ANAYLTICS-ZOO TUTORIAL

Zhichao Li(zhichao.li@intel.com)

Data Analytics Technologies, Software and Service Group, Intel

About me

Software Architect at Intel. Contributor of Spark, BigDL and Analytics-zoo

Focusing area

- Large scale machine learning, deep learning implementation and optimization
- Machine learning / deep learning applications on big data

Agenda

Analytics-zoo basics

- Keras support
- Hands-on practice

Hands-on practice

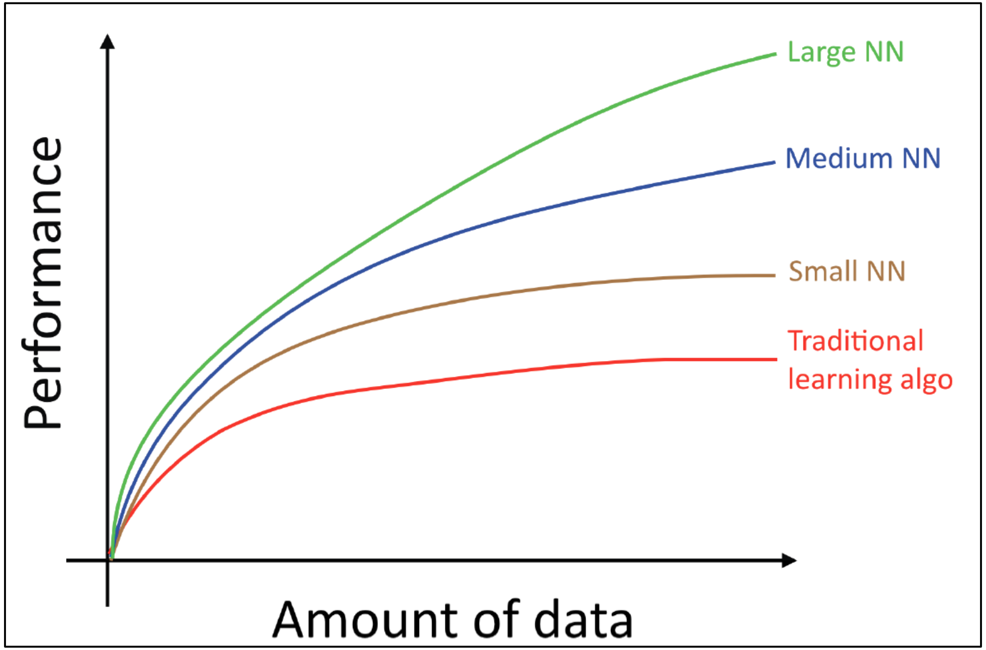
- Customer case
- Pre-trained ResNet
- Anomaly detection
- Recommendation (NCF wide and deep)
- VAE

