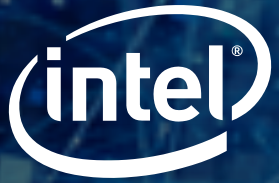


Simplify AI Solutions on Big Data with BigDL / Analytics Zoo - Part 1

Sajan Govindan, Jason Dai, Radhika Rangarajan
Intel

December 2018

Legal notices & disclaimers

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

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.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future, are forward-looking statements that involve a number of risks and uncertainties. A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo, Pentium, Celeron, Atom, Core, Xeon, Movidius, Saffron and others are trademarks of Intel Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

Part 1

- Introduction - AI at Intel
- Motivation
  - Trends, Real world scenarios & challenges
- BigDL
  - Apache Spark recap, BigDL overview
- Analytics Zoo for Apache Spark and BigDL
  - High level pipeline APIs, feature engineering, built-in models, reference use cases
- Analytics Zoo Examples
  - Transfer Learning, Object Detection, TFNet

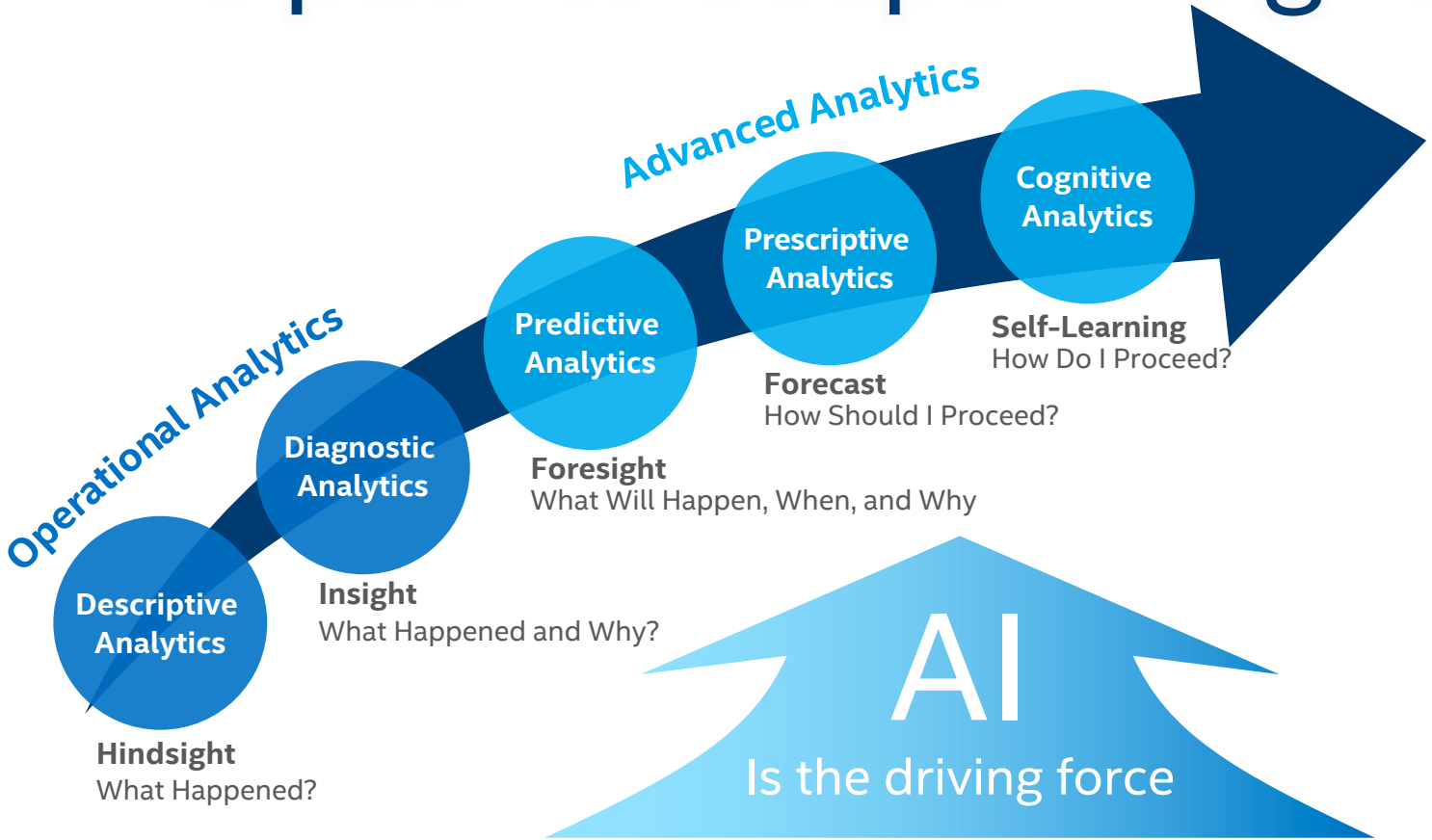
Part 2

- **Building & Deploying**
  - Applications, Partnerships, Cloud, Intel Select Solutions
- **Distributed Training in BigDL**
  - Data parallel training, parameter synchronization, scaling & convergence, task scheduling, etc.
- **Real-world Applications**
  - Object detection and image feature extraction at *JD.com*
  - Image similarity based house recommendation for *MLSlistings*
  - Transfer learning based image classifications for *World Bank*
- **Conclusion**

Introduction – AI at Intel


Data is the basis of competition in the digital era

The path to deeper insight



Intel Analytics & AI Strategy

Smarter Analytics/AI Through the Industry's Most Comprehensive Platform

Data

Intel analytics ecosystem to get your data ready from integration to analysis

Future

Driving AI forward through R&D, investment and policy leadership

AI Is Evolving

Proofs of Concepts → Unlocking real value

AI Is Expanding

End point | edge | Data center

intel MOVIDIUS inside | intel ATOM inside | intel FPGA inside | intel CORE i7 inside | intel NERVANA inside | intel XEON PLATINUM inside

Comprehensive AI portfolio

One Size Does Not Fit ALL

End point

IOT SENSORS
(Security, home, retail, industrial...)

