End-to-End Big Data AI Pipeline on Ray and Apache Spark using Analytics Zoo

Jason Dai

Intel Fellow

intel.

CVPR 2021 Tutorial

Agenda

- End-to-End Big Data AI Pipelines
- Analytics Zoo: Open Source Platform for Big Data AI
- Case Study
- Seamlessly Scaling out Big Data AI using Orca in Analytics Zoo

Agenda

- End-to-End Big Data AI Pipelines
- Analytics Zoo: Open Source Platform for Big Data AI
- Case Study
- Seamlessly Scaling out Big Data AI using Orca in Analytics Zoo

Open Source Big Data AI Projects at Intel

BigDL

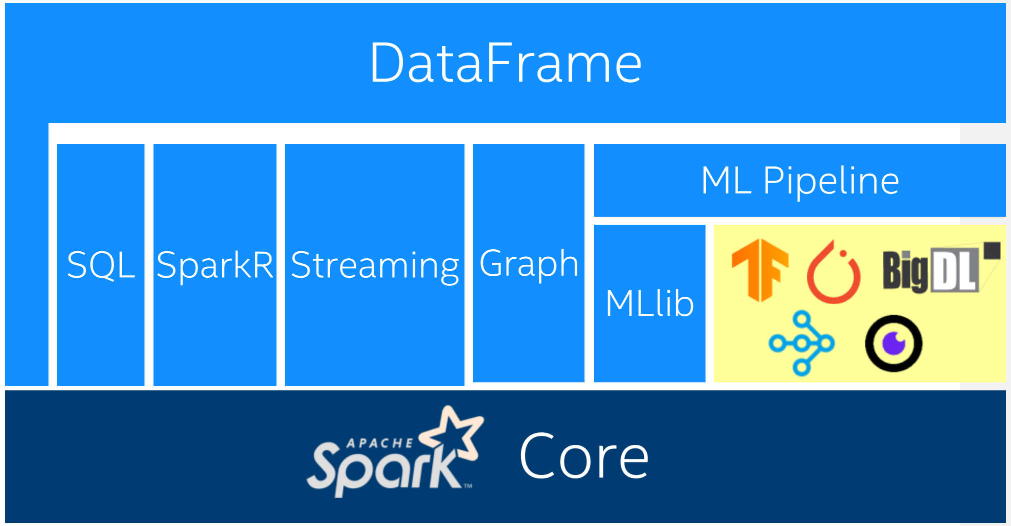
Distributed deep learning framework for Apache Spark

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

ANALYTICS ZOO

Big Data AI Platform
(*distributed TF, PyTorch, BigDL, Ray and OpenVINO on Apache Spark*)

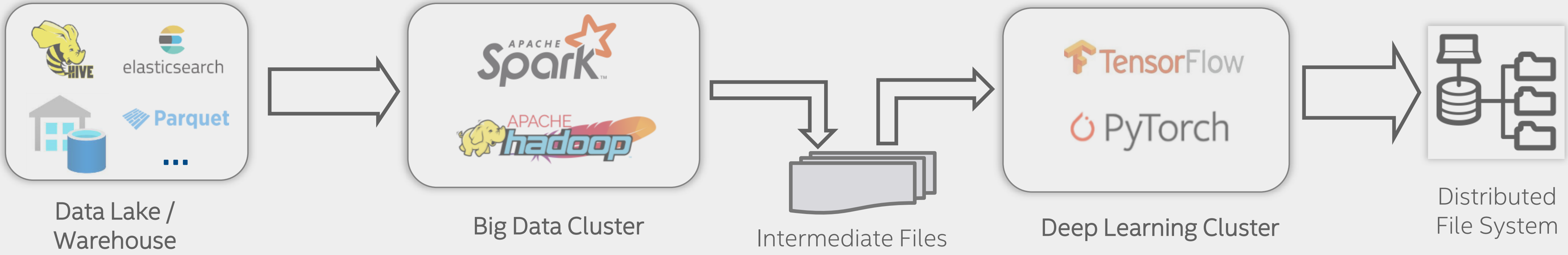
https://github.com/intel-analytics/analytics-zoo



