ANALYTICS ZOO

构建统一的大数据分析+AI 流水

史栋杰· Intel / 资深软件架构师

Apache Flink Community China

CONTENT

目录 >>

01

Overview

AI on Big Data

Distributed, High-Performance

Deep Learning Framework

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

Analytics + AI Platform

Distributed TensorFlow*, Keras*, PyTorch* and BigDL

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

Accelerating Data Analytics + AI Solutions At Scale

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

Real-World ML/DL Applications Are Complex Data Analytics Pipelines

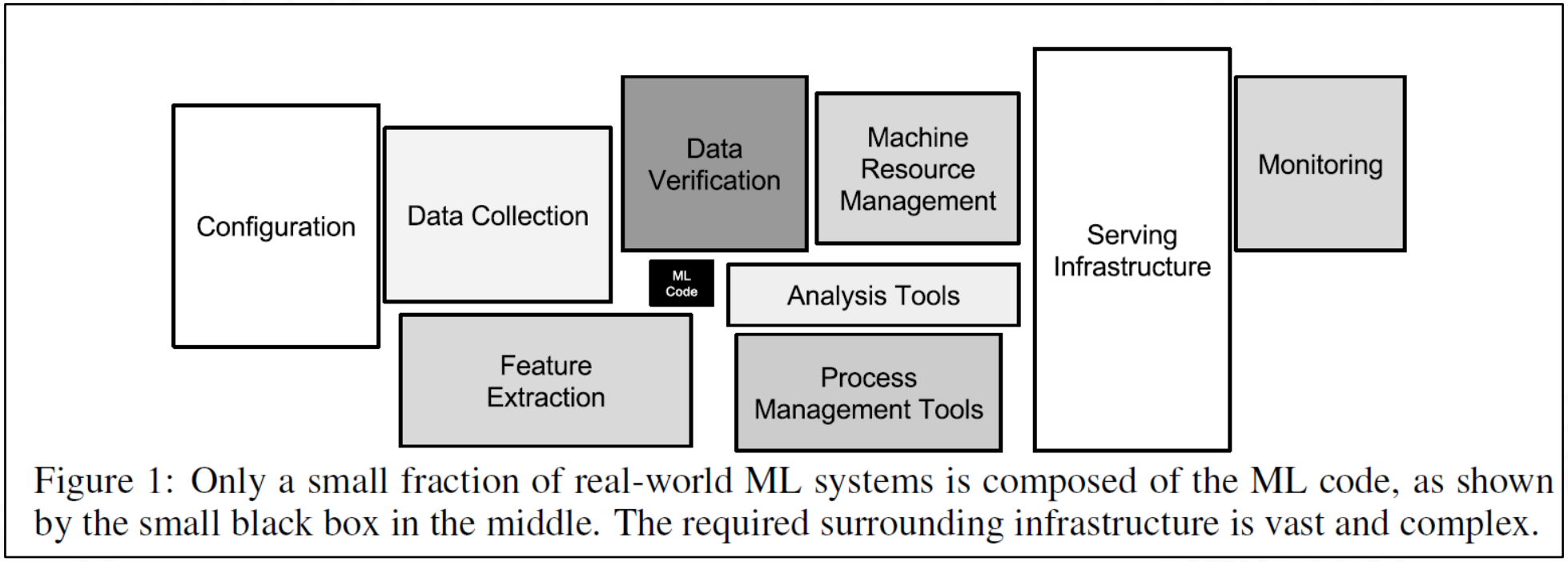
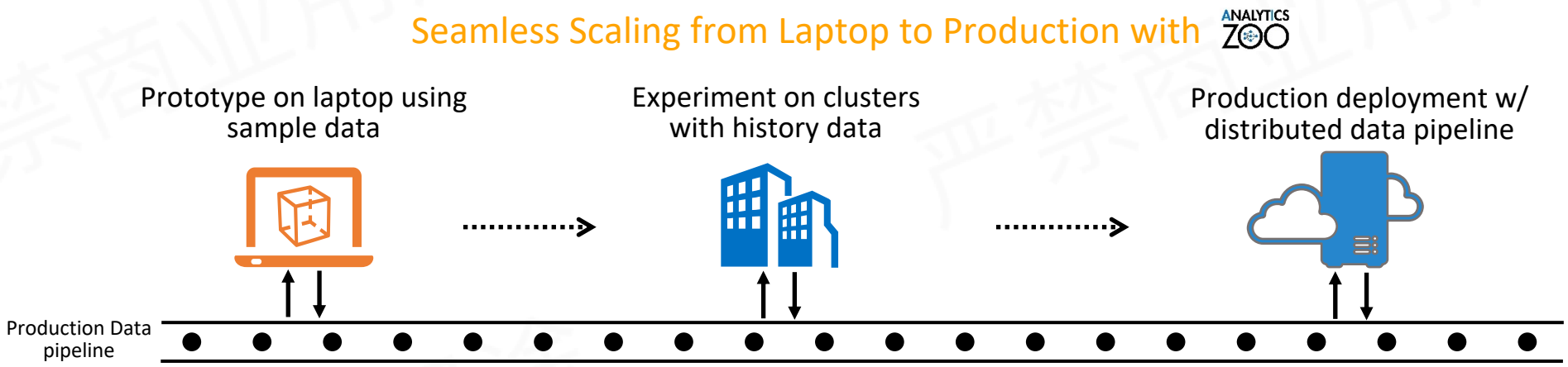