Vision & Inference

Speech

SELF-DRIVING VEHICLE

Autonomous Driving

DESKTOP & MOBILE

Display, video, AR/VR, gestures

CONVERGED MOBILITY

Vision, speech, AR/VR

edge

SERVERS, APPLIANCES & GATEWAYS

Most use cases

Streaming latency-bound systems

Vision & Inference for various systems types

Data center

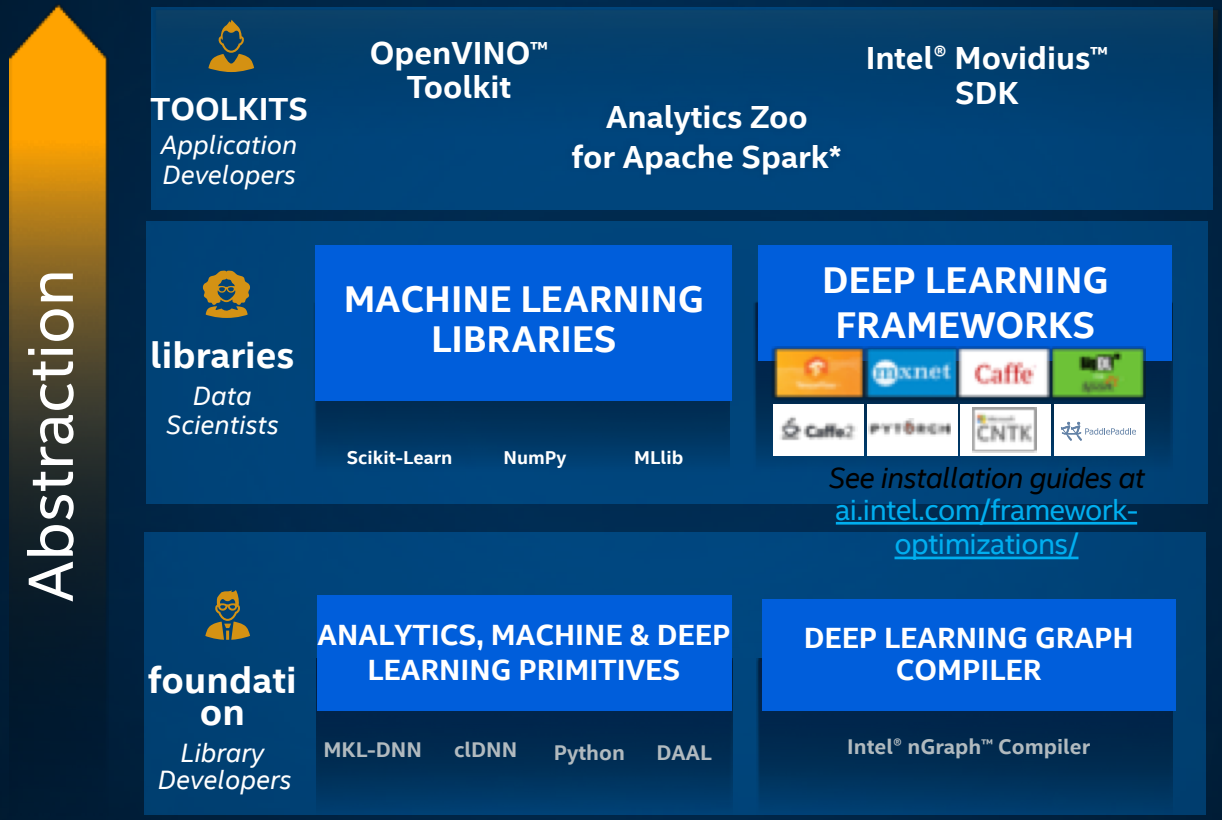
SERVERS & APPLIANCES

Foundation for AI

Built for Deep Learning

Flexible & memory bandwidth bound use cases

Intel-Optimized AI Software & Tools

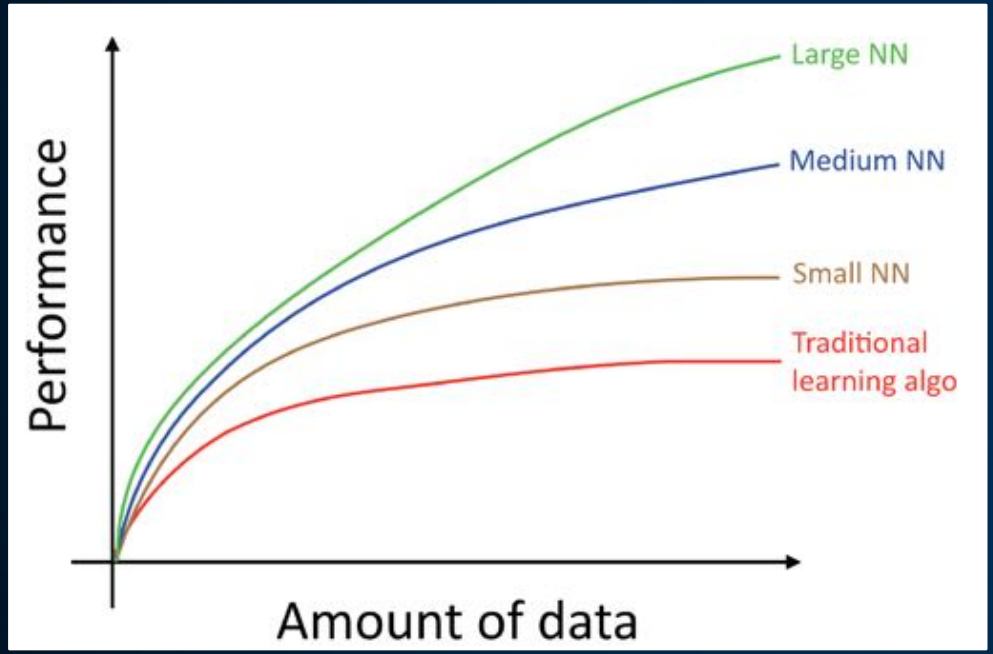


Motivations

Technology and Industry Trends

Real World Scenarios & Challenges

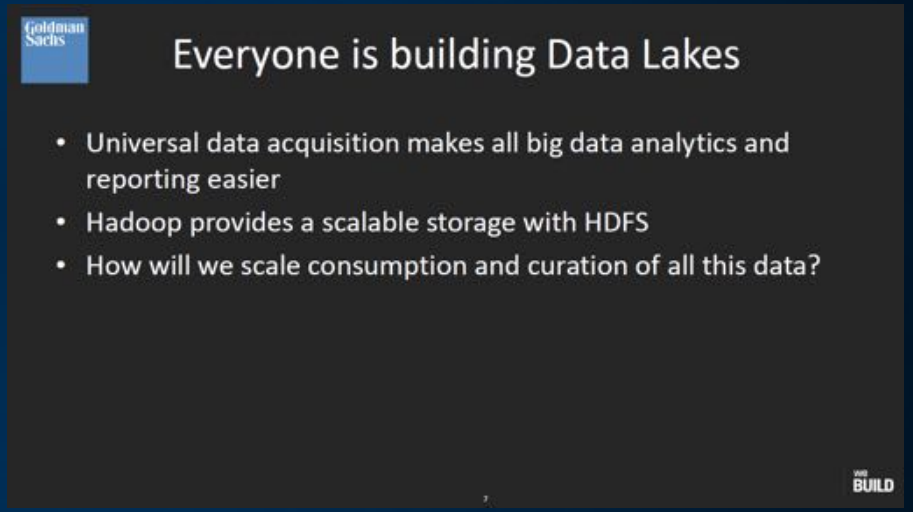
Trend #1: Data Scale Driving Deep Learning Process



