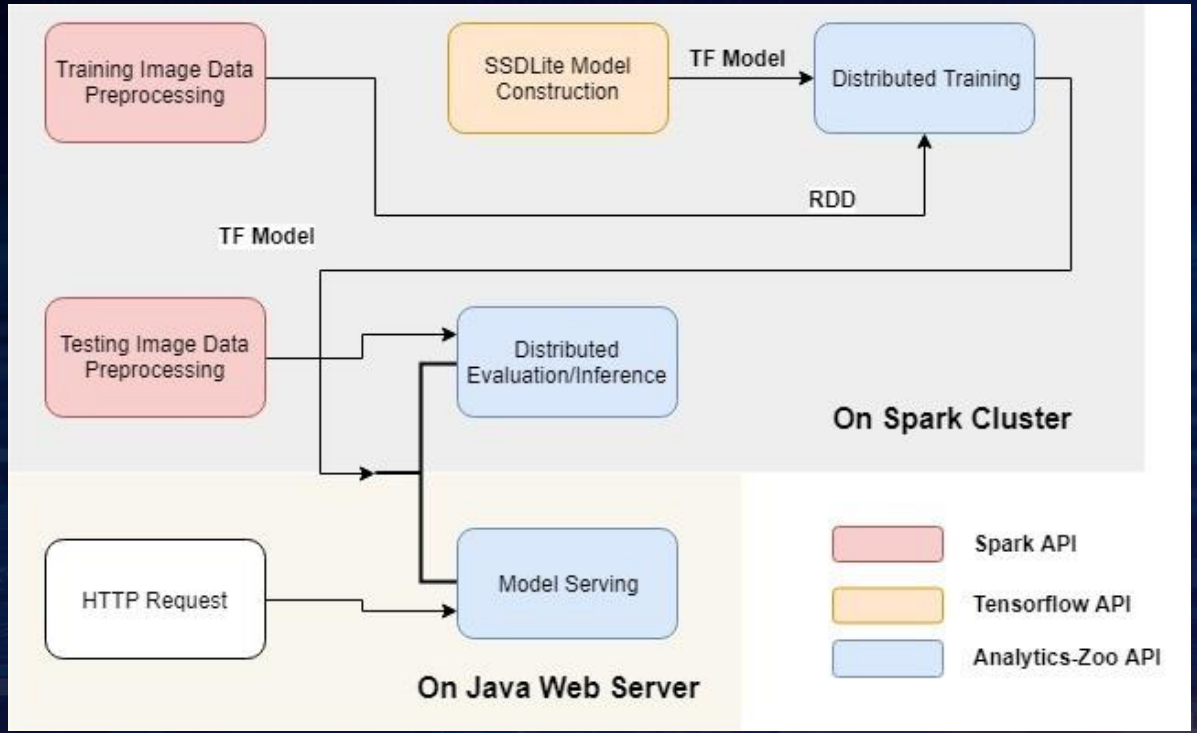
Epoch 5

Epoch 10



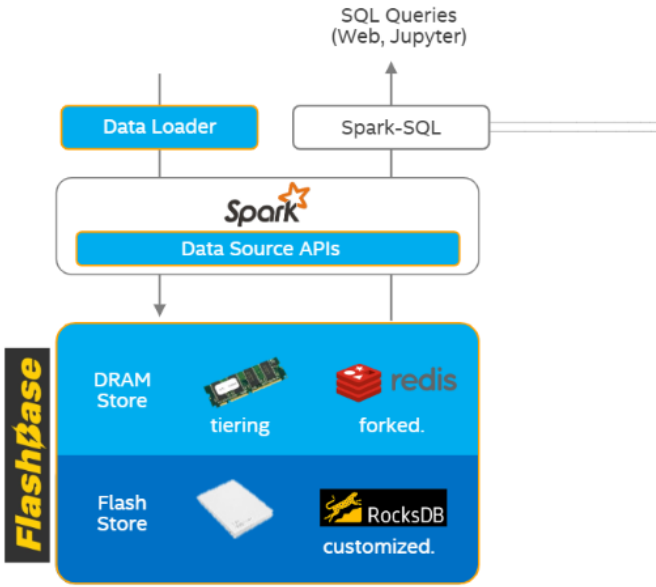
Epoch 300

Computer Vision Based Product Defect Detection in Midea



https://software.intel.com/en-us/articles/industrial-inspection-platform-in-midea-and-kuka-using-distributed-tensorflow-on-analytics