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

"Hidden Technical Debt in Machine Learning Systems",
Sculley et al., Google, NIPS 2015 Paper

End-to-End Big Data Analytics and AI Pipeline



- **"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**

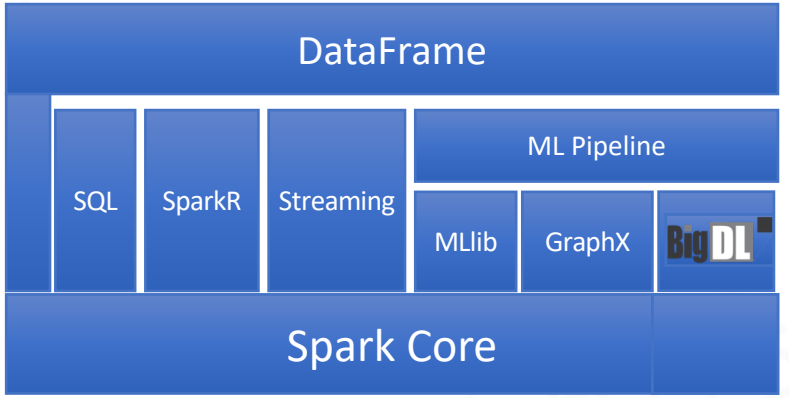
02

BigDL

BigDL

Bringing Deep Learning To Big Data Platform

- **Distributed** deep learning framework for Apache Spark
- Make deep learning more accessible to **big data users** and **data scientists**
  - Write deep learning applications as **standard Spark programs**
  - Run on existing Spark/Hadoop clusters (**no changes needed**)
- Feature parity with popular deep learning frameworks
  - E.g., Caffe, Torch, Tensorflow, etc.
- High performance (on CPU)
  - Powered by Intel MKL and multi-threaded programming
- Efficient scale-out
  - Leveraging Spark for distributed training & inference