"Machine Learning Yearning", Andrew Ng, 2016

Trend #2: Hadoop Becoming the Center of Data Gravity

Phillip Radley, BT Group
Strata + Hadoop World 2016 San Jose



Matthew Glickman, Goldman Sachs
Spark Summit East 2015

Trend #3: Real-World ML/DL Systems re Complex Big Data Analytics Pipelines

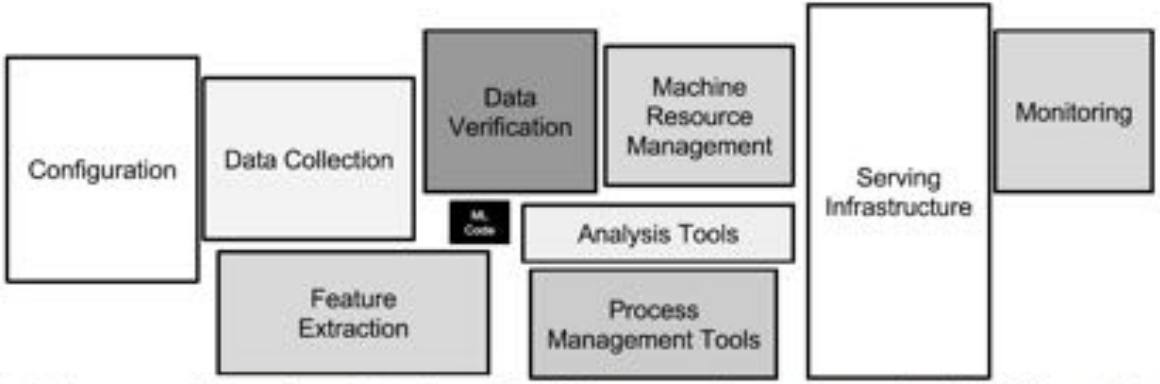


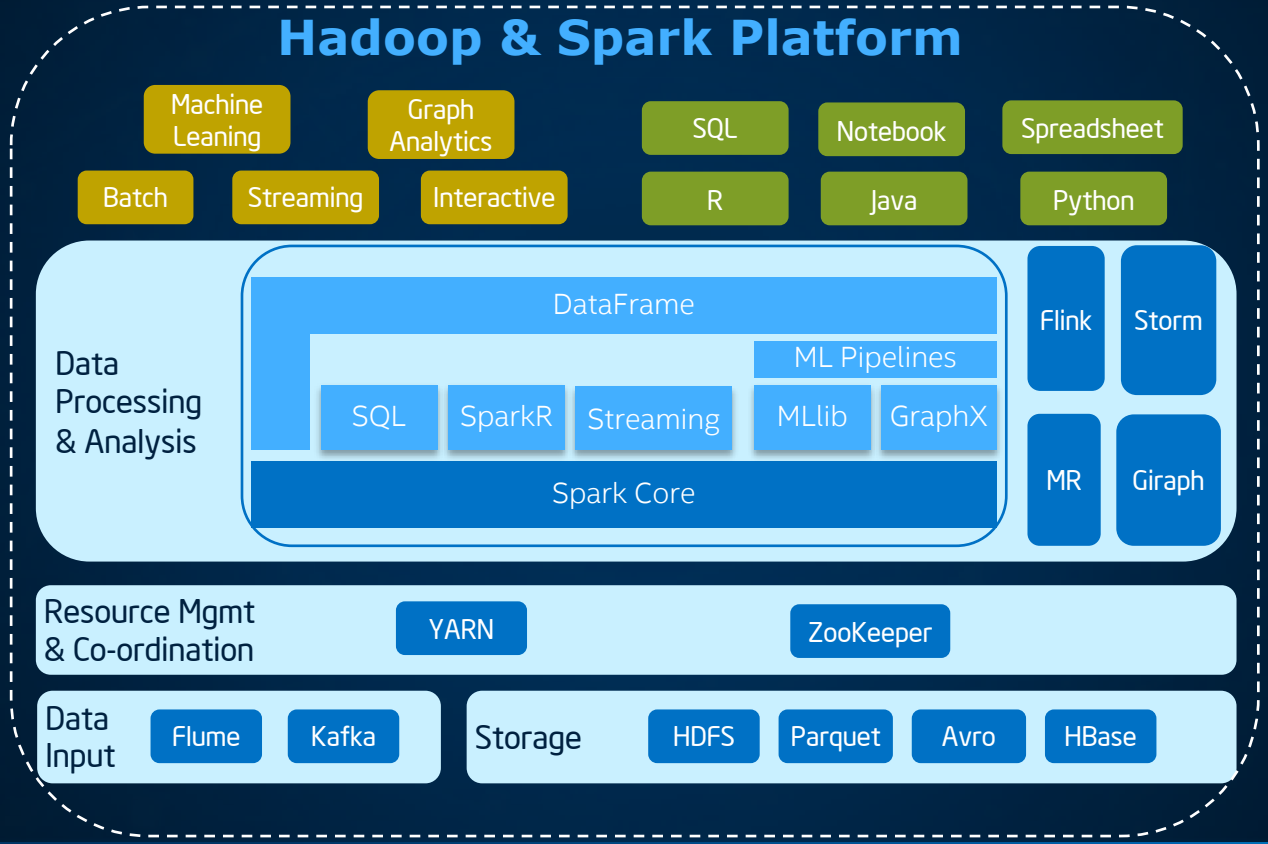
Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

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

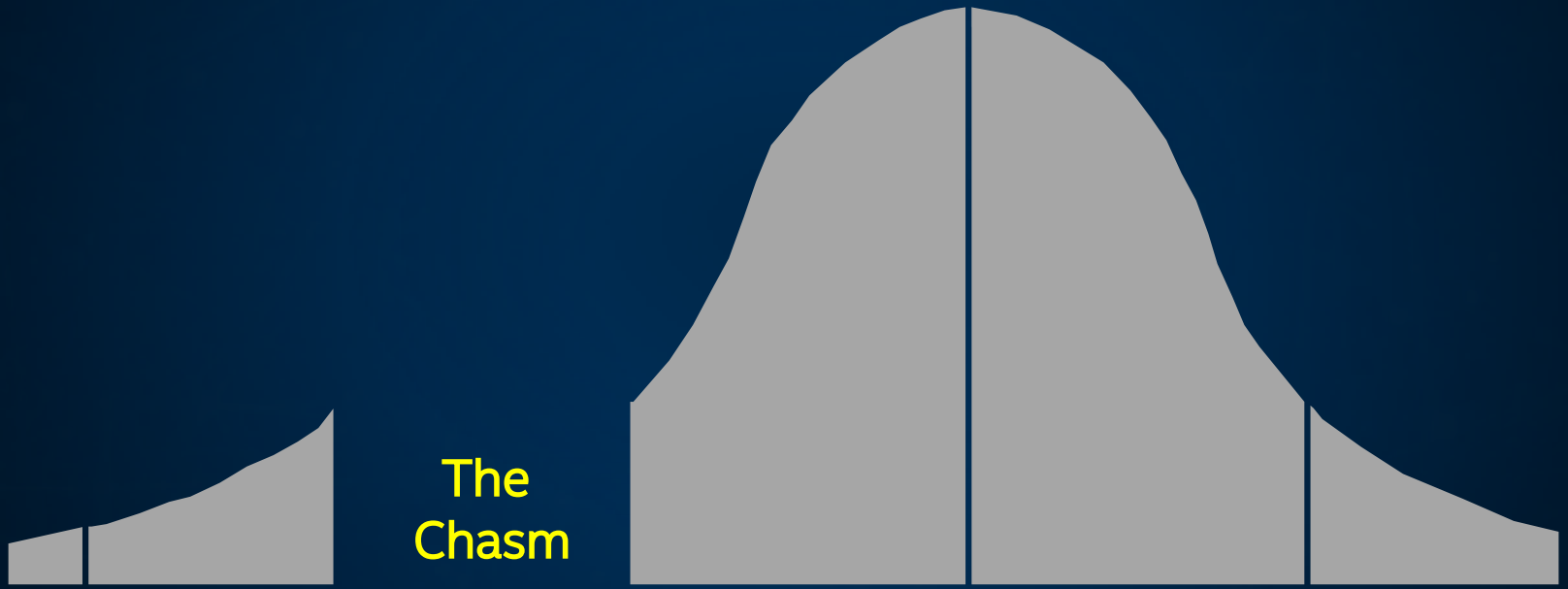
Trend #4: Unified Big Data Platform Driving Analytics & Data Science

Ion Stoica, UC Berkeley,
Spark Summit 2013 Keynote

Unified Big Data Analytics Platform



Chasm b/w Deep Learning and Big Data Communities



Deep learning experts

Average users (Big Data users, data scientists, analysts, etc.)