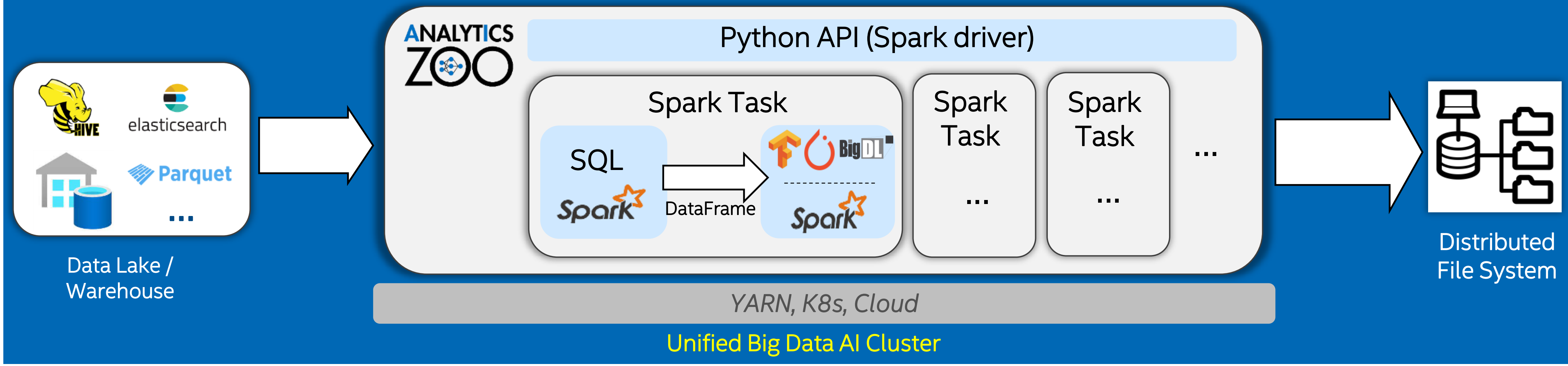
Analytics Zoo

Unified Architecture for E2E AI Pipelines

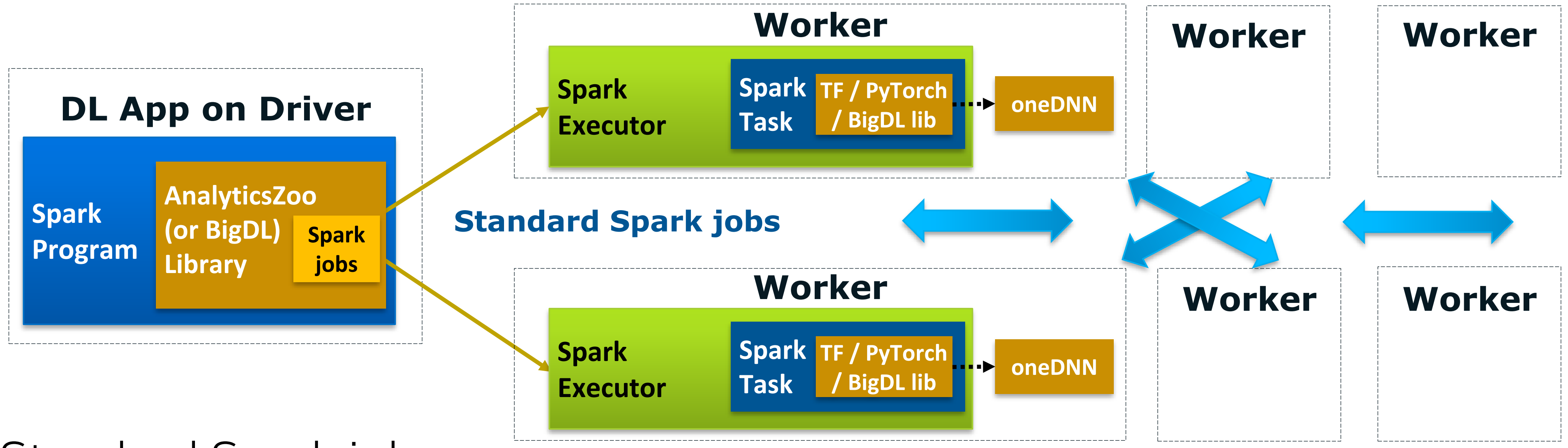
Traditional, Segregated Infrastructure



Unified Architecture (using Analytics Zoo)



Distributed Deep Learning as Spark Jobs



- Standard Spark jobs
  - *Run distributed DL on existing, general-purpose Big Data clusters (Spark, Hadoop, K8s, Hosted, …)*
  - *Seamless integration with Big Data ecosystem (Spark Dataframes & MLlib, Kafka, etc.)*
- Iterative, data-parallel, synchronous SGD
  - *Each jobs runs a training iteration, each task runs the same model on a subset of the batch*
  - *Efficient AllReduce built on top of existing Spark primitives*

* *"BigDL: A Distributed Deep Learning Framework for Big Data", ACM SoCC 2019,* https://arxiv.org/abs/1804.05839

Agenda

- End-to-End Big Data AI Pipelines
- Analytics Zoo: Open Source Platform for Big Data AI
- Case Study
- Seamlessly Scaling out Big Data AI using Orca in Analytics Zoo

Analytics Zoo Stack for Big Data AI

Chronos	**Scalable AutoML for Time Series Prediction**
PPML	**Privacy Preserving Big Data Analytics & ML on SGX**
RayOnSpark	**Run Ray programs directly on Spark Cluster**
Orca	**Seamless scale out TF, PyTorch, BigDL & OpenVINO on Spark**
BigDL	**Distributed deep learning library for Spark**

Powered by oneAPI

BigDL: Distributed DL Framework for Spark

Keras-like API and Spark ML Pipeline Support (Python and Scala APIs)

```
#Keras-like API for BigDL
model = Sequential().add(InputLayer(inputShape = Shape(10)) \
  .add(Dense(12)).add(Activation("softmax"))
model.compile(…)

#Spark Dataframe preprocessing
trainingDF = spark.read.parquet("train_data")
validationDF = spark.read.parquet("val_data")

#Spark ML Pipeline for BigDL
scaler = MinMaxScaler(inputCol="in", outputCol="value")
estimator = NNEstimator(model, CrossEntropyCriterion())  \
  .setBatchSize(size).setOptimMethod(Adam()).setMaxEpoch(epoch)
pipeline = Pipeline().setStages([scaler, estimator])

pipelineModel = pipeline.fit(trainingDF)
predictions = pipelineModel.transform(validationDF)
```

Analytics Zoo API in blue

Orca: Distributed TF/PyTorch/BigDL on Spark

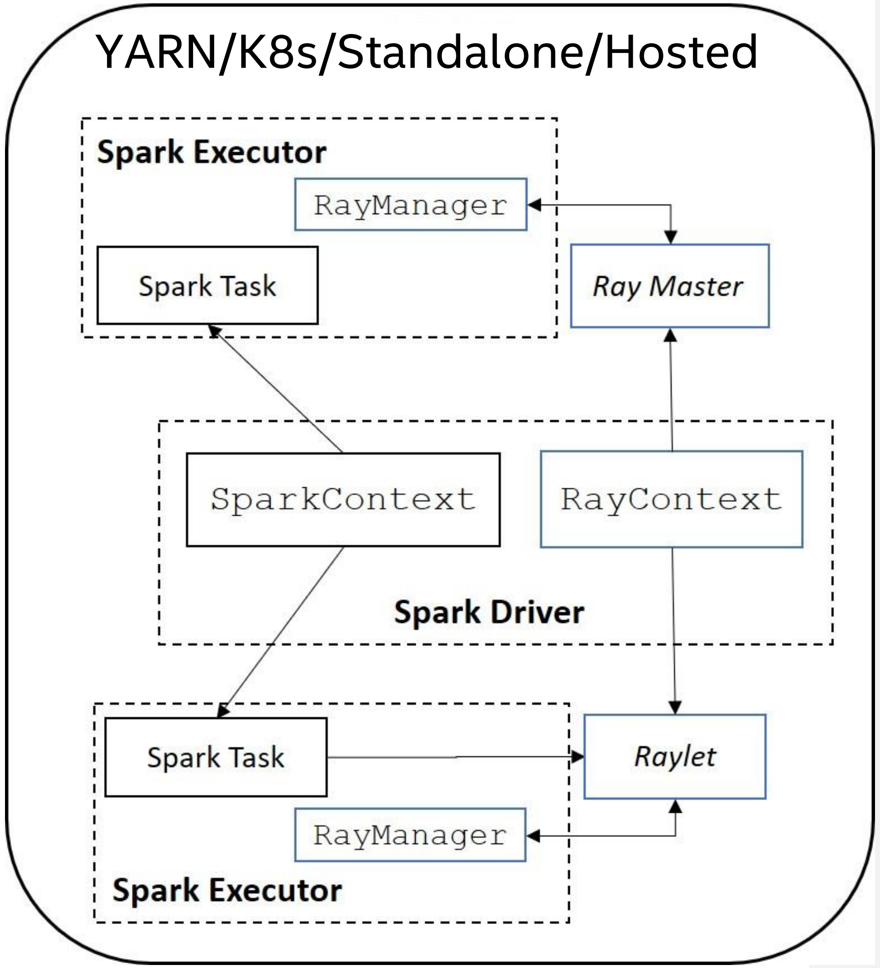
Write TensorFlow/PyTorch inline with Spark Program

```
#PySpark DataFrame
train_df = sqlcontext.read.parquet(…).withColumn(…)

#TensorFlow Model
from tensorflow import keras
…
model = keras.Model(inputs=[user, item], outputs=outputs)
model.compile(optimizer= "adam",
              loss= "sparse_categorical_crossentropy",
              metrics=['accuracy'])

#Distributed training on Spark
from zoo.orca.learn.tf.estimator import Estimator
est = Estimator.from_keras(keras_model=model)
est.fit(train_df, feature_cols=['user', 'item'], label_cols=['label'])
```