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

https://bigdl-project.github.io/

BigDL Run as Standard Spark Programs

Standard Spark jobs

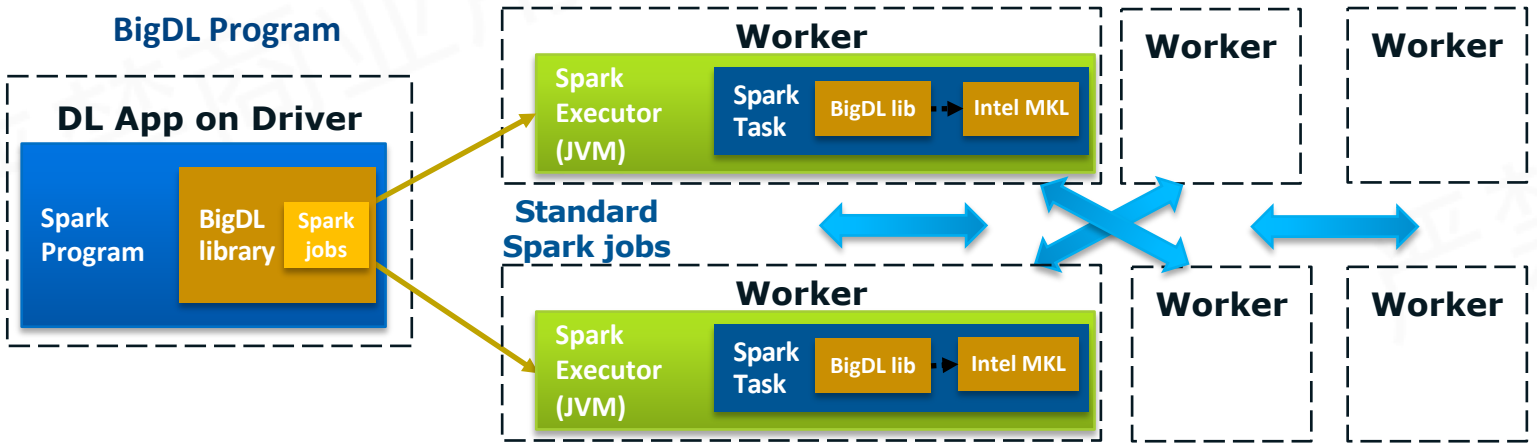
- No changes to the Spark or Hadoop clusters needed

Iterative

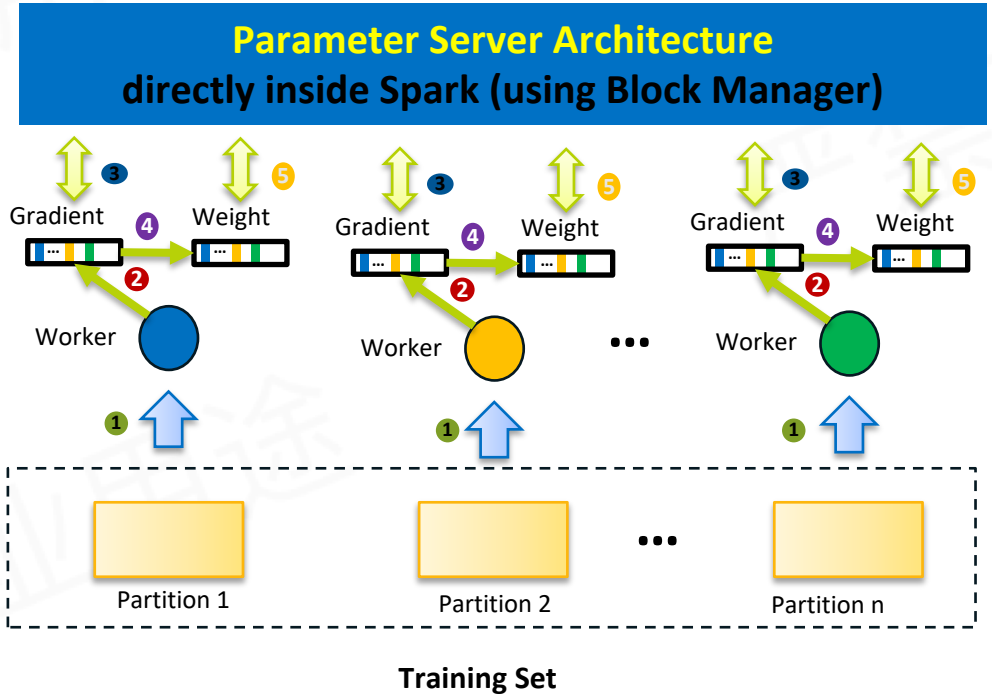
- Each iteration of the training runs as a Spark job