Bridging the Chasm

Make deep learning more accessible to big data and data science communities

- **Continue the use of familiar SW tools and HW infrastructure to build deep learning applications**
- **Analyze "big data" using deep learning on the same Hadoop/Spark cluster where the data are stored**
- **Add deep learning functionalities to the Big Data (Spark) programs and/or workflow**
- **Leverage existing Hadoop/Spark clusters to run deep learning applications**
  - **Shared with other workloads (*e.g.*, *ETL*, *data warehouse*, *feature engineering*, *statistic machine learning*, *graph analytics*, *etc.*) in a dynamic and elastic fashion**

AI Is Integrating with Big DATA

AI Models Training And Inference
On Existing Big DATA Clusters

Apache Spark Emerging as the Unified Analytics + AI Platform

AI on Apache Spark

High-Performance
Deep Learning Framework
for Apache Spark

software.intel.com/bigdl

Analytics + AI Pipelines
for Apache Spark and BigDL
Reference Use Cases, AI Models,
High-level APIs, Feature Engineering, etc.

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

Unifying Analytics + AI on Apache Spark

BigDL

Apache Spark recap, BigDL overview

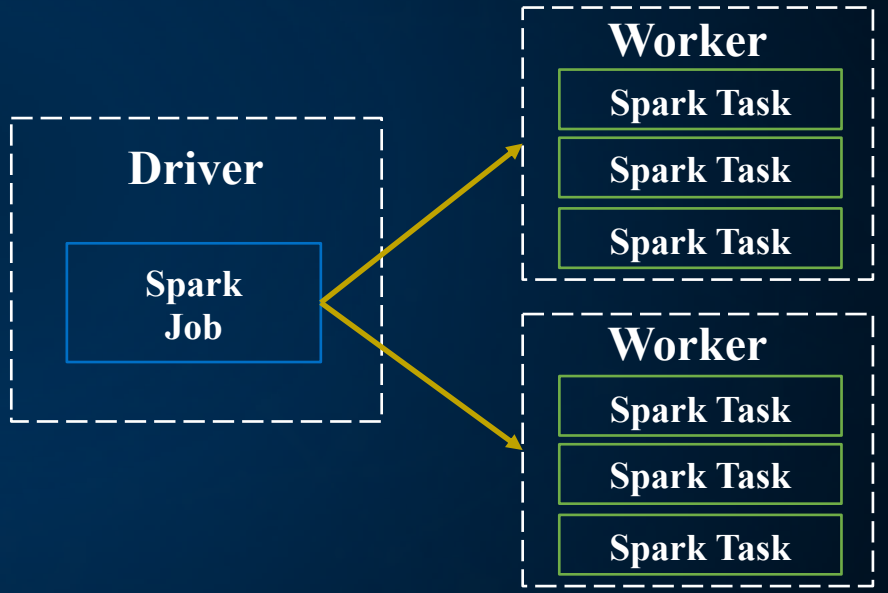
Apache Spark overview

- Apache Spark is a fast and general engine for large-scale data processing.
- Originally developed in the AmpLab at UC Berkeley in 2009. Later donated to the Apache Software Foundation.
- Largest open source project in data processing and became one of the key big data distributed processing frameworks in the world.
- Speed - Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. Apache Spark has an advanced DAG execution engine that supports acyclic data flow and in-memory computing.
- Ease of Use - Write applications quickly in Java, Scala, Python, R.
- Unified Engine - Spark comes packaged with higher-level libraries, including support for SQL queries, streaming data, machine learning and graph processing.

Apache Spark

Low Latency, Distributed Data Processing Framework

- A Spark cluster consists of a single *driver* node and multiple *worker* nodes
- A Spark *job* contains many Spark *tasks*, each working on a data *partition*
- Driver is responsible for scheduling and dispatching the tasks to workers, which runs the actual Spark tasks

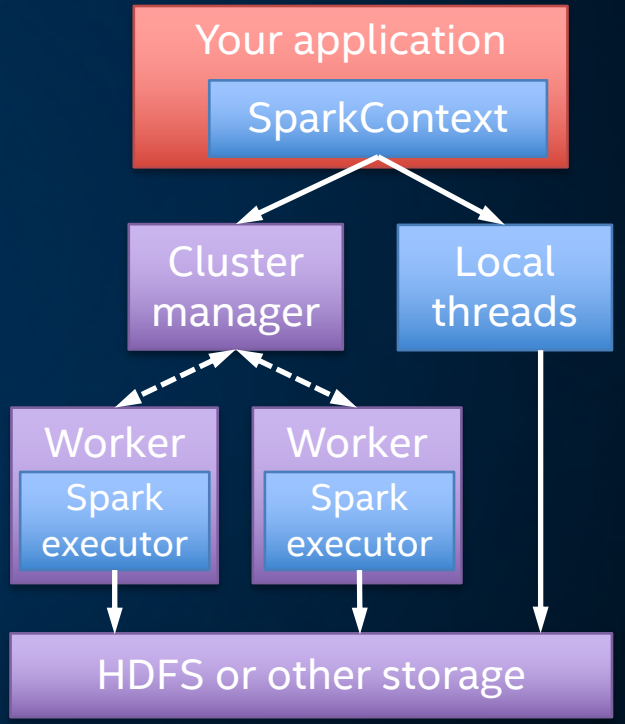


https://spark.apache.org

Apache Spark

Spark Program

- Spark runs as a library in your program (1 instance per app)
- Runs tasks locally or on cluster
  - K8s, YARN, Mesos or standalone mode
- Accesses storage systems via Hadoop InputFormat API
  - Can use HBase, HDFS, S3, …