Analytics Zoo API in blue

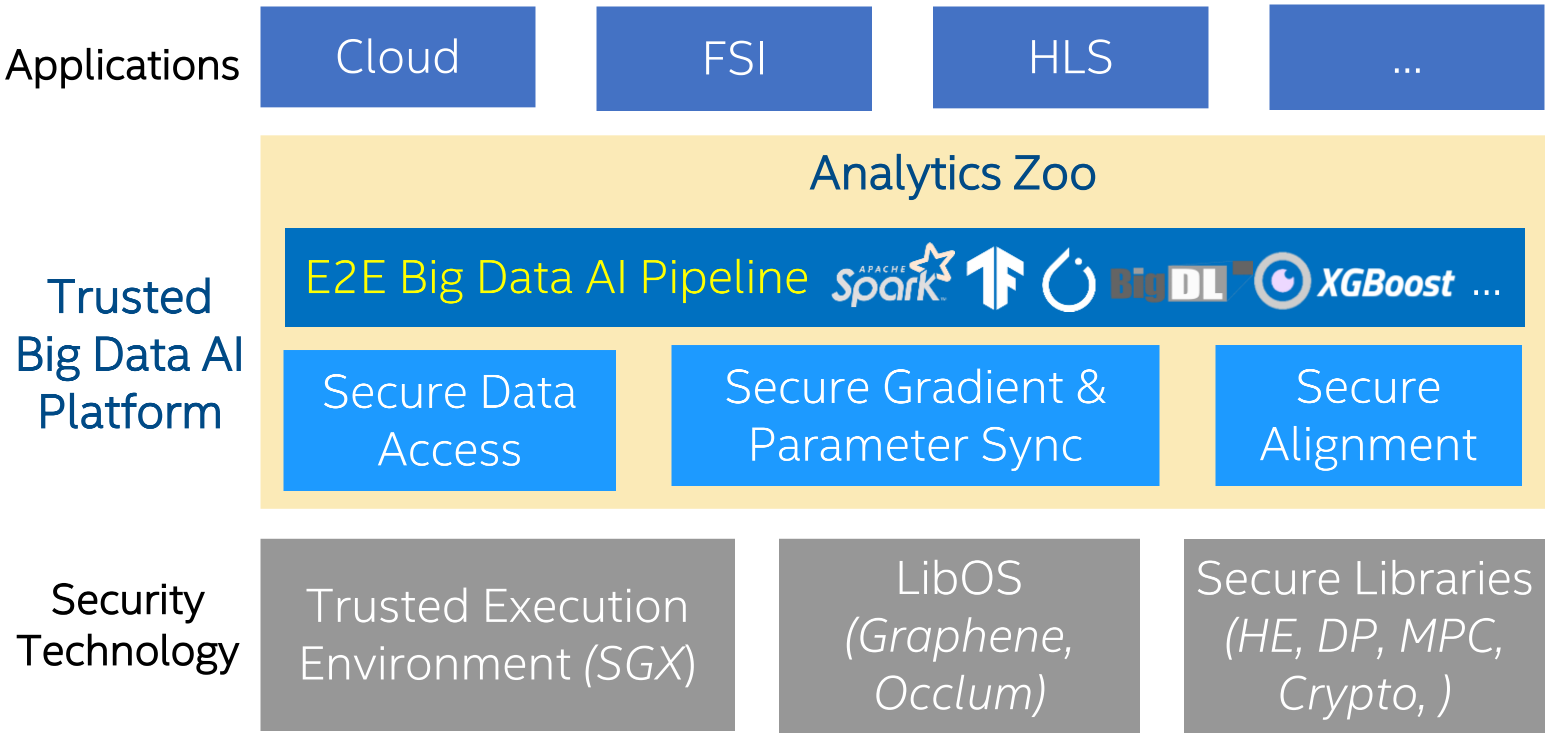
RayOnSpark: Run Ray Programs Directly on Spark

```
from zoo.orca import init_orca_context
sc = init_orca_context(cluster_mode="yarn",  ...,
    init_ray_on_spark=True)

import ray
@ray.remote
class Counter(object):
      def __init__(self):
          self.n = 0

      def inc(self):
          self.n += 1
          return self.n

counters = [Counter.remote() for i in range(5)]
print(ray.get([c.inc.remote() for c in counters]))
```



Analytics Zoo API in blue

PPML: Privacy Preserving Big Data Analytics & ML on SGX

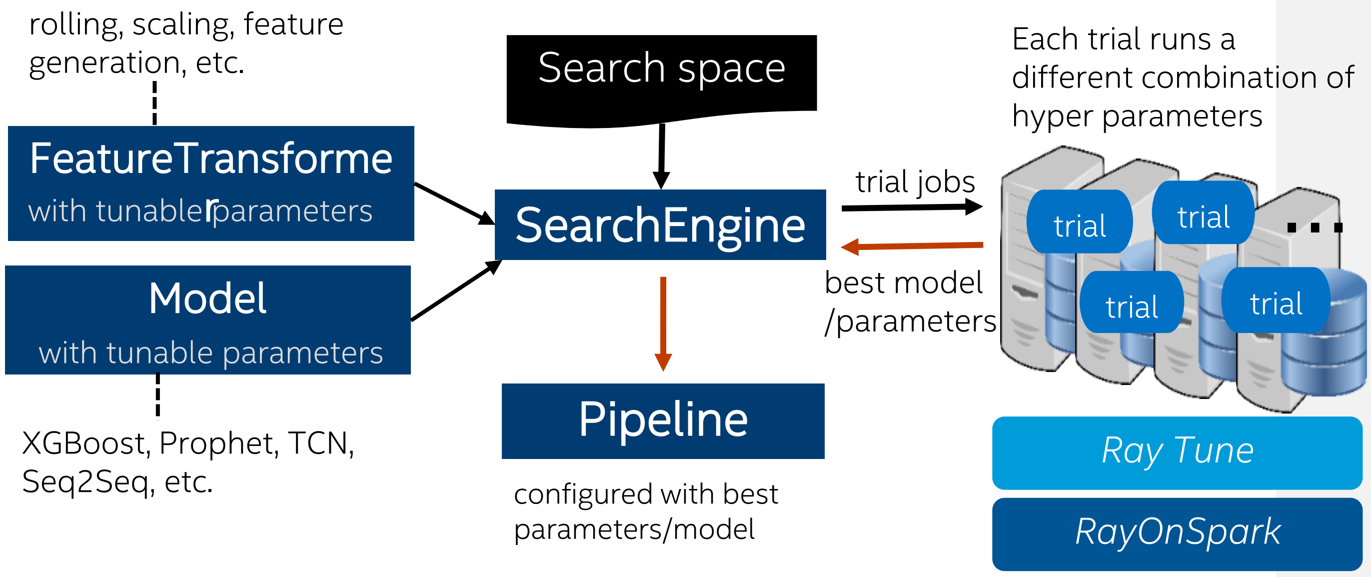


Trusted Big Data AI on *Untrusted Cloud*

- Compute / memory protected by SGX enclaves
- Network protected by TLS and remote attestation
- Storage (e.g., data and model) protected by encryption
- User request / response protected by TLS and encryption