https://github.com/zhichao-li/tzoo

ANALYTICS-ZOO INTRODUCTION

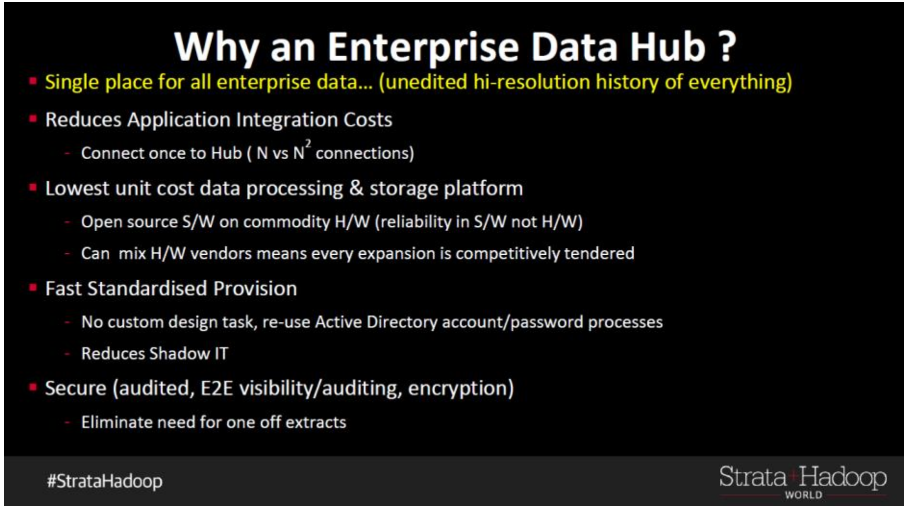
Motivations

Trend #1: Data Scale Driving Deep Learning Process

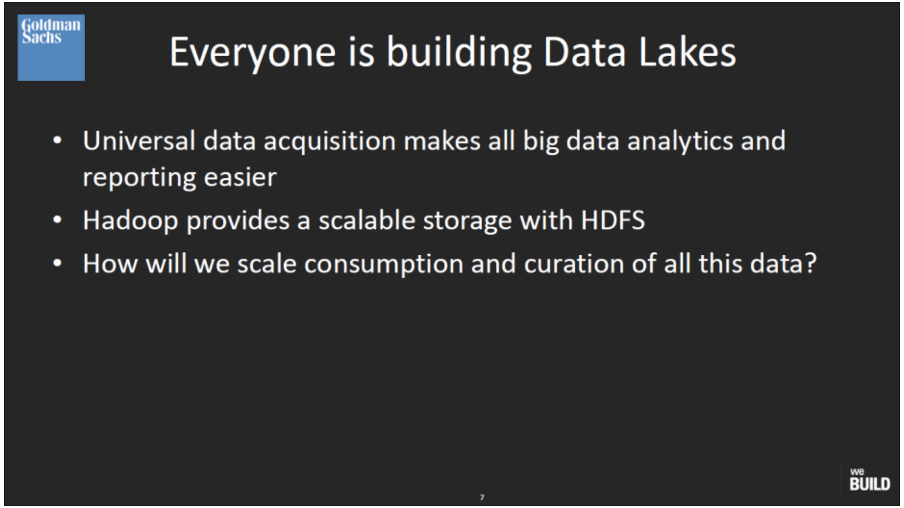


"Machine Learning Yearning", Andrew Ng, 2016

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 Are Complex Big Data Analytics Pipelines

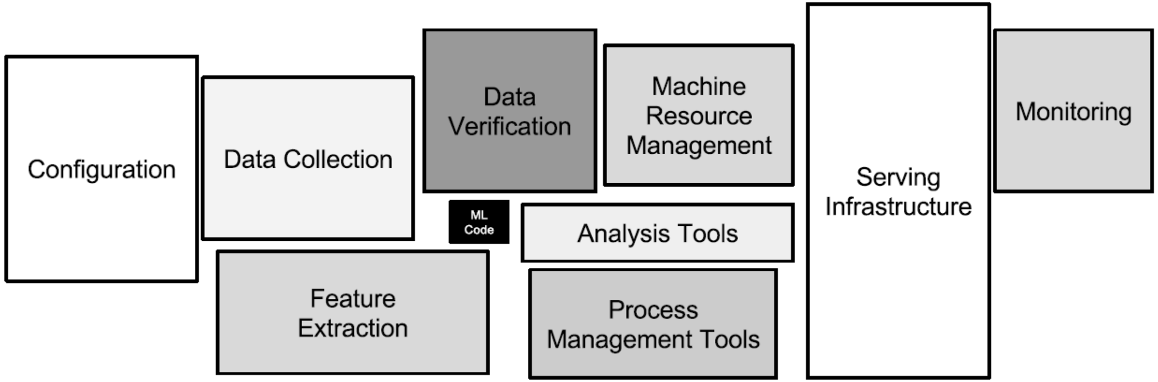
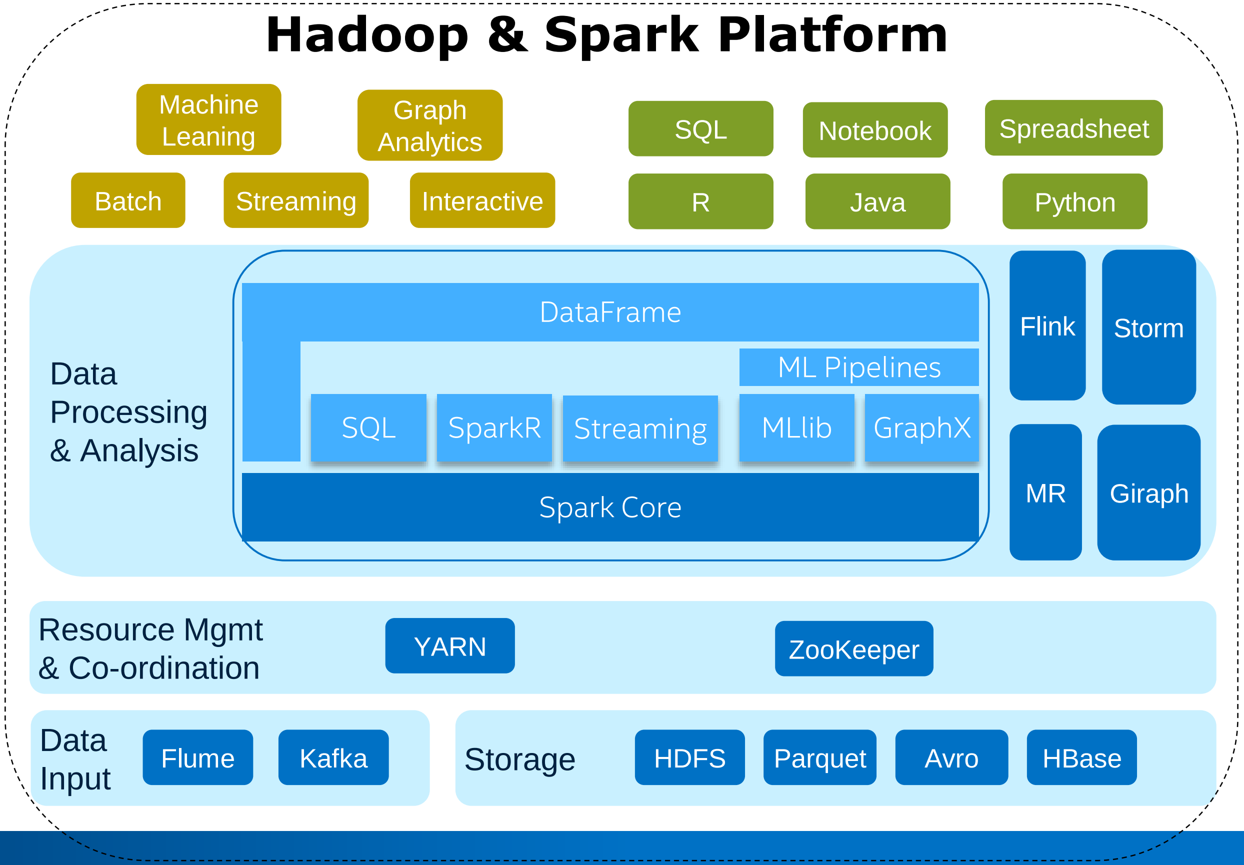