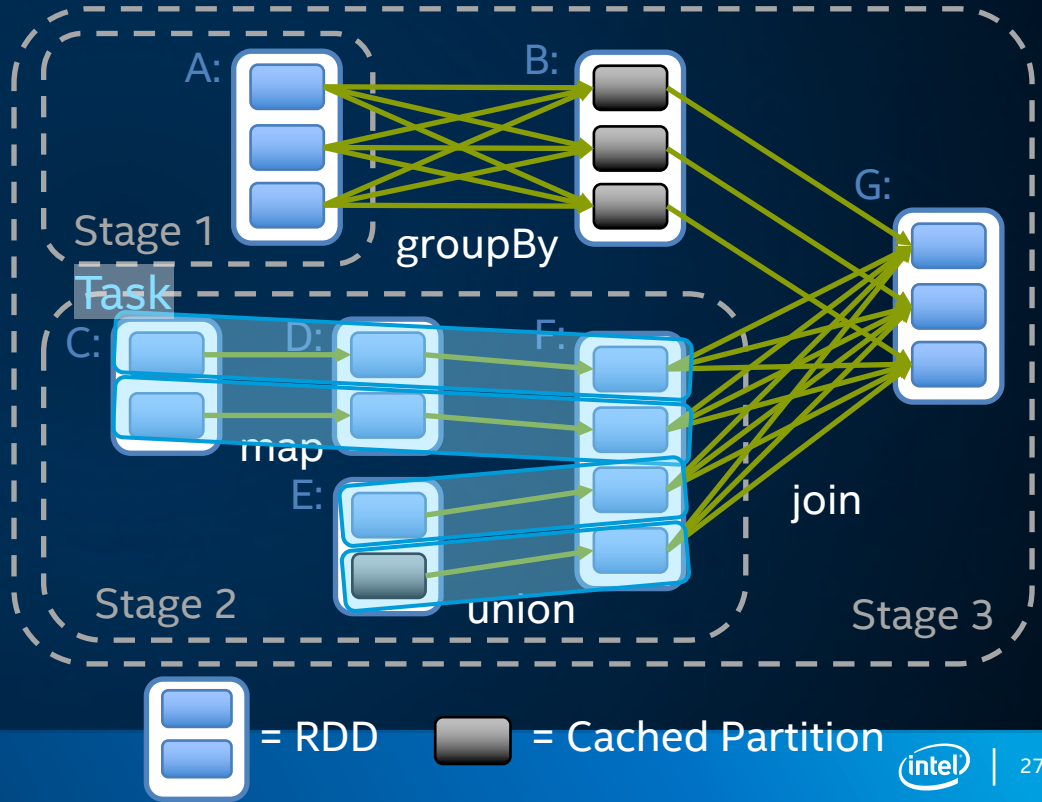


Source: "Parallel programming with Spark", Matei Zaharia, AMPCamp 3

Apache Spark

Distributed Task Execution

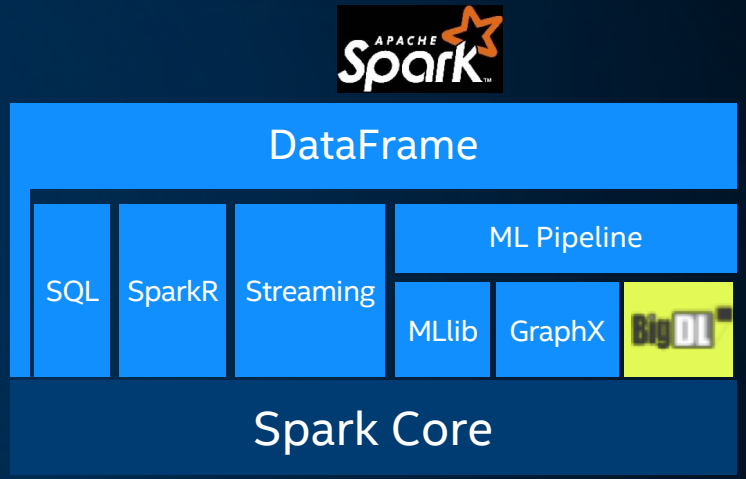
- General task graphs
- Automatically pipelines functions
- Data locality aware
- Partitioning aware to avoid shuffles



Source: "Parallel programming with Spark", Matei Zaharia, AMPCamp 3

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

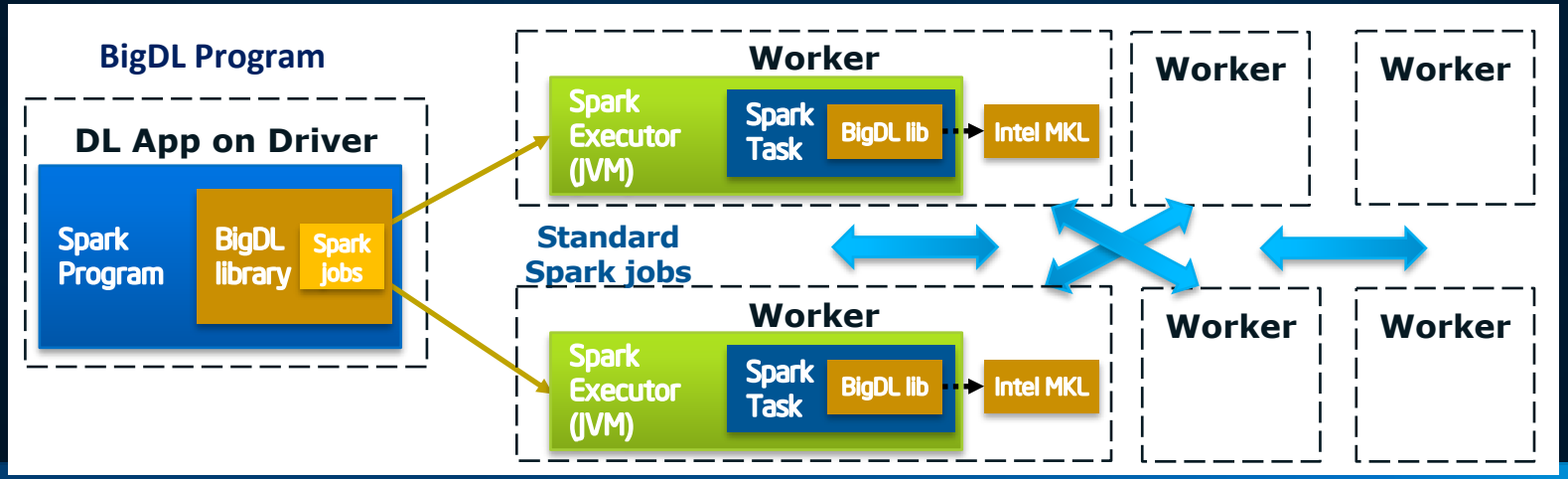
https://bigdl-project.github.io/

Run as standard Apache Spark Program

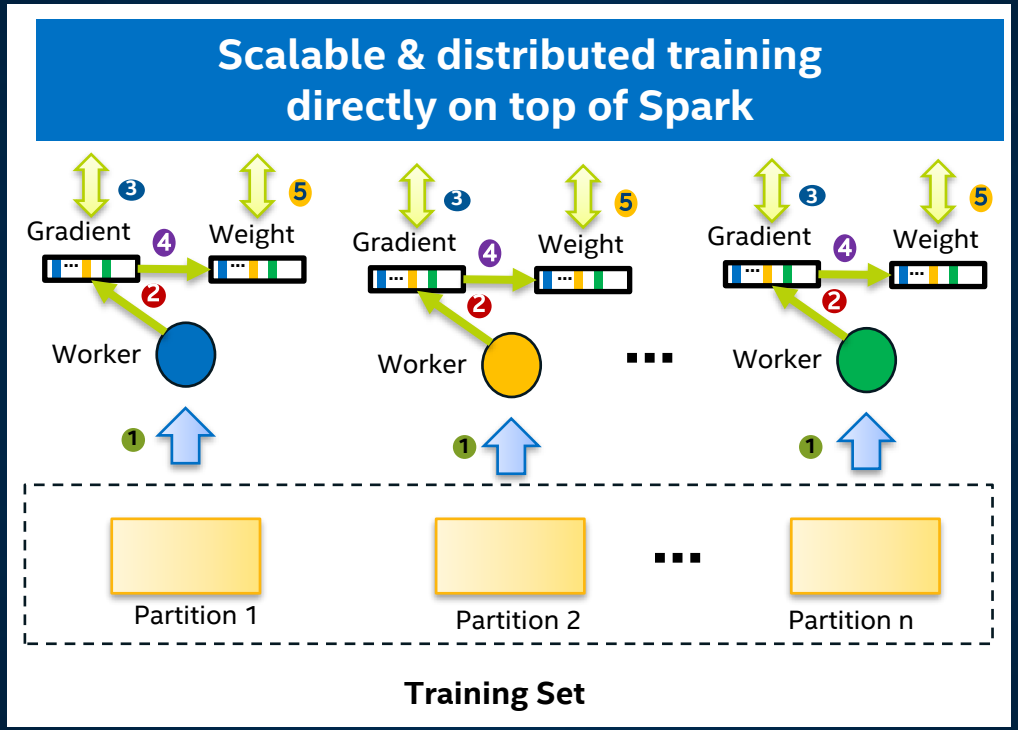
Data parallel
Each Spark task runs the same model on a subset of the data (batch)

Iterative
Each iteration of the training runs as a Spark job

Standard Spark jobs
No changes to the Spark or Hadoop clusters needed



Parameter Synchronization in BigDL



Peer-2-Peer All-Reduce synchronization

Analytics Zoo

Analytics + AI platform for Apache Spark & BigDL

Analytics Zoo

Build E2E Analytics & AI Applications for Big Data at Scale

Unified Big Data Analytics + AI Open Source Platform

Reference Use Cases	▪ Anomaly detection, sentiment analysis, fraud detection, image generation, chatbot, sequence prediction, etc.
Built-In Deep Learning Models	▪ Image classification, object detection, text classification, recommendations, GANs, Sequence to Sequence, etc.
Feature Engineering	Feature transformations for ▪ Image, text, 3D imaging, time series, speech, etc.
High-Level Pipeline APIs	▪ Distributed Tensorflow & Keras on Spark/BigDL ▪ Support for autograd, transfer learning, Spark DataFrame and ML Pipeline ▪ Model serving API for model serving/inference pipelines
Backends	Apache Spark, TensorFlow*, Keras*, BigDL, etc.