Data parallel

- Each Spark task runs the same model on a subset of the data (batch)



Distributed Training in BigDL

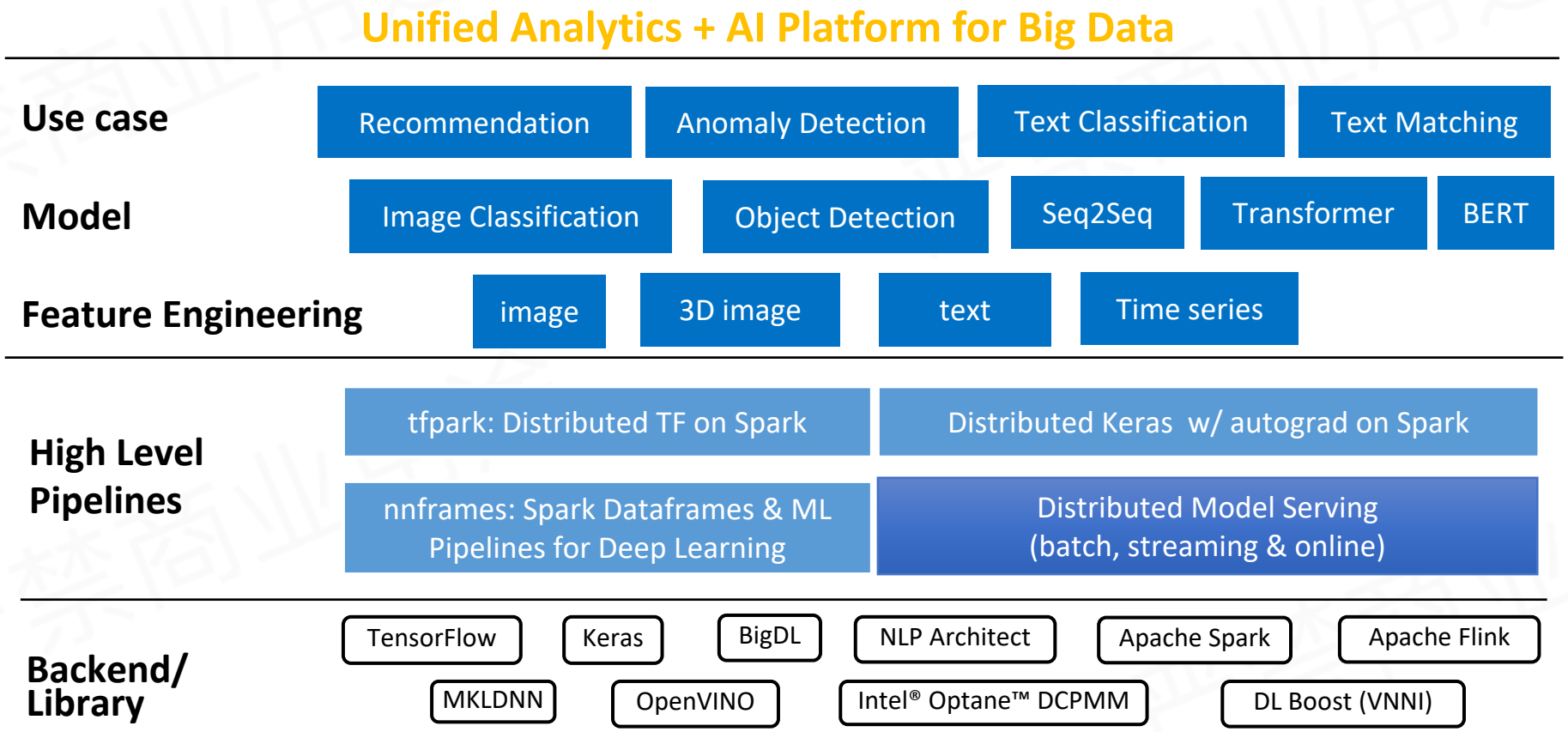


Peer-2-Peer All-Reduce synchronization

03

Analytics Zoo

Analytics Zoo



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

Analytics Zoo

Unified Analytics + AI Platform for Big Data

Build end-to-end deep learning applications for big data

- Distributed *TensorFlow* on Spark
- *Keras* API (with autograd & transfer learning support) on Spark
- *nnframes*: native DL support for Spark DataFrames and ML Pipelines

Productionize deep learning applications for big data at scale

- Plain Java/Python *model serving* APIs (w/ OpenVINO support)
- Support Web Services, Spark, Flink, Storm, Kafka, etc.

Out-of-the-box solutions

- Built-in deep learning *models*, *feature engineering* operations, and reference *use cases*

Distributed TF & Keras on Spark

- **Data wrangling and analysis using PySpark**
- **Deep learning model development using TensorFlow or Keras**
- **Distributed training / inference on Spark**

Write TensorFlow code inline in PySpark program

```
#pyspark code
train_rdd = spark.hadoopFile(…).map(…)
dataset = TFDataset.from_rdd(train_rdd,…)

#tensorflow code
import tensorflow as tf
slim = tf.contrib.slim
images, labels = dataset.tensors
with slim.arg_scope(lenet.lenet_arg_scope()):
   logits, end_points = lenet.lenet(images, …)
loss = tf.reduce_mean( \
   tf.losses.sparse_softmax_cross_entropy( \
   logits=logits, labels=labels))

#distributed training on Spark
optimizer = TFOptimizer.from_loss(loss, Adam(…))
optimizer.optimize(end_trigger=MaxEpoch(5))
```