Chronos: Scalable AutoML for Time Series Prediction

```
sc = init_orca_context(...,
     init_ray_on_spark=True)

auto_est = AutoProphet(...)
#auto_est = AutoXGBRegressor(…)
data = get_data()
search_space = {
    "changepoint_prior_scale": ...,
    "seasonality_prior_scale": ...,
    ...
 }

auto_est.fit(data=data,
       search_space=search_space,
       ...)
best_model = auto_est.get_best_model()
```



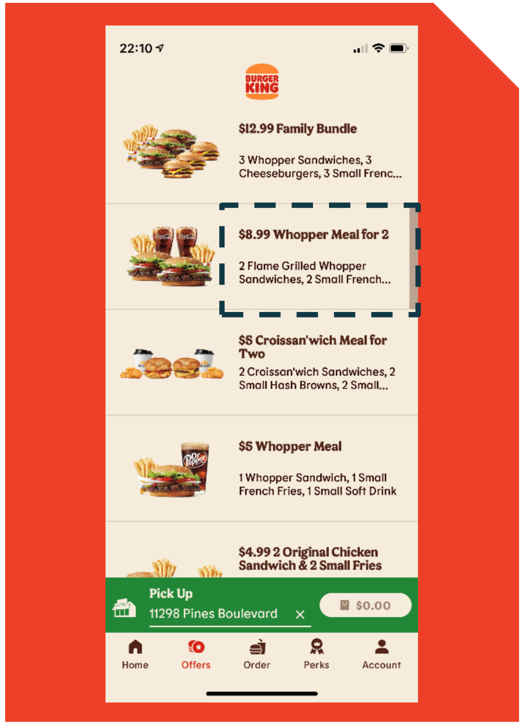
Analytics Zoo API in blue

Agenda

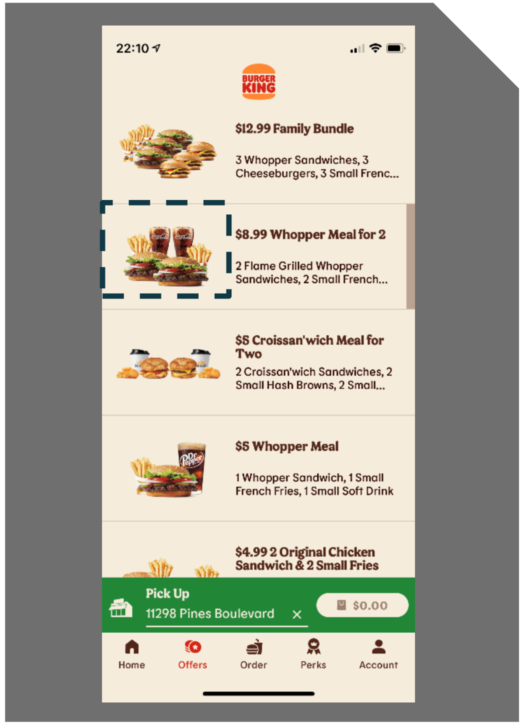
- End-to-End Big Data AI Pipelines
- Analytics Zoo: Open Source Platform for Big Data AI
- Case Study
- Seamlessly Scaling out Big Data AI using Orca in Analytics Zoo

Burger King's Offer Recommendation System: DeepFlame

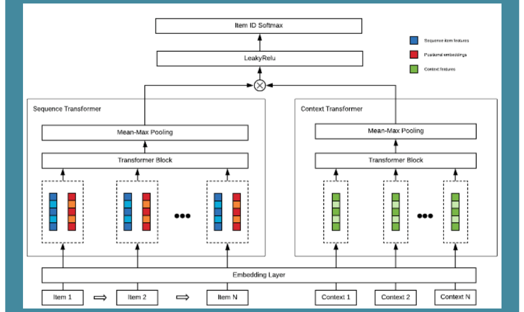
- BERT



- ResNET50



- TxT*



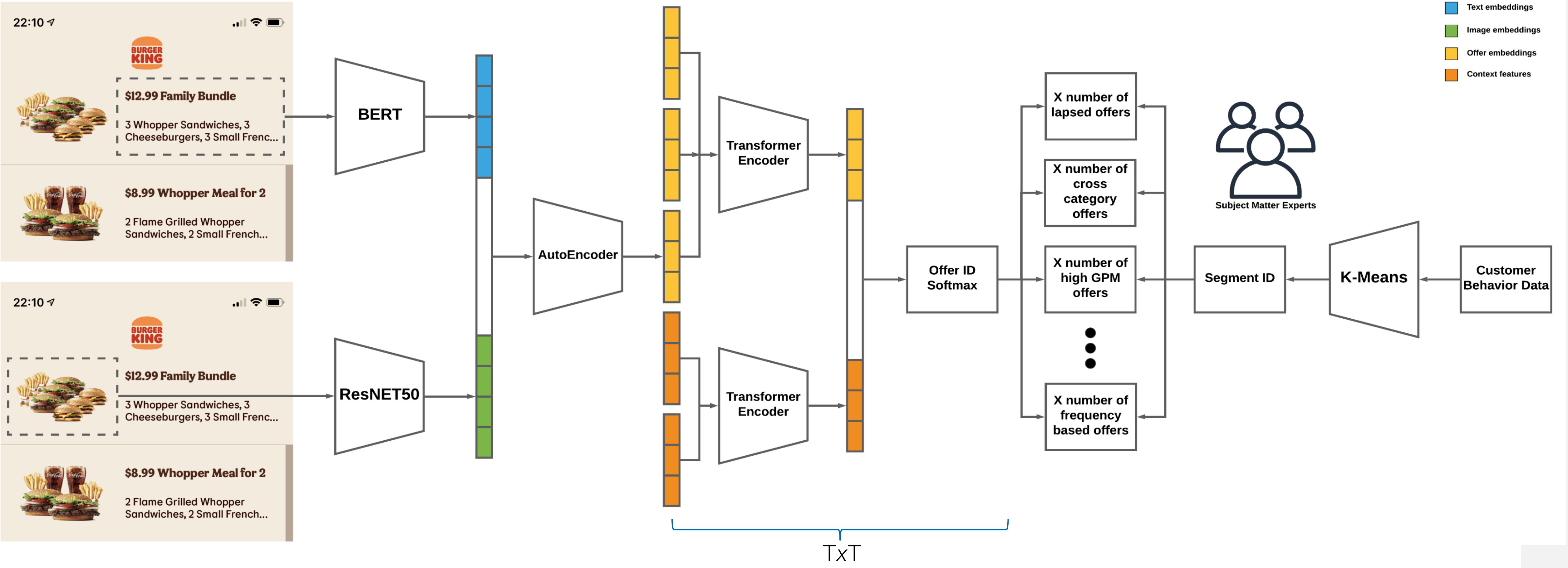
- K-Means
  - K-Means Clustering based on customer's behavior data such as average spend, primary service channel, average ticket GPM, and visit frequency, etc.