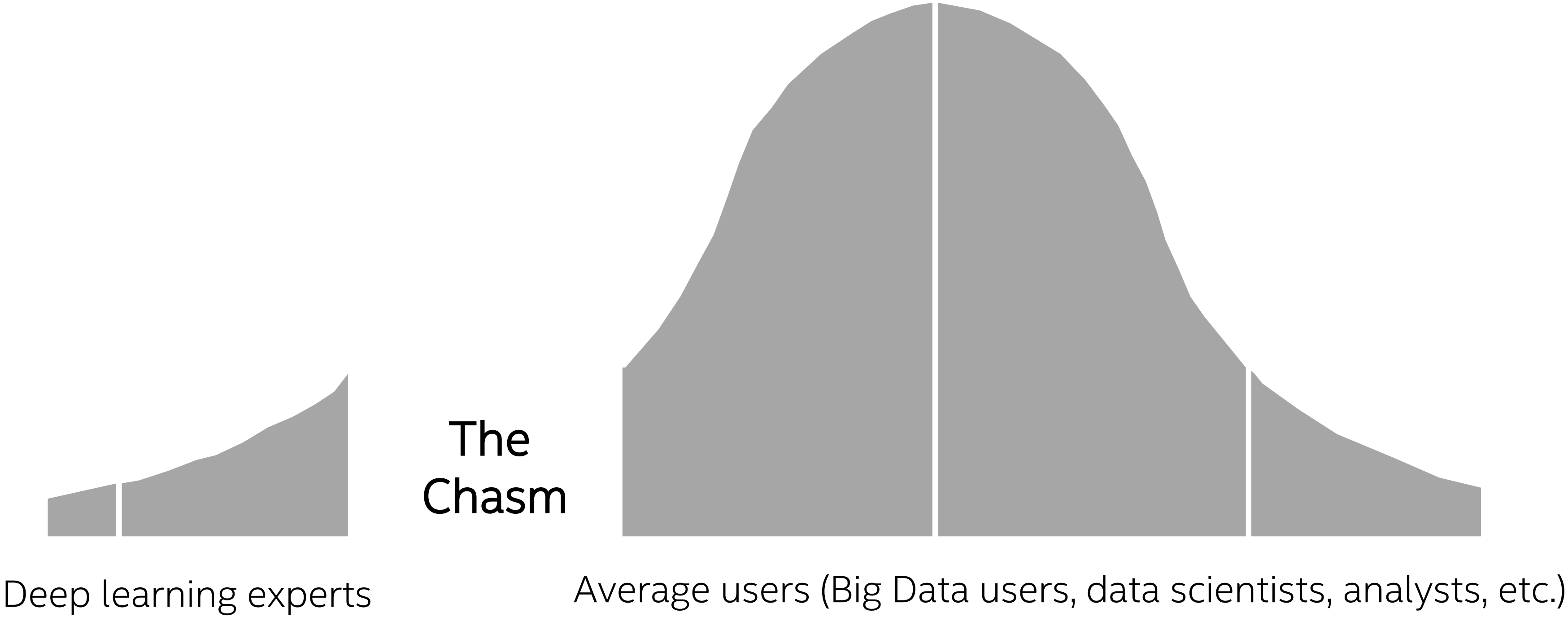
Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

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

Unified Big Data Analytics Platform



Chasm b/w Deep Learning and Big Data Communities

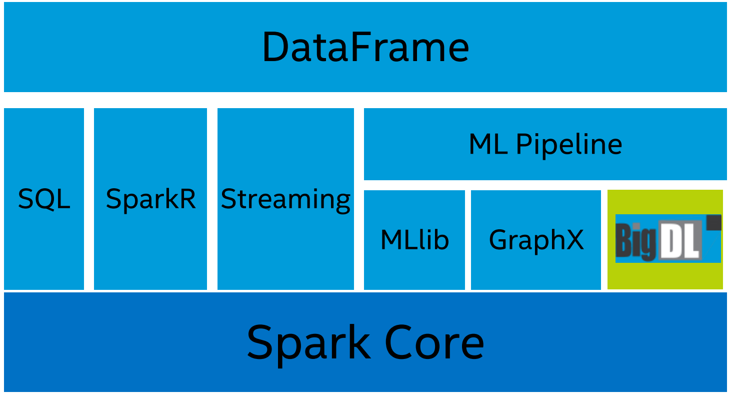


Overview

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
  - 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/