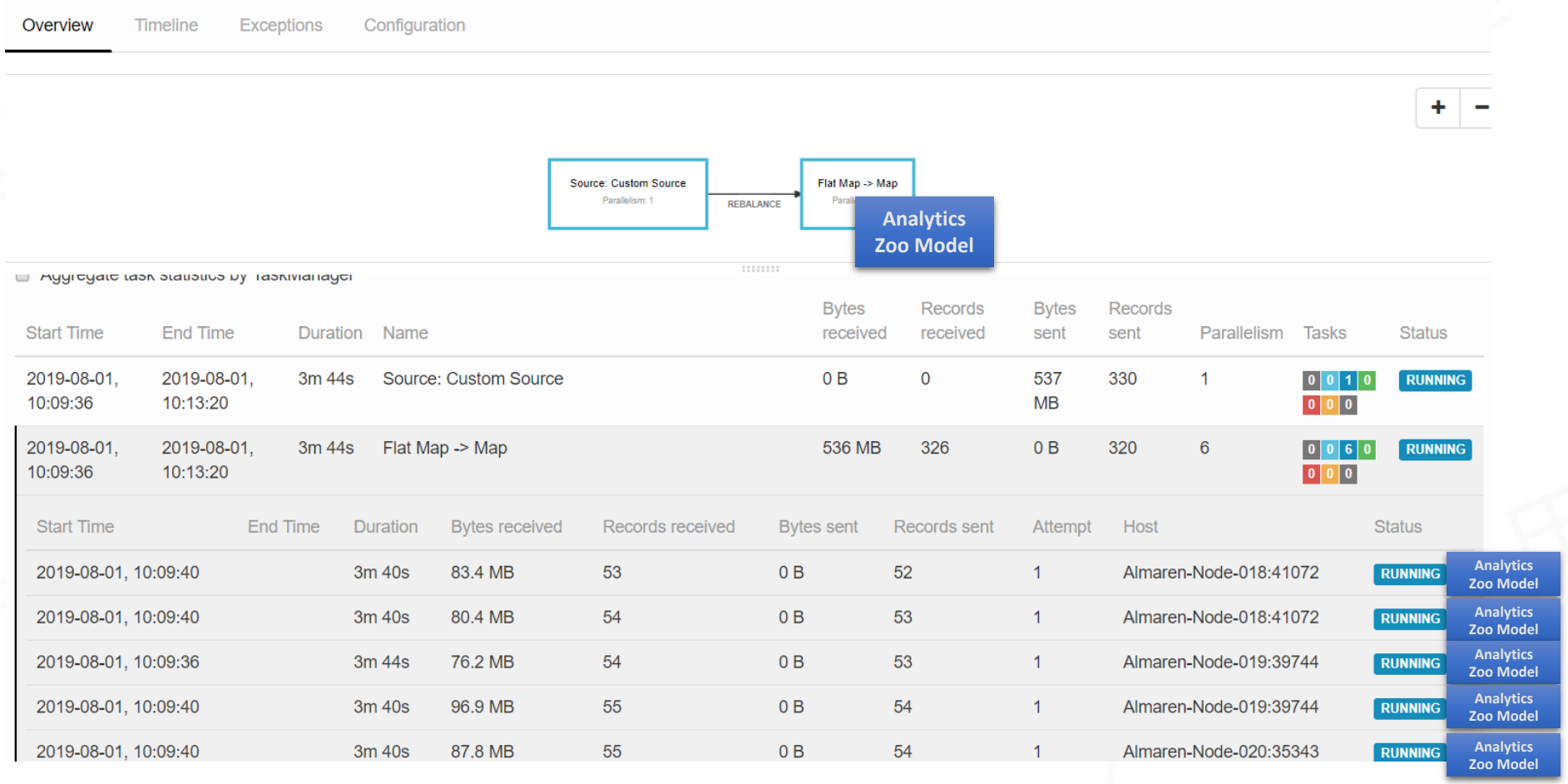
Spark Dataframe & ML Pipeline for DL

```
#Spark dataframe transformations
parquetfile = spark.read.parquet(…)
train_df = parquetfile.withColumn(…)

#Keras API
model = Sequential()
          .add(Convolution2D(32, 3, 3, activation='relu', input_shape=…)) \
          .add(MaxPooling2D(pool_size=(2, 2))) \
          .add(Flatten()).add(Dense(10, activation='softmax')))

#Spark ML pipeline
Estimater = NNEstimater(model, CrossEntropyCriterion()) \
                .setLearningRate(0.003).setBatchSize(40).setMaxEpoch(5) \
                .setFeaturesCol("image")
nnModel = estimater.fit(train_df)
```