*https://arxiv.org/abs/2010.06197
https://github.com/intel-analytics/analytics-zoo/blob/master/pyzoo/zoo/models/recommendation/txt.py

Source: "Offer Recommendation System with Apache Spark at Burger King", Luyang Wang and Kai Huang, Data+AI Summit 2021

DeepFlame Overview – Model Training

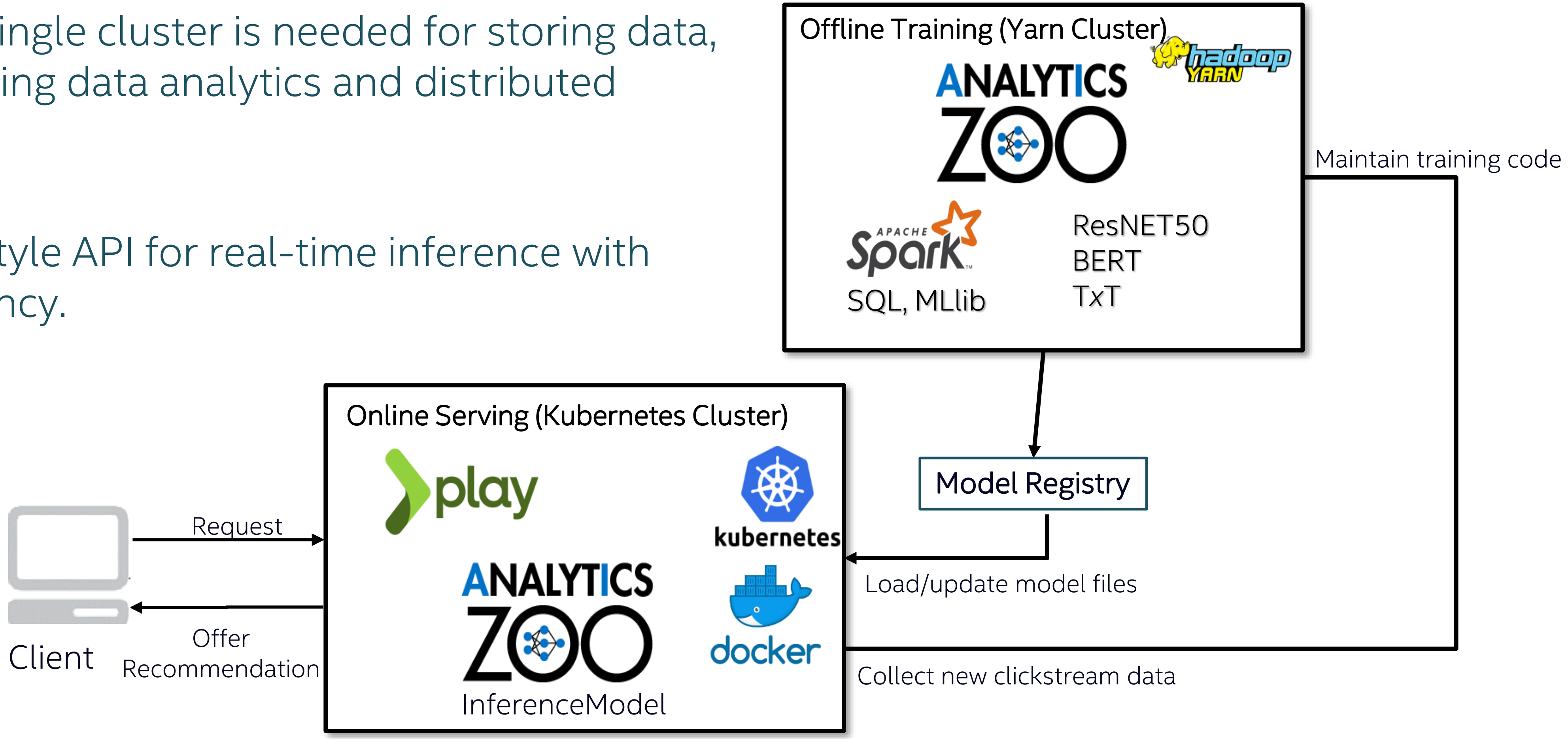
A hybrid approach that allows SME to easily maintain and modify offer rules based on segmentations while still allowing DL models to automatically pick the best offers according to preset offer rules.



Source: *"Offer Recommendation System with Apache Spark at Burger King", Luyang Wang and Kai Huang, Data+AI Summit 2021*

Offer Recommendation System In Production

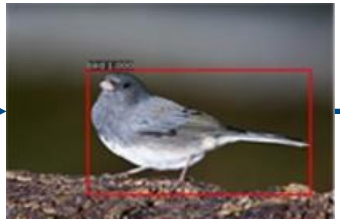
- Only a single cluster is needed for storing data, performing data analytics and distributed training.
- POJO-style API for real-time inference with low latency.



Source: *"Offer Recommendation System with Apache Spark at Burger King", Luyang Wang and Kai Huang, Data+AI Summit 2021*

Case Study: Image Feature Extraction at JD.com

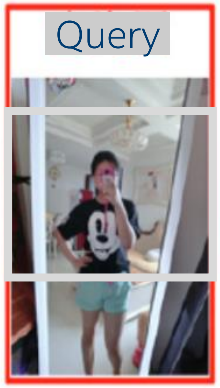
Image Feature Extraction:



1
0
1
0
1
0

Applications:

Similar Image Search



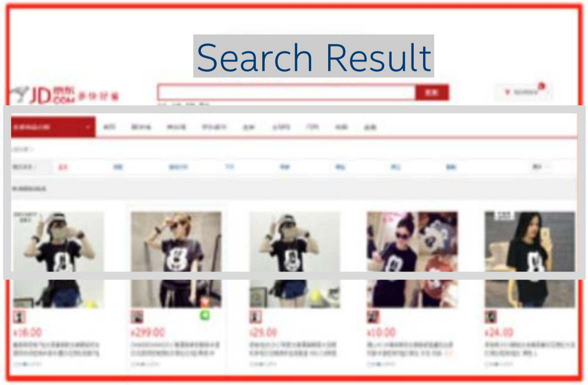
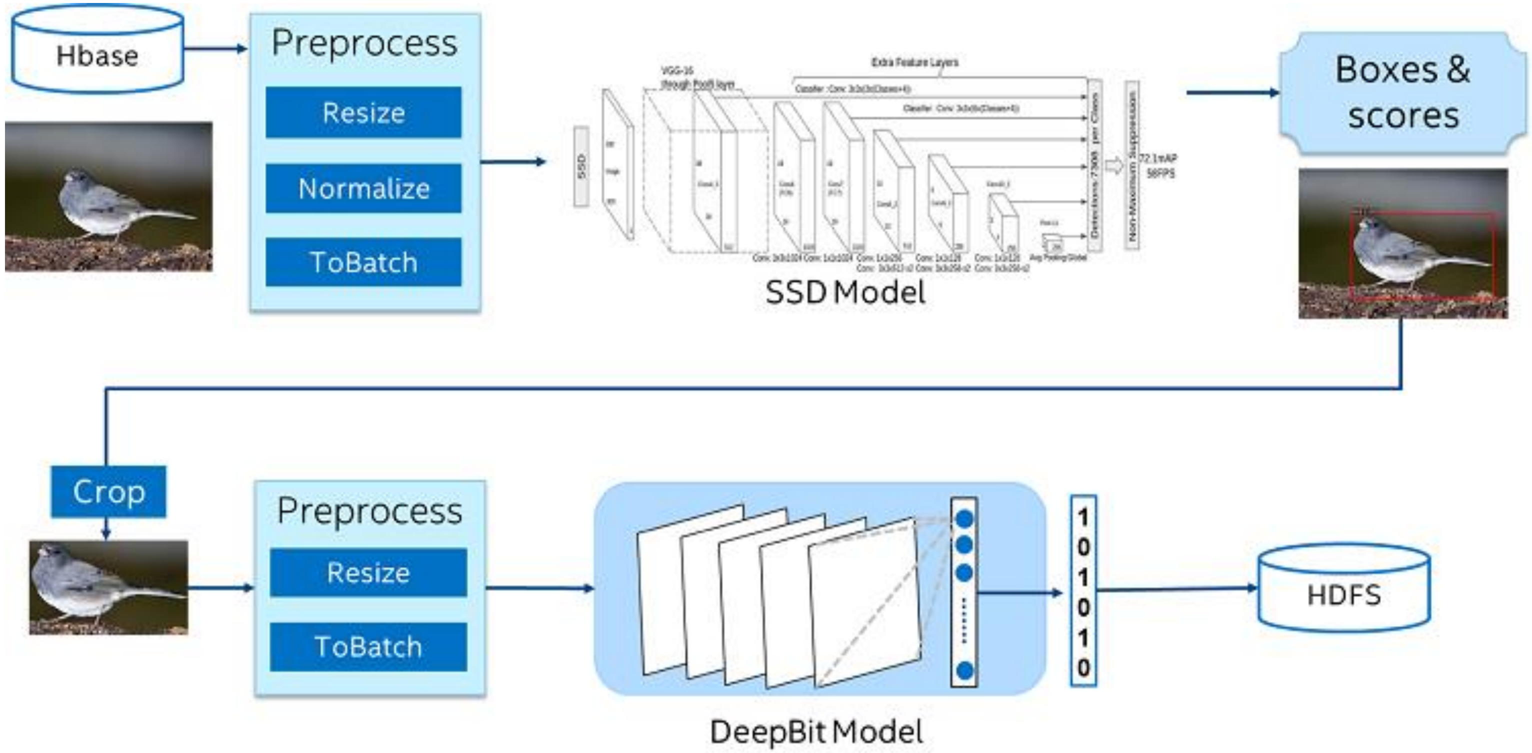


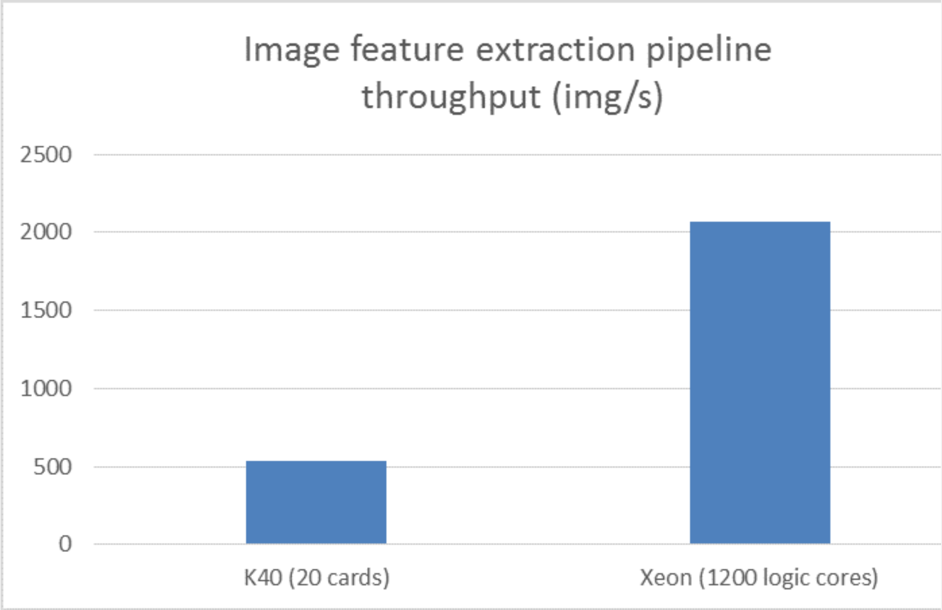
Image Deduplication

Source: "Bringing deep learning into big data analytics using BigDL", Xianyan Jia and Zhenhua Wang, Strata Data Conference Singapore 2017

3.83x Speedup of E2E Inference Pipeline at JD.com

3.83x speedup for end-to-end inference running BigDL on Xeon (vs. Nvidia GPU severs)*