Model Zoo

Image Classification

- Inception
- Resnet
- VGG
- MobileNet
- Alexnet
- DenseNet
- SqueezeNet

Object Detection

- SSD (Single Shot Multibox Detector)
    - VGG
    - MobileNet
- Faster-RCNN
    - VGG
    - PvaNet

Analytics Zoo

Analytics + AI Pipelines for Spark and BigDL

"Out-of-the-box" ready for use

- **Reference use cases**
  - Fraud detection, anomaly detection, chatbot, sequence prediction, sentiment analysis, etc.
- **Predefined models**
  - Object detection, image classification, text classification, recommendations, GAN, etc.
- **Feature engineering & transformations**
  - Image, text, speech, 3D imaging, time-series, etc.
- **High level pipeline APIs**
  - Dataframes, ML Pipelines, Keras/Keras2, autograd, etc.

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

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

Analytics-zoo run as Standard Spark Programs

Standard Spark jobs

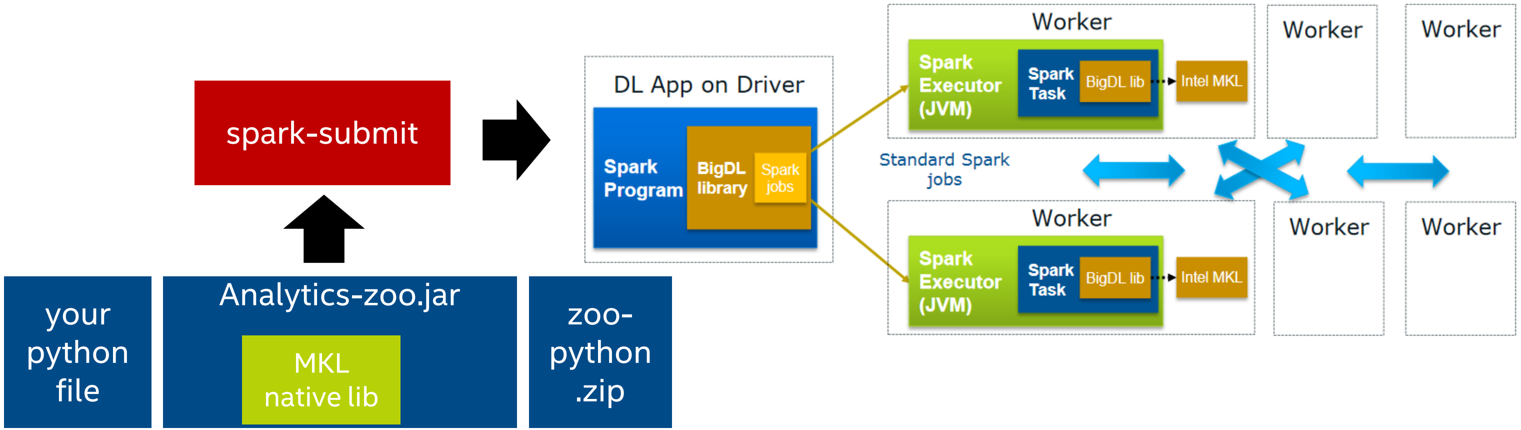
- **No changes to the Spark or Hadoop clusters needed**

Iterative

- **Each iteration of the training runs as a Spark job**

Data parallel

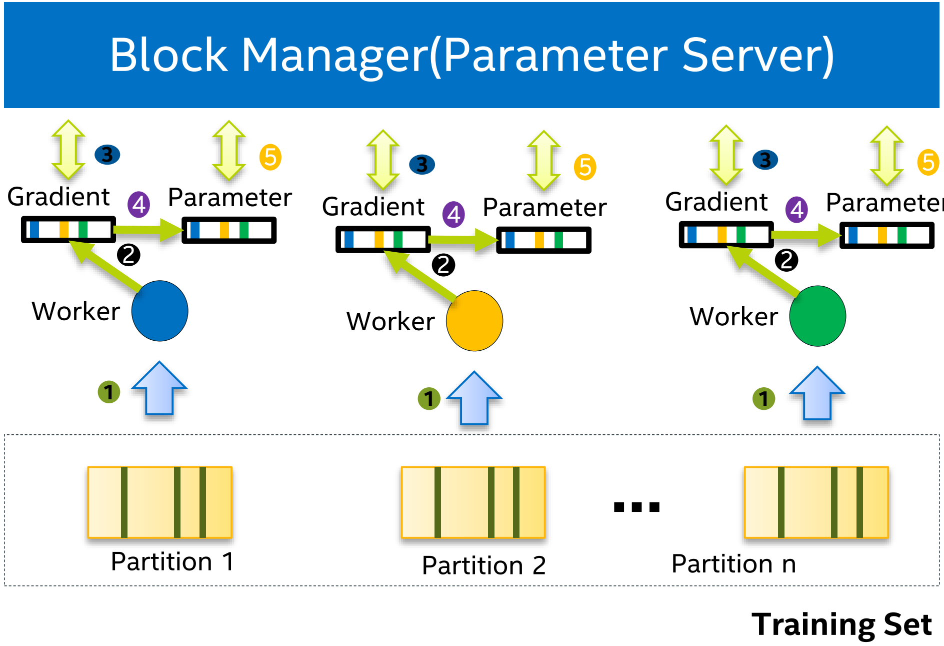
- **Each Spark task runs the same model on a subset of the data (batch)**