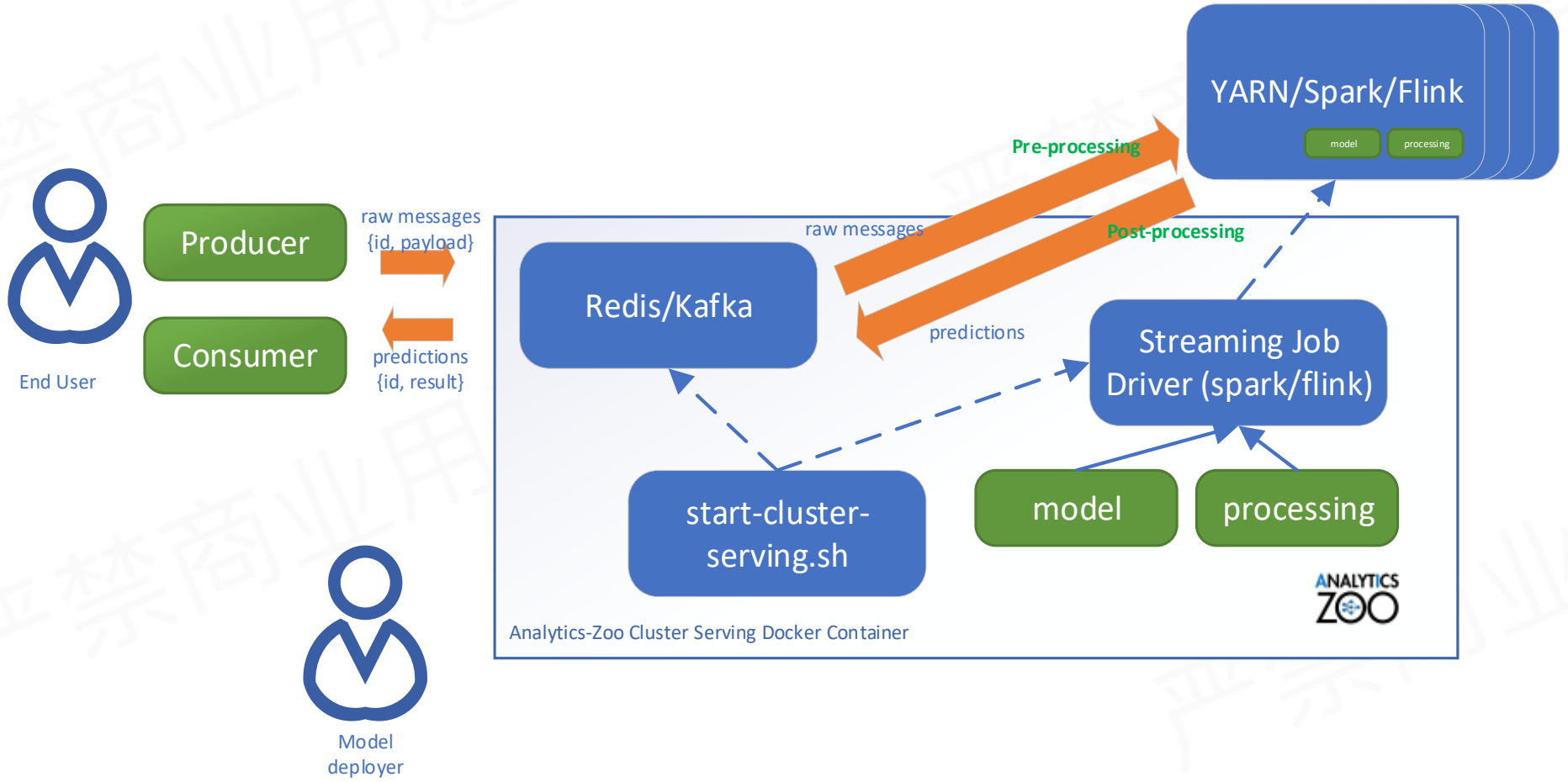
Distributed Model Serving



Distributed model serving in Web Service, Flink, Kafka, Storm, etc.

- **Plain Java or Python API, with OpenVINO and DL Boost (VNNI) support**

Distributed Model Serving



OpenVINO Support for Model Serving

```
from zoo.common.nncontext import init_nncontext
from zoo.feature.image import ImageSet
from zoo.pipeline.inference import InferenceModel

sc = init_nncontext("OpenVINO Object Detection Inference Example")
images = ImageSet.read(options.img_path, sc,
                       resize_height=600, resize_width=600).get_image().collect()
input_data = np.concatenate([image.reshape((1, 1) + image.shape) for image in images], axis=0)

model = InferenceModel()
model.load_tf(options.model_path, backend="openvino", model_type=options.model_type)
predictions = model.predict(input_data)

# Print the detection result of the first image.
print(predictions[0])
```

Transparently support OpenVINO in model serving, which deliver a significant boost for inference speed