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

https://analytics-zoo.github.io/

Analytics Zoo

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

- E2E analytics + AI pipelines (natively in Spark DataFrames and ML Pipelines) using ***nnframes***
- Flexible model definition using ***autograd, Keras-style & transfer learning APIs***
- Data preprocessing using built-in ***feature engineering operations***
- Out-of-the-box solutions for a variety of problem types using ***built-in deep learning models and reference use cases***

Productionize deep learning applications for big data at scale

- Serving models in web services and big data frameworks (e.g., Storm or Kafka) using POJO ***model serving APIs***
- Large-scale distributed TensorFlow model inference using ***TFNet***

Analytics Zoo

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

- **E2E analytics + AI pipelines (natively in Spark DataFrames and ML Pipelines) using *nnframes***
- *Flexible model definition using autograd, Keras-style & transfer learning APIs*
- *Data preprocessing using built-in feature engineering operations*
- *Out-of-the-box solutions for a variety of problem types using built-in deep learning models and reference use cases*

Productionize deep learning applications at scale for big data

- *Serving models in web services and big data frameworks (e.g., Storm or Kafka) using POJO model serving APIs*
- *Large-scale distributed TensorFlow model inference using TFNet*

nnframes

Native DL support in Spark DataFrames and ML Pipelines

1. **Initialize *NNContext* and load images into *DataFrames* using *`NNImageReader`***

```
from zoo.common.nncontext import *
from zoo.pipeline.nnframes import *
sc = init_nncontext()
imageDF = NNImageReader.readImages(image_path, sc)
```

2. **Process loaded data using *`DataFrame`* transformations**

```
getName = udf(lambda row: ...)
df = imageDF.withColumn("name", getName(col("image")))
```

3. **Processing image using built-in *feature engineering* operations**

```
from zoo.feature.image import *
transformer = ChainedPreprocessing(
        [RowToImageFeature(), ImageChannelNormalize(123.0, 117.0, 104.0),
         ImageMatToTensor(), ImageFeatureToTensor()])
```

nnframes

Native DL support in Spark DataFrames and ML Pipelines

4. Define model using *Keras-style API*

```
from zoo.pipeline.api.keras.layers import *
from zoo.pipeline.api.keras.models import *
model = Sequential()
          .add(Convolution2D(32, 3, 3, activation='relu', input_shape=(1, 28, 28))) \
          .add(MaxPooling2D(pool_size=(2, 2))) \
          .add(Flatten()).add(Dense(10, activation='softmax')))
```

5. Train model using *Spark ML Pipelines*

```
Estimater = NNEstimater(model, CrossEntropyCriterion(), transformer) \
                .setLearningRate(0.003).setBatchSize(40).setMaxEpoch(1) \
                .setFeaturesCol("image").setCachingSample(False)
nnModel = estimater.fit(df)
```

Analytics Zoo

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

- *E2E analytics + AI pipelines (natively in Spark DataFrames and ML Pipelines) using nnframes*
- **Flexible model definition using *autograd, Keras-style & transfer learning APIs***
- *Data preprocessing using built-in feature engineering operations*
- *Out-of-the-box solutions for a variety of problem types using built-in deep learning models and reference use cases*

Productionize deep learning applications at scale for big data

- *Serving models in web services and big data frameworks (e.g., Storm or Kafka) using POJO model serving APIs*
- *Large-scale distributed TensorFlow model inference using TFNet*

Autograd, Keras &Transfer Learning APIs

1. **Use transfer learning APIs to**
   - **Load an existing Caffe model**
   - **Remove last few layers**
   - **Freeze first few layers**
   - **Append a few layers**

```
from zoo.pipeline.api.net import *
full_model = Net.load_caffe(def_path, model_path)
# Remove layers after pool5
model = full_model.new_graph(outputs=["pool5"])
# freeze layers from input to res4f inclusive
model.freeze_up_to(["res4f"])
# append a few layers
image = Input(name="input", shape=(3, 224, 224))
resnet = model.to_keras()(image)
resnet50 = Flatten()(resnet)
```

Build Siamese Network Using Transfer Learning

Autograd, Keras &Transfer Learning APIs

2. Use *autograd and Keras-style* APIs to build the Siamese Network

```
import zoo.pipeline.api.autograd as A
from zoo.pipeline.api.keras.layers import *
from zoo.pipeline.api.keras.models import *

input = Input(shape=[2, 3, 226, 226])
features = TimeDistributed(layer=resnet50)(input)
f1 = features.index_select(1, 0) #image1
f2 = features.index_select(1, 1) #image2
diff = A.abs(f1 - f2)
fc = Dense(1)(diff)
output = Activation("sigmoid")(fc)
model = Model(input, output)
```

Build Siamese Network Using Transfer Learning

Analytics Zoo

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

- *E2E analytics + AI pipelines (natively in Spark DataFrames and ML Pipelines) using nnframes*
- *Flexible model definition using autograd, Keras-style & transfer learning APIs*
- **Data preprocessing using built-in *feature engineering operations***
- *Out-of-the-box solutions for a variety of problem types using built-in deep learning models and reference use cases*

Productionize deep learning applications at scale for big data

- *Serving models in web services and big data frameworks (e.g., Storm or Kafka) using POJO model serving APIs*
- *Large-scale distributed TensorFlow model inference using TFNet*

Feature Engineering

1. **Read images into local or distributed *ImageSet***

```
from zoo.common.nncontext import *
from zoo.feature.image import *
spark = init_nncontext()
local_image_set = ImageSet.read(image_path)
distributed_image_set = ImageSet.read(image_path, spark, 2)
```

2. **Image augmentations using built-in *ImageProcessing* operations**

```
transformer = ChainedPreprocessing([ImageBytesToMat(),
                    ImageColorJitter(),
                    ImageExpand(max_expand_ratio=2.0),
                    ImageResize(300, 300, -1),
                    ImageHFlip()])
new_local_image_set = transformer(local_image_set)
new_distributed_image_set = transformer(distributed_image_set)
```

Image Augmentations Using Built-in Image Transformations (w/ OpenCV on Spark)

Analytics Zoo

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

- *E2E analytics + AI pipelines (natively in Spark DataFrames and ML Pipelines) using nnframes*
- *Flexible model definition using autograd, Keras-style & transfer learning APIs*
- *Data preprocessing using built-in feature engineering operations*
- **Out-of-the-box solutions for a variety of problem types using *built-in deep learning models and reference use cases***