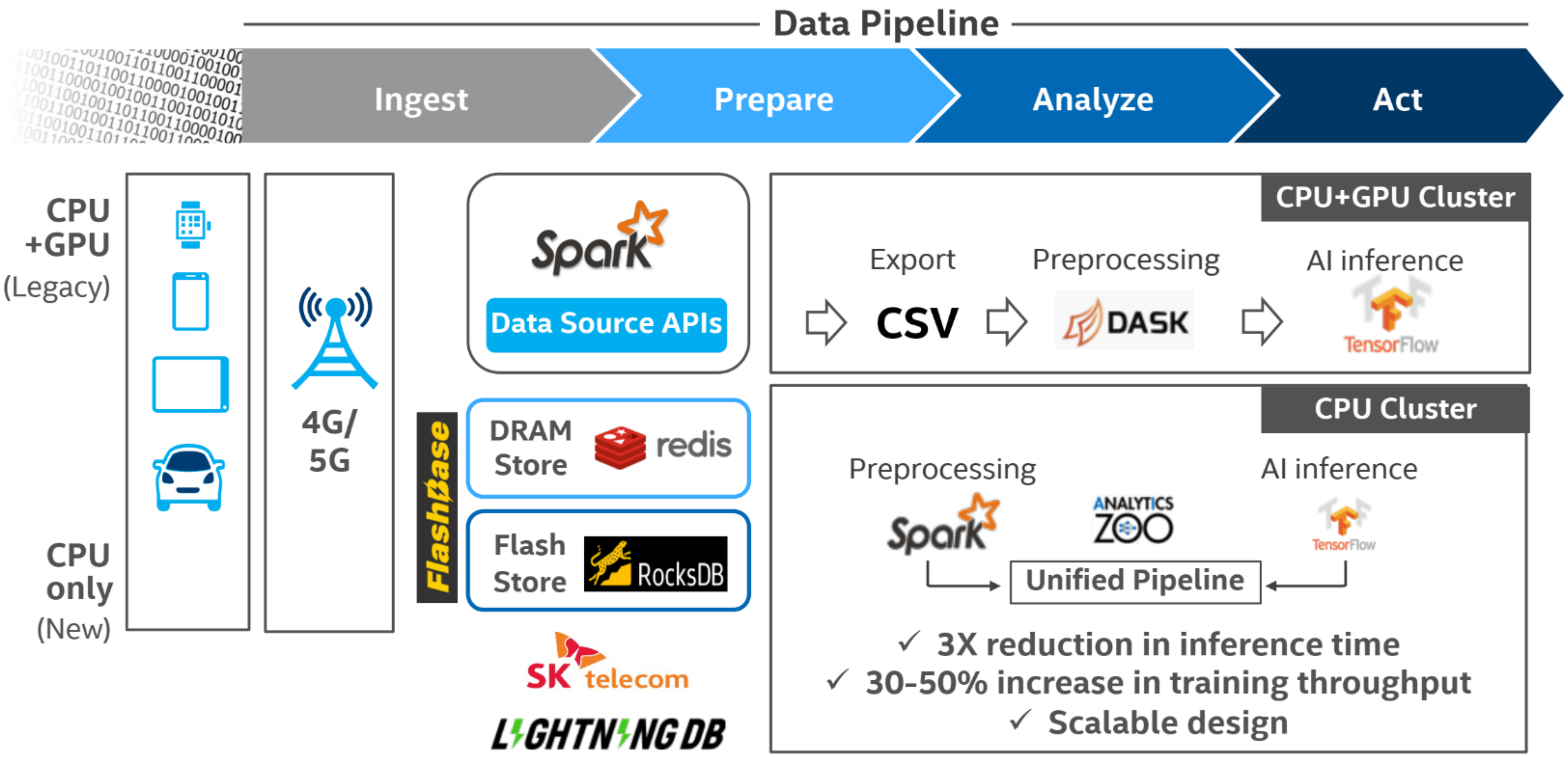




* https://software.intel.com/en-us/articles/building-large-scale-image-feature-extraction-with-bigdl-at-jdcom

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

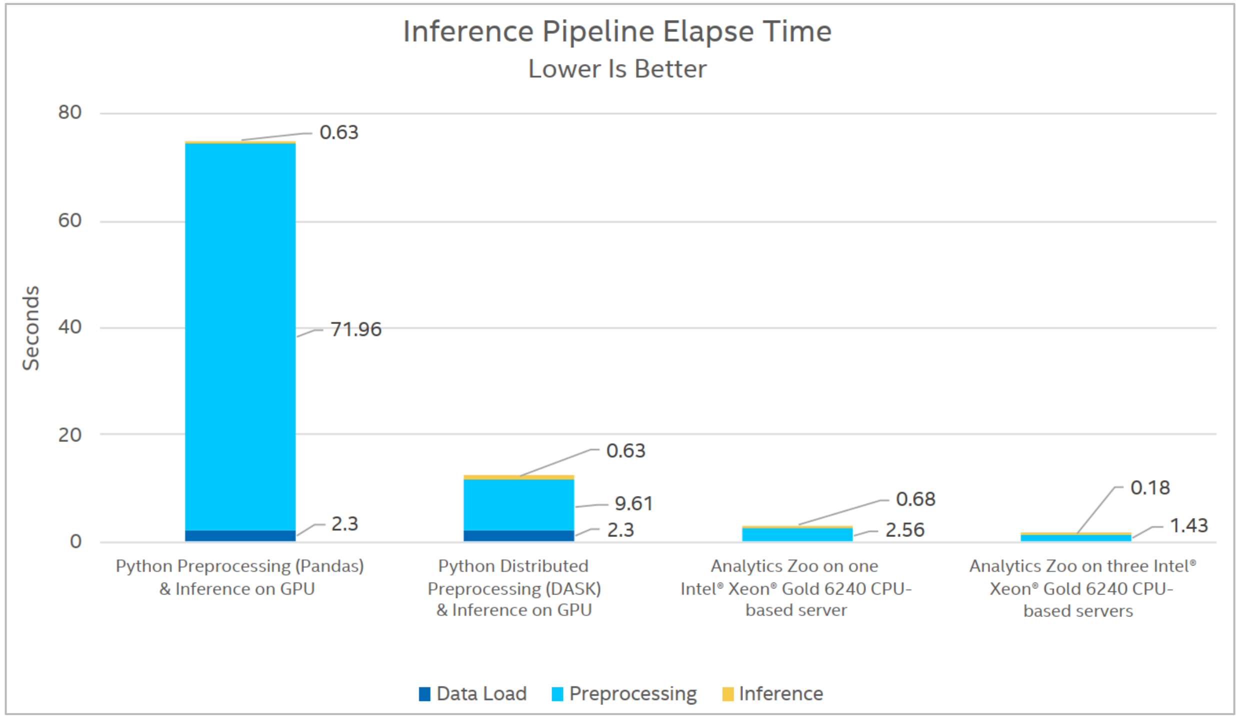
Case Study: Time Series Based Network Quality Prediction in SK Telecom



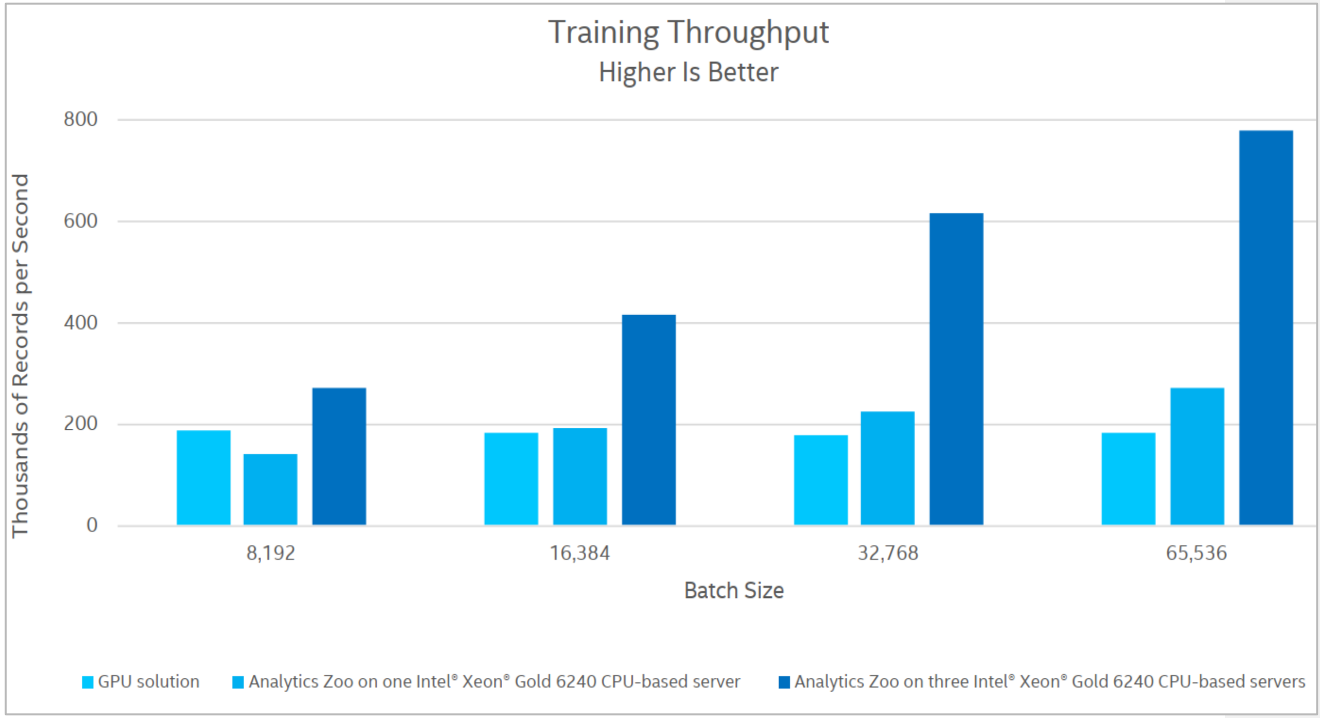
https://networkbuilders.intel.com/solutionslibrary/sk-telecom-intel-build-ai-pipeline-to-improve-network-quality