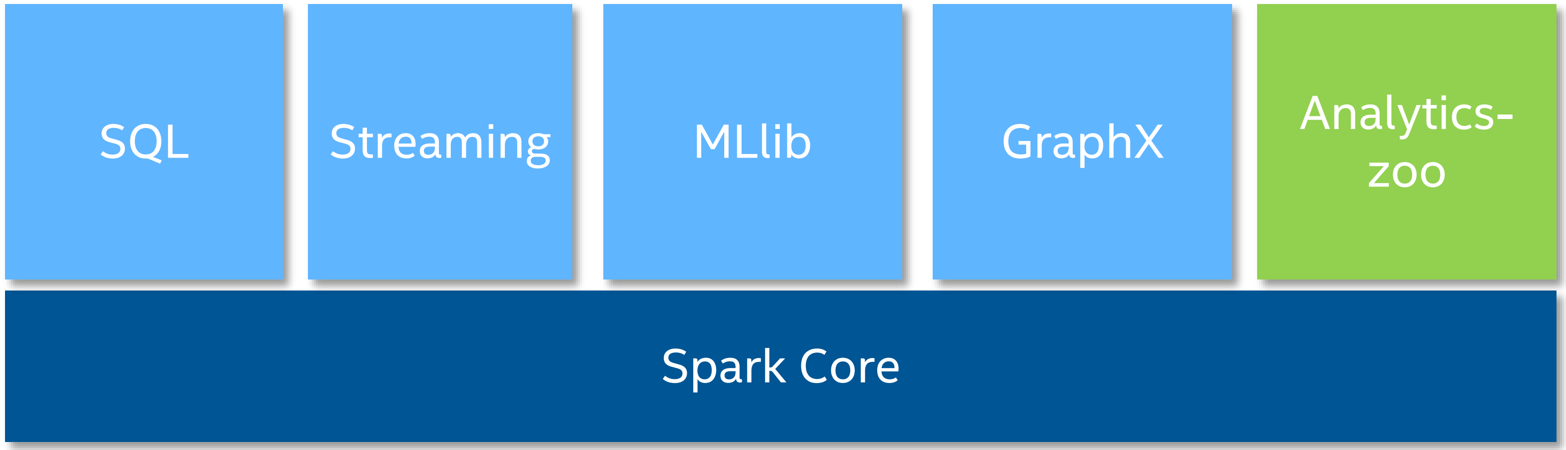
Parameter Synchronization in Analytics-zoo

Highlight

- Implement an P2P All Reduce Algorithm on Apache Spark
- Spark block manager as parameter server (handle different APIs of Spark 1.x/2.x)
- Compress float32 parameter to float16 parameter



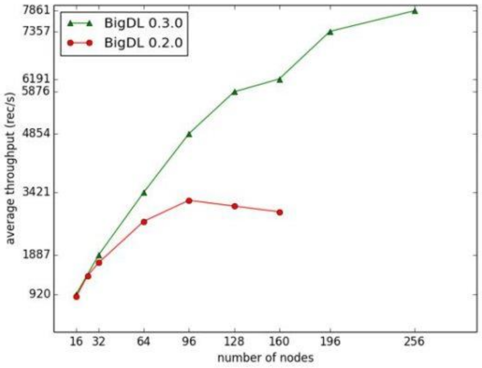
Apache Spark and Analytics-zoo



Rich deep learning features

- Tensor, Layers
  - More than 100 (Linear, Conv2D, Conv3D, Embedding, Recurrent…)
- Loss function
  - Dozens of loss functions(Cross Entropy, SmoothL1, DiceCoffient…)
- Optimization algorithm
  - SGD, Adagrad, Adam…
- Save and Load model files
  - Include torch / caffe / tensorflow

High performance from your server



- Powered by Intel Math Kernel Library
- Extremely high performance on Xeon CPUs
  - Order of magnitude faster than out of box caffe / torch / tensorflow
- Good scalability
  - Hundreds of nodes
  - https://www.cray.com/blog/scalable-deep-learning-bigdl-urika-xc-software-suite/

Use Cases

HANDS-ON PRACTICES

SPARK BASIC

The Big Data Problem

- One machine can not process or even store all the data !
- Solution is to distribute data over cluster of machine