Productionize deep learning applications at scale for big data

- *Serving models in web services and big data frameworks (e.g., Storm or Kafka) using POJO model serving APIs*
- *Large-scale distributed TensorFlow model inference using TFNet*

Built-in Deep Learning Models

- *Object detection API*
  - High-level API and pretrained models (e.g., SSD, Faster-RCNN, etc.) for object detection
- *Image classification API*
  - High-level API and pretrained models (e.g., VGG, Inception, ResNet, MobileNet, etc.) for image classification
- *Text classification API*
  - High-level API and pre-defined models (using CNN, LSTM, etc.) for text classification
- *Recommendation API*
  - High-level API and pre-defined models (e.g., Neural Collaborative Filtering, Wide and Deep Learning, etc.) for recommendation

Object Detection API

1. **Load pretrained model in *Detection Model Zoo***

```
from zoo.common.nncontext import *
from zoo.models.image.objectdetection import *
spark = init_nncontext()
model = ObjectDetector.load_model(model_path)
```

2. **Off-the-shell inference using the loaded model**

```
image_set = ImageSet.read(img_path, spark)
output = model.predict_image_set(image_set)
```

3. **Visualize the results using utility methods**

```
config = model.get_config()
visualizer = Visualizer(config.label_map(), encoding="jpg")
visualized = visualizer(output).get_image(to_chw=False).collect()
```

Off-the-shell Inference Using Analytics Zoo Object Detection API

https://github.com/intel-analytics/analytics-zoo/tree/master/pyzoo/zoo/examples/objectdetection

Reference Use Cases

- ***Anomaly Detection***
  - **Using LSTM network to detect anomalies in time series data**
- ***Fraud Detection***
  - **Using feed-forward neural network to detect frauds in credit card transaction data**
- ***Recommendation***
  - **Use Analytics Zoo Recommendation API (i.e., Neural Collaborative Filtering, Wide and Deep Learning) for recommendations on data with explicit feedback.**
- ***Sentiment Analysis***
  - **Sentiment analysis using neural network models (e.g. CNN, LSTM, GRU, Bi-LSTM)**
- ***Variational Autoencoder (VAE)***
  - **Use VAE to generate faces and digital numbers**

https://github.com/intel-analytics/analytics-zoo/tree/master/apps

Analytics Zoo

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

- *E2E analytics + AI pipelines (natively in Spark DataFrames and ML Pipelines) using nnframes*
- *Flexible model definition using autograd, Keras-style & transfer learning APIs*
- *Data preprocessing using built-in feature engineering operations*
- *Out-of-the-box solutions for a variety of problem types using built-in deep learning models and reference use cases*

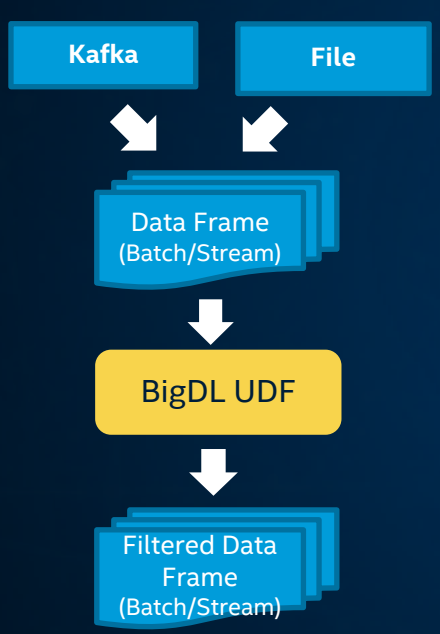
Productionize deep learning applications at scale for big data

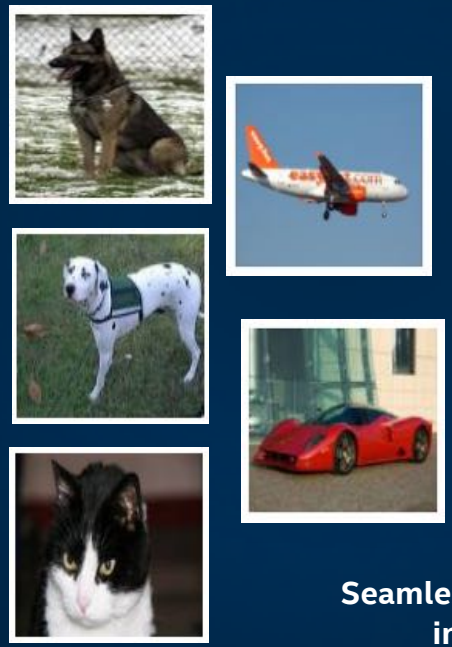
- **Serving models in web services and big data frameworks (e.g., Storm or Kafka) using POJO *model serving APIs***
- *Large-scale distributed TensorFlow model inference using TFNet*

POJO Model Serving API

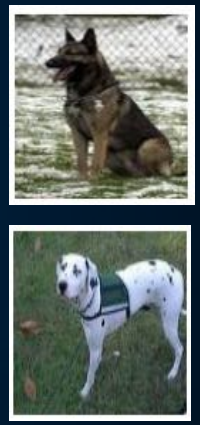
```
import com.intel.analytics.zoo.pipeline.inference.AbstractInferenceModel;

public class TextClassification extends AbstractInferenceModel {
  public RankerInferenceModel(int concurrentNum) {
    super(concurrentNum);
  }
  ...
}

public class ServingExample {
  public static void main(String[] args) throws IOException {
    TextClassification model = new TextClassification();
    model.load(modelPath, weightPath);

    texts = …
    List<JTensor> inputs = preprocess(texts);
    for (JTensor input : inputs) {
      List<Float> result = model.predict(input.getData(), input.getShape());
      ...
  }
}
```

Integration with Spark SQL, DataFrames and Structure Streaming





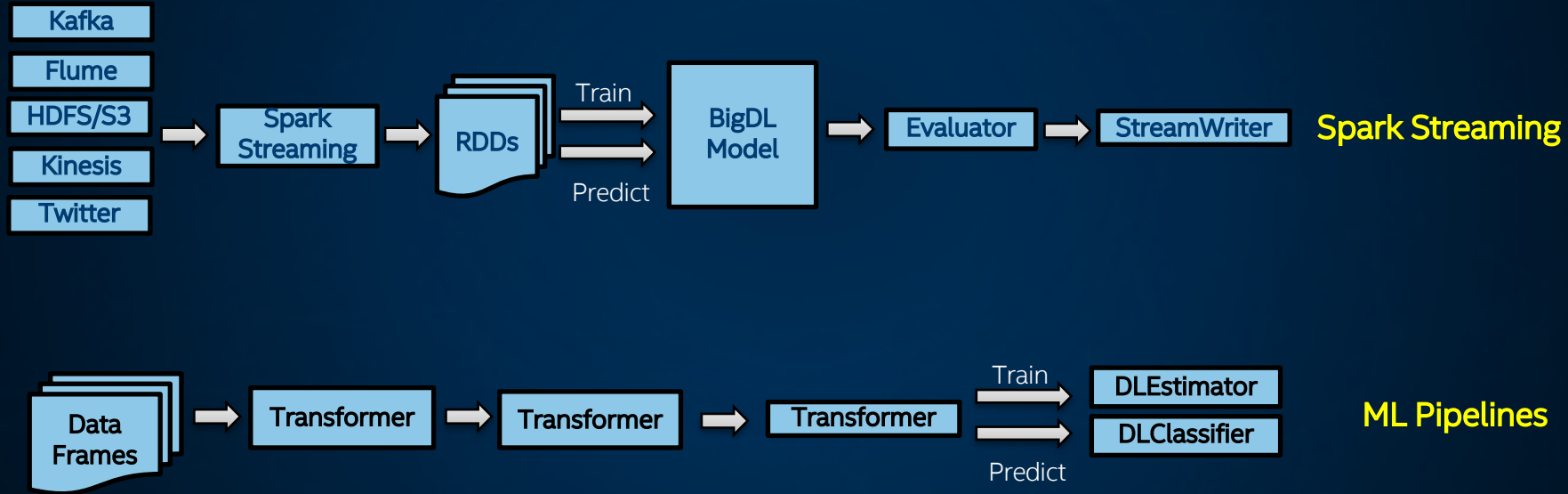
```
df.select($'image')
  .withColumn( "image_type",
     ImgClassifyInBigDL("image"))
  .filter($'image_type' == 'dog')
```