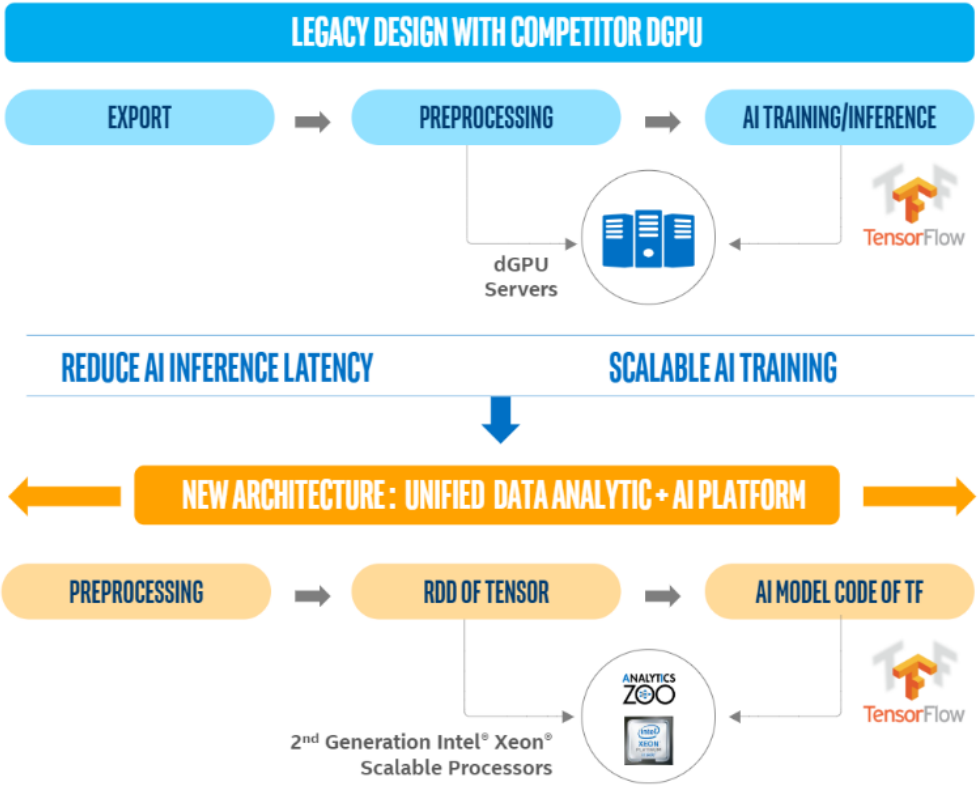
Time Series Forecasting in SKT



SK telecom

Lightning DB

*Other names and brands may be claimed as the property of others.

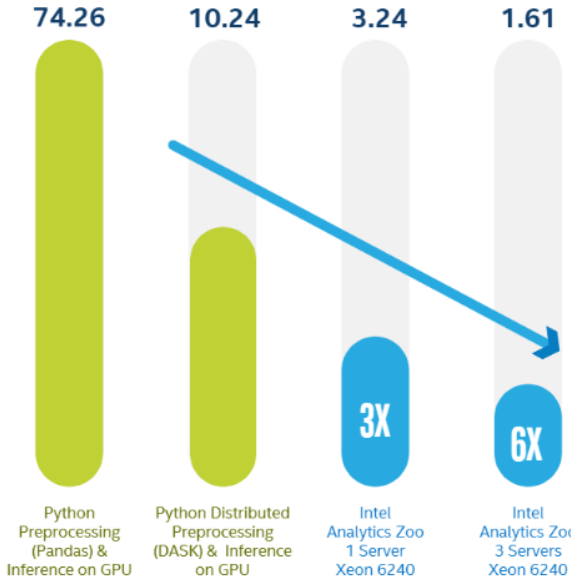


https://github.com/analytics-zoo/analytics-zoo.github.io/blob/master/presentations/0326%20Webinar%20SKT_AI%20Network%20Analytics_FINAL.pdf

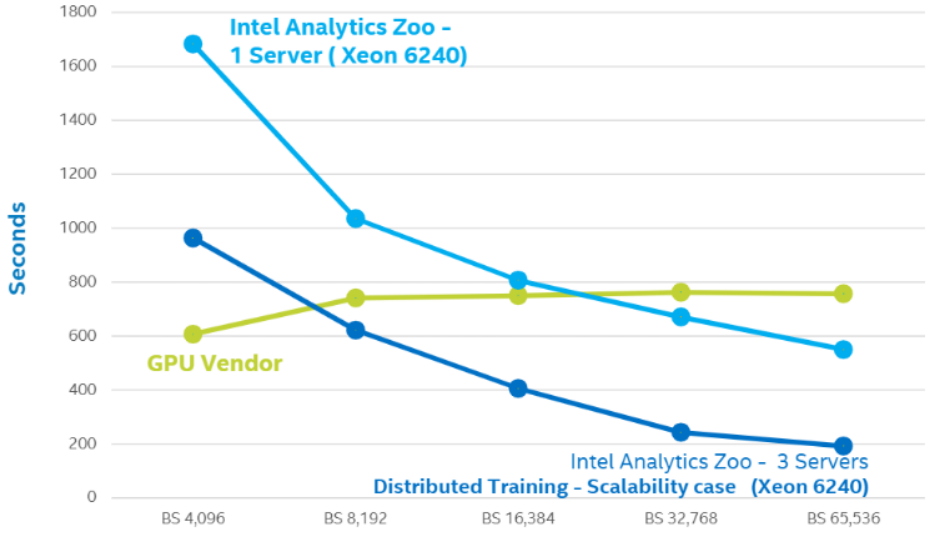
Time Series Forecasting in SKT

TCO OPTIMIZED AI PERFORMANCE WITH [1] ANALYTICS ZOO [2] INTEL OPTIMIZED TENSORFLOW [3] DISTRIBUTED AI PROCESSING

[1] PRE-PROCESSING & INFERENCE LATENCY



[2] TIME-TO-TRAINING PERFORMANCE



Performance test validation @ SK Telecom Testbed

Test Data: **80K** Cell Tower, **8** days, **5**mins period, **8** Quality Indicator

Recap

- AI on Spark can simply your System architecture and speed up end2end AI pipeline and application development
- Analytics Zoo is an AI platform that helps you to build end2end AI pipelines on Spark
- TFPark in Analytics Zoo can make you easy to build TF application on Spark

LEGAL DISCLAIMERS

- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.
- No computer system can be absolutely secure.
- Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit **http://www.intel.com/performance**.