Big Data

Word	Index	Count
I	0	1
am	2	1
Sam	5	1
I	9	1
am	11	1
Sam	14	1

I	0	1
am	2	1

Partition 1

Sam	5	1
I	9	1

Partition 2

am	11	1
Sam	14	1

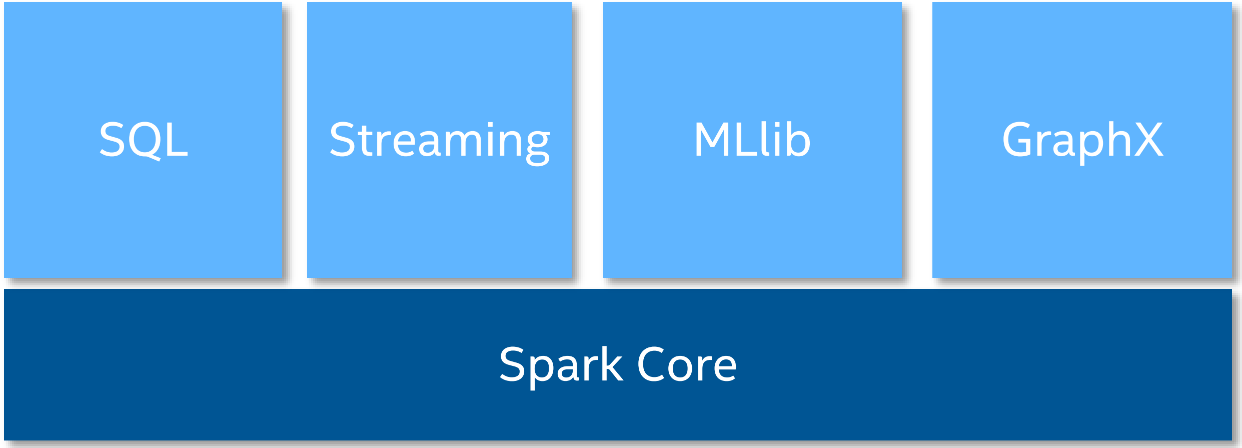
Partition 3

Apache Spark

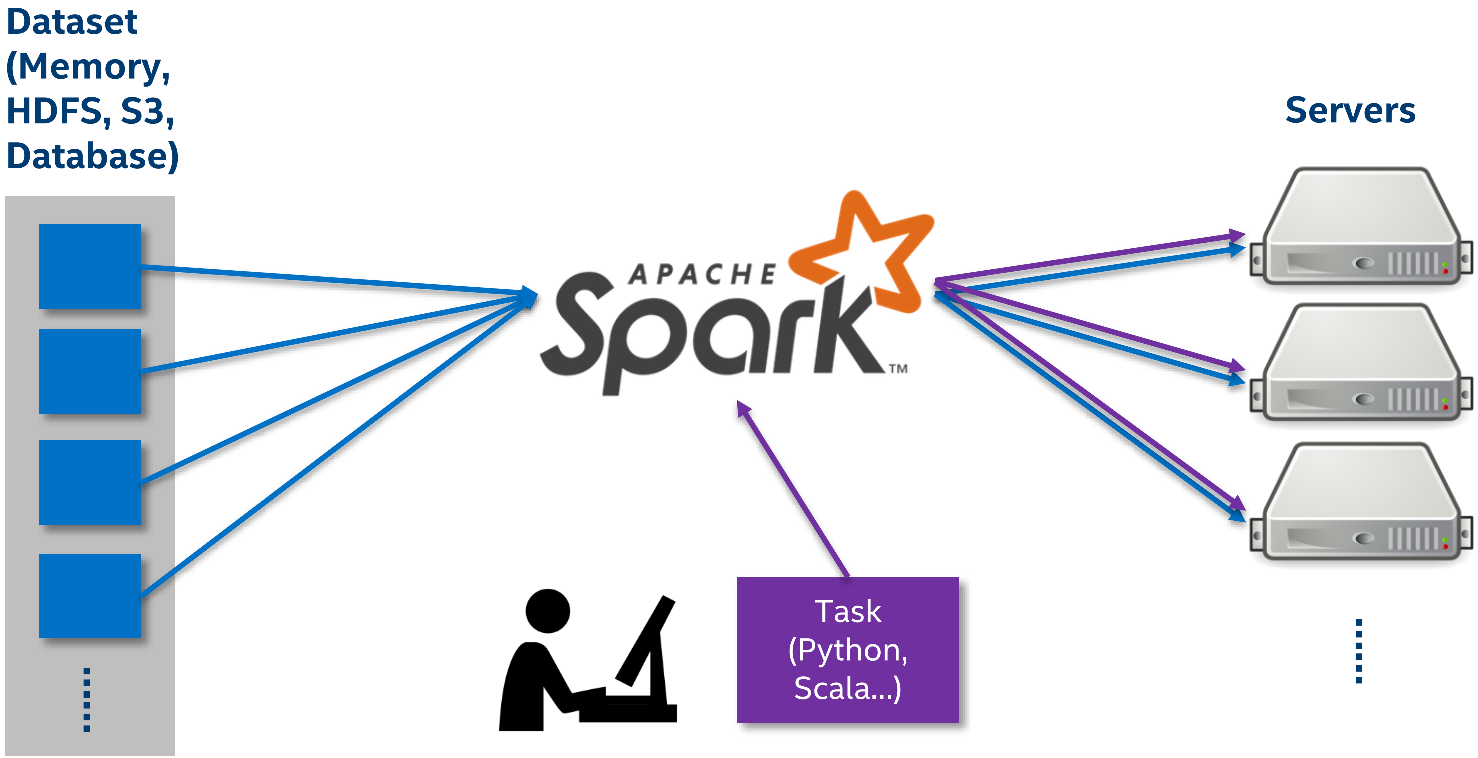
Apache Spark is a fast and general engine for large-scale data processing.