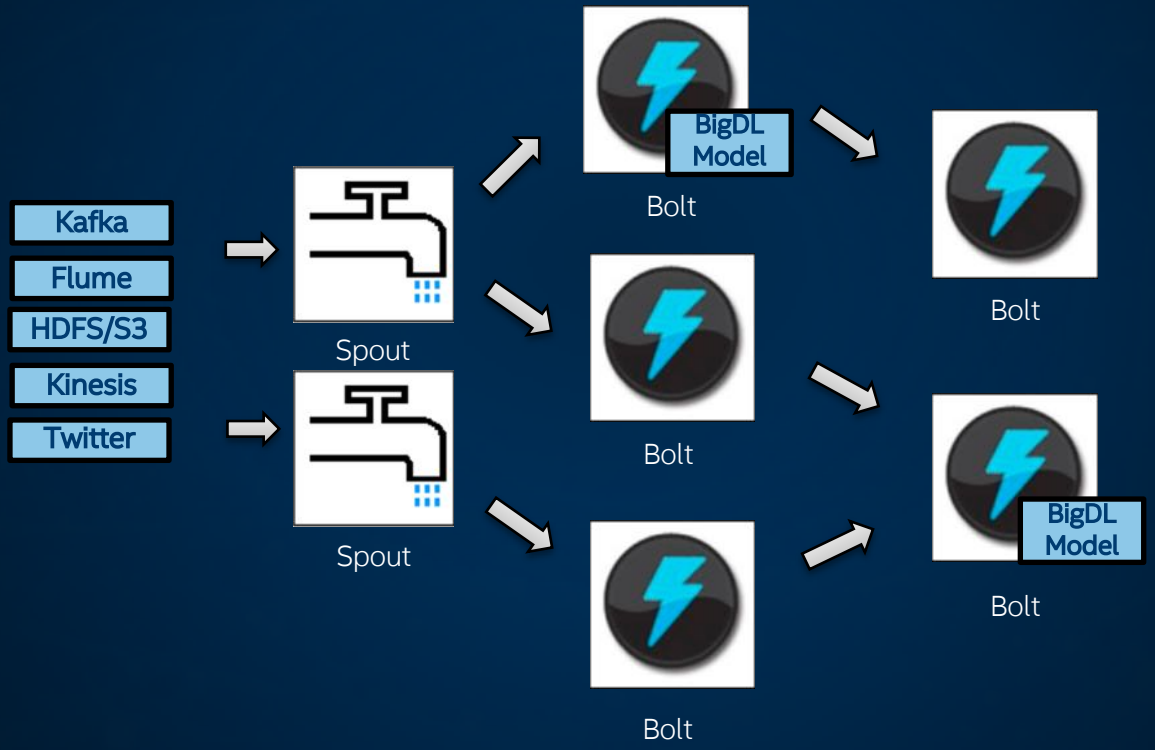
Seamless support of deep learning functionalities in *SQL queries* and *stream processing*

ImageNet dataset (http://www.image-net.org)

Integration with Spark Streaming and ML Pipelines



Integration with Big Data Ecosystem



Analytics Zoo

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

- *E2E analytics + AI pipelines (natively in Spark DataFrames and ML Pipelines) using nnframes*
- *Flexible model definition using autograd, Keras-style & transfer learning APIs*
- *Data preprocessing using built-in feature engineering operations*
- *Out-of-the-box solutions for a variety of problem types using built-in deep learning models and reference use cases*

Productionize deep learning applications at scale for big data

- *Serving models in web services and big data frameworks (e.g., Storm or Kafka) using POJO model serving APIs*
- **Large-scale distributed TensorFlow model inference & fine tuning using *TFNet***

Distributed TensorFlow Model Inference

1. **Export TensorFlow models (in your TensorFlow program)**

```
import tensorflow as tf

batch_size_tensor = tf.placeholder_with_default(128, shape=[])
x_batch, y_batch = tf.train.shuffle_batch (..., batch_size=batch_size_tensor, ...)
cnn = cnn_model_fn(x_batch, y_batch)

sess = tf.Session()
init_op = tf.group(tf.global_variables_initializer(), tf.local_variables_initializer())
sess.run(init_op)
for step in range(600):
    _, loss = sess.run([cnn.train_op, cnn.loss], ...)

from zoo.utils.tf import *
export_tf(sess, folder_path, [x_batch], [cnn.prediction])
```

Distributed TensorFlow Model Inference

2. Load exported TensorFlow model into Analytics Zoo

```
from zoo.pipeline.api.net import *
model = TFNet.from_export_folder(folder_path)

# Alternatively, you may directly load a frozen TensorFlow model as follows
# model = TFNet(model_path, ["image_tensor:0"], ["output_tensor:0"])
```

3. Add a few layers and run distributed model inference

```
import zoo.pipeline.api.autograd as A
from zoo.pipeline.api.keras.layers import *
from zoo.pipeline.api.keras.models import *

input = Input(shape=[2, 3, 226, 226])
features = TimeDistributed(layer=model)(input)
f1 = features.index_select(1, 0)
f2 = features.index_select(1, 1)
diff = A.abs(f1 - f2)
result = Model(input, diff)
result.predict_image_set(...)
```

Integration With TensorBoard for Visualization

TensorBoard integration for visualizing BigDL program behaviors

https://bigdl-project.github.io/master/#ProgrammingGuide/visualization/

Models Interoperability Support

- Load existing TensorFlow, Keras, Caffe, Torch Model
  - Useful for inference and model fine-tuning
  - Allows for transition from single-node for distributed application deployment
  - Allows for model sharing between data scientists and production engineers

Analytics Zoo Examples

Transfer Learning, Object detection, TFNet

Transfer Learning

Notebook:

https://github.com/intel-analytics/analytics-zoo/blob/master/apps/dogs-vs-cats/transfer-learning.ipynb

Object detection API

Notebook:

https://github.com/intel-analytics/analytics-zoo/blob/master/apps/object-detection/object-detection.ipynb

Image Classification using TFNet

Notebook:

https://github.com/intel-analytics/analytics-zoo/blob/master/apps/tfnet/image_classification_inference.ipynb

Break

Notices and Disclaimers

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

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.

The cost reduction scenarios described are intended to enable you to get a better understanding of how the purchase of a given Intel based product, combined with a number of situation-specific variables, might affect future costs and savings. Circumstances will vary and there may be unaccounted-for costs related to the use and deployment of a given product. Nothing in this document should be interpreted as either a promise of or contract for a given level of costs or cost reduction.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

No computer system can be absolutely secure.

Intel® Advanced Vector Extensions (Intel® AVX)* provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

Intel processors of the same SKU may vary in frequency or power as a result of natural variability in the production process.