Up-to 3x End-to-End Speedup at SK Telecom

3x speedup for E2E inference running Analytics Zoo on Xeon*



30~50% speedup for training throughput running Analytics Zoo on Xeon*



*https://networkbuilders.intel.com/solutionslibrary/sk-telecom-intel-build-ai-pipeline-to-improve-network-quality

For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Agenda

- End-to-End Big Data AI Pipelines
- Analytics Zoo: Open Source Platform for Big Data AI
- Case Study
- Seamlessly Scaling out Big Data AI using Orca in Analytics Zoo

Orca: Distributed TF/PyTorch/BigDL on Spark

Write TensorFlow/PyTorch inline with Spark program

PyTorch Quickstart

- Seamless scaling-out of standard PyTorch Dataloader and model on distributed clusters

  https://analytics-zoo.readthedocs.io/en/latest/doc/Orca/QuickStart/orca-pytorch-quickstart.html

Orca: Distributed TF/PyTorch/BigDL on Spark

Write TensorFlow/PyTorch inline with Spark program

TensorFlow 1.15 Quickstart

- Seamless scaling-out of standard TensorFlow Dataset and compute graph on distributed clusters

  https://analytics-zoo.readthedocs.io/en/latest/doc/Orca/QuickStart/orca-tf-quickstart.html

Orca: Distributed TF/PyTorch/BigDL on Spark

Write TensorFlow/PyTorch inline with Spark program

Keras Quickstart

- Seamless scaling-out of standard TensorFlow Dataset and Keras model on distributed clusters

  https://analytics-zoo.readthedocs.io/en/latest/doc/Orca/QuickStart/orca-keras-quickstart.html

Orca: Distributed TF/PyTorch/BigDL on Spark

Write TensorFlow/PyTorch inline with Spark program

Distributed Pandas with XShards for Deep Learning

- XShards: Seamless scaling-out of existing Python codes in a distributed and data-parallel fashion

  https://analytics-zoo.readthedocs.io/en/latest/doc/UseCase/xshards-pandas.html

End-to-End Big Data AI Pipelines on Orca

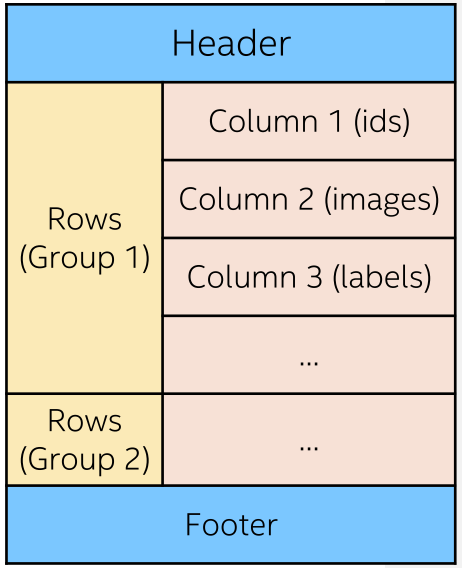
Seamless Scaling-Out of End-to-End AI Pipeline

Computer Vision Pipelines	Massive amount of small (image) files on distributed file system	Distributed (image) preprocessing and transformations	Distributed training	Distributed inference

Organizing Massive Amount of Image Files for Distributed Cluster

- Conventional approach
  - Directory of many small image files
  - Inefficient for distributed storage system
- Orca library
  - Storing small image files as large file(s) in Apache Parquet format
  - Directly read as TensorFlow Dataset or PyTorch Dataload in a distributed fashion

Apache Parquet format




```python
from orca.data import image

#Support common image formats (image directory, ImageNet, VOC, COCO, etc.)
image.write_parquet(format, path, …)
#Support TensorFlow Dataset, PyTorch DataLoader, etc.
data = image.read_parquet(format, path, …)
```

Distributed YOLO V3 Training

```
#Init Orca Context
from zoo.orca import init_orca_context, stop_orca_context
init_orca_context(cluster_mode="k8s", ...)

#Prepare Data
from orca.data import image
image.write_parquet("voc", input_path, …)

#Process Data
def train_data_creator(config, batch_size):
   train_dataset = image.read_parquet("tf_dataset", voc_train_path, …)
   train_dataset = train_dataset.shuffle(buffer_size=512)
   train_dataset = train_dataset.map(…)
   train_dataset = train_dataset.batch(batch_size)
   return train_dataset
```

Distributed YOLO V3 Training

```
#Define TensorFlow model
from tensorflow import keras
def model_creator(config):
   model = YoloV3(DEFAULT_IMAGE_SIZE, training=True, classes=80)
   ...
   optimizer = keras.optimizers.Adam(lr=1e-3)
   loss = [YoloLoss(anchors[mask], classes=options.class_num)
               for mask in anchor_masks]
   model.compile(optimizer=optimizer, loss=loss,
                      run_eagerly=False)
   return model

#Distributed Training
trainer = Estimator.from_keras(model_creator=model_creator)
trainer.fit(train_data_creator, epochs=3, ...)

stop_orca_context()
```

Summary

- Analytics Zoo: Software Platform for Big Data AI
  - E2E Big Data AI pipeline (distributed TF/PyTorch/BigDL/OpenVINO on Spark & Ray)
  - Advanced AI workflow (AutoML, Time-Series, PPML, etc.)
- Github
  - Project repo: https://github.com/intel-analytics/analytics-zoo
  - Use case: https://analytics-zoo.readthedocs.io/en/latest/doc/Application/powered-by.html
- Technical paper/tutorials
  - ACM SoCC 2019 paper: https://arxiv.org/abs/1804.05839
  - CVPR 2021 tutorial: https://jason-dai.github.io/cvpr2021/
  - AAAI 2019 tutorial: https://jason-dai.github.io/aaai2019/