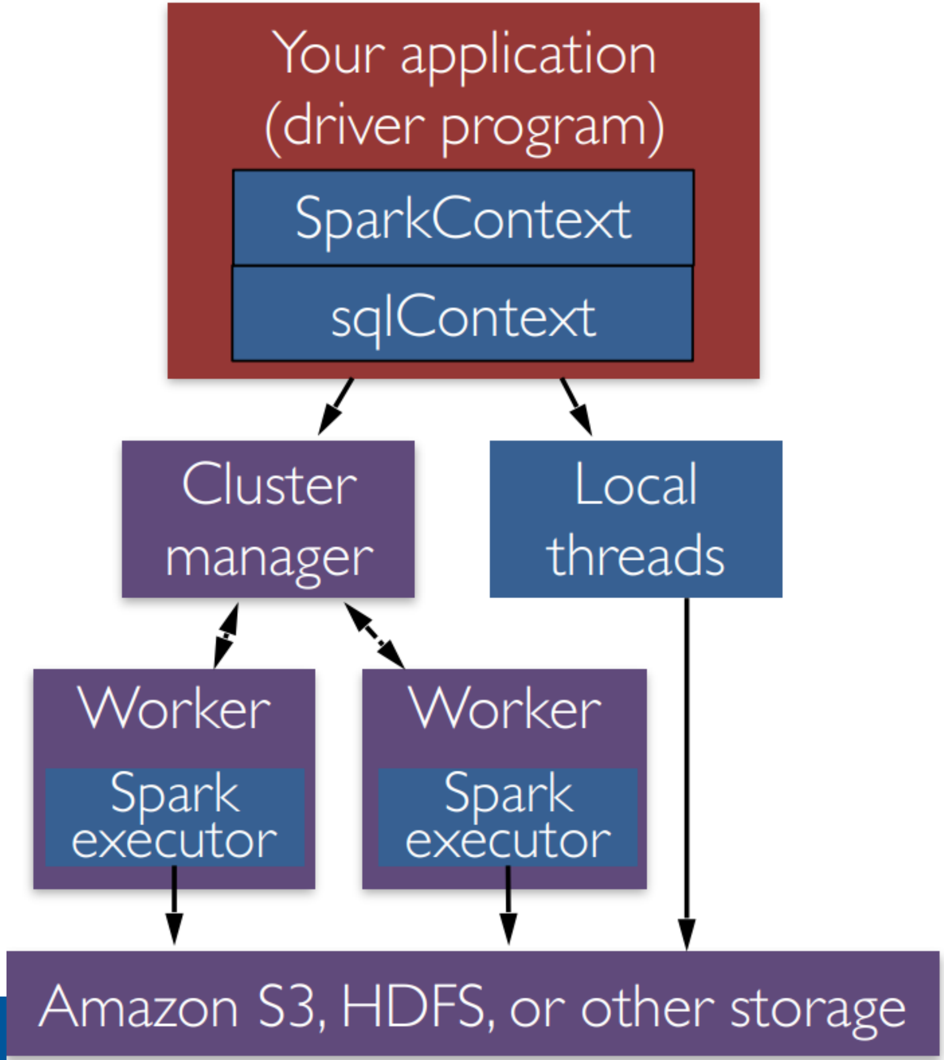
- Up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.
- Unified engine/interface for complete data applications
- SQL, Streaming, ML, Graph in the same framework
- Write applications quickly in Java, Scala, Python, R
- Runs on Hadoop, Mesos, standalone, or in the cloud (K8S is WIP)
- Access diverse data sources including HDFS, Cassandra, HBase, and S3.

Apache Spark Components



How does Apache Spark work




Your application (driver program)
SparkContext
sqlContext
Cluster manager
Local threads
Worker
Spark executor
Worker
Spark executor
Amazon S3, HDFS, or other storage

Get Analytics-zoo packages

- Pip install
  - Recommend for python user (only support spark 2.2)
- Download
  - If your spark is other version
- Maven / Sbt
  - For Java/scala user
- Build from source code
  - For Analytics-zoo developer

Run Analytics-zoo program (pip install)

```
from zoo.common.nncontext import *
from zoo.pipeline.api.keras.layers import *
from zoo.pipeline.api.keras.models import *
from zoo.pipeline.api.autograd import *

sc = get_nncontext()
dense = Dense(1, input_shape=[2])
```

$ python your_python_file.py

Run Analytics-zoo program (on the cluster)

```
spark-submit \
  --master xxx
  --jars path_to_zoo_jar
  --py-files path_to_zoo_python_zip
  your_python_file
  ……
```