Upcoming Analytics Zoo 0.6 Release

- **Distributed PyTorch on Spark**
- **Ray on Spark**
  - **Run Ray programs directly on standard Hadoop/YARN clusters**
- **AutoML support**
  - **Automatic feature generation, model selection and hyper-parameter tuning for *time series prediction***
- **Cluster serving**
  - **Distributed, real-time (streaming) model serving with simple pub-sub interface**

Use Cases

Technology

bluedata
cloudera
CRAY THE SUPERCOMPUTER COMPANY
databricks
DELL EMC
inspur
GIGASPACES innovate with confidence
Lightbend
Qubole

Cloud Service Providers

Alibaba Cloud aliyun.com
aws
Azure
Tencent 腾讯
Baidu 百度
IBM Cloud
Cloud Dataproc
KINGSOFT

End Users

cdhi
Telefonica
中国电信 CHINA TELECOM
THE WORLD BANK
JD.COM 京东
CERN openlab
Midea
韵达 EXPRESS
CISCO
UnionPay 银联

software.intel.com/AIonBigData

Not a full list

04

Analytics Zoo Inference on Flink

Analytics Zoo Inference Model

- **Java AbstractInferenceModel**
  - public void load(String modelPath, String weightPath)
  - public void loadCaffe(String modelPath, String weightPath)
  - public void loadTF(String modelPath)
  - public void loadTF(String modelPath, …)
  - public void loadTF(String savedModelDir, …)
  - public void loadTF(byte[] savedModelBytes, …)
  - public void loadOpenVINO(String modelFilePath, String weightFilePath…)
  - public List<List<JTensor>> predict(List<List<JTensor>> inputs)
- **Scala InferenceModel**

- **POJO style**
- **Thread-Safe**

pom

```xml
<repositories>
    <repository>
        <id>ossrh</id>
        <name>sonatype repository</name>
        <url>https://oss.sonatype.org/content/groups/public/</url>
        <releases>
            <enabled>true</enabled>
        </releases>
        <snapshots>
            <enabled>true</enabled>
        </snapshots>
    </repository>
</repositories>
```

```xml
<dependency>
    <groupId>com.intel.analytics.zoo</groupId>
    <artifactId>analytics-zoo-bigdl_0.9.0-spark_2.4.3</artifactId>
    <version>0.6.0-SNAPSHOT</version>
</dependency>
<dependency>
    <groupId>com.intel.analytics.zoo</groupId>
    <artifactId>zoo-core-dist-all</artifactId>
    <version>0.6.0-SNAPSHOT</version>
    <type>jar</type>
</dependency>
<dependency>
    <groupId>org.scala-lang</groupId>
    <artifactId>scala-library</artifactId>
    <version>2.11.11</version>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-scala_2.11</artifactId>
    <version>1.8.1</version>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-streaming-scala_2.11</artifactId>
    <version>1.8.1</version>
</dependency>
<dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-clients_2.11</artifactId>
    <version>1.8.1</version>
</dependency>
```

Load Model from tf saved model tar, and do prediction

```
class GarbageClassificationInferenceModel(savedModelTarBytes: Array[Byte], inputShape: Array[Int],
  extends InferenceModel( concurrentNum = 1) with Serializable {
  doLoadTF(savedModelTarBytes, inputShape, ifReverseInputChannels, meanValues, scale, input)
  println(this)
}
```

```
class InferenceMapFunction(savedModelTarBytes: Array[
  extends RichMapFunction[A, B] {
  var model: GarbageClassificationInferenceModel = _

  override def open(parameters: Configuration): Unit
    model = new GarbageClassificationInferenceModel(s
  }

  override def close(): Unit = {
    model.release()
  }

  override def map(in: A): B = {
    ...
    val output = model.doPredict(input)
    ...
  }
}
```

THANKS

关注微信公众号 Vererica

获取 Apache Flink 极客挑战赛最新信息

大赛由 intel 、阿里云计算平台事业部、天池平台联合举办

1.两大赛题详细解析

2.大赛最新环节通知预告

3.优秀项目经验分享

4.Flink 系列学习干货