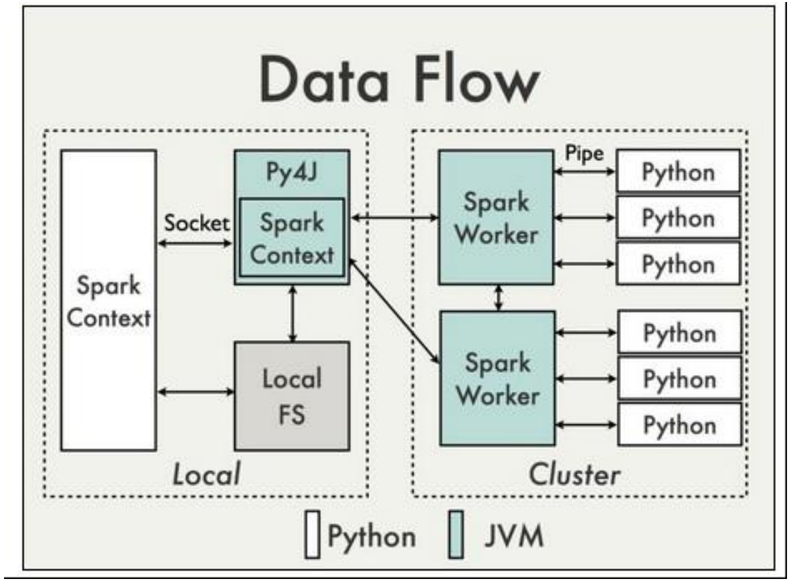
ESSENTIAL API

Define A Model

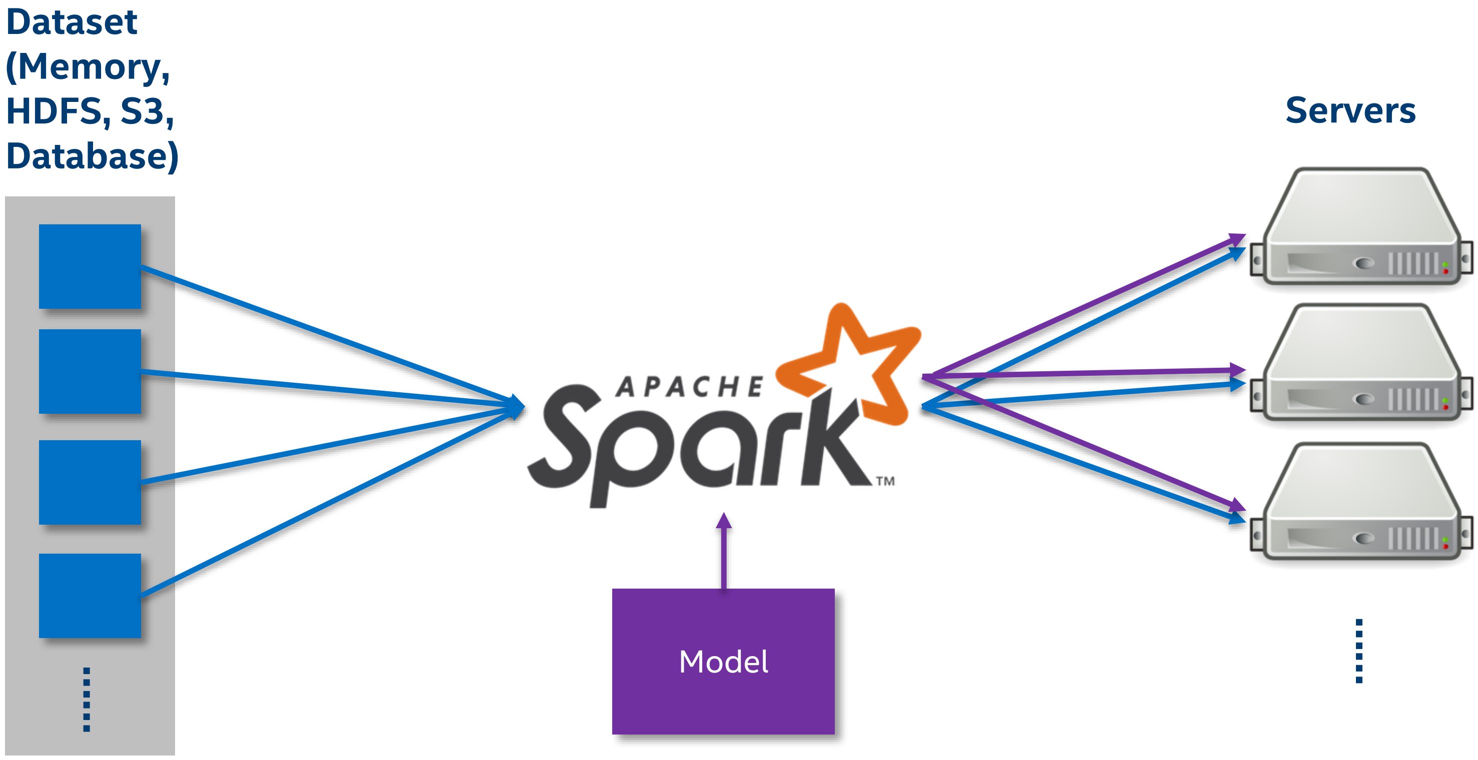
- Sequential API
  - In sequential API, user adds layers into some containers to build the model
- Functional API
  - In functional API, the model is described as a graph

Pipeline

RDD[raw data] → **Transform (python)** → **RDD[Sample(ndarray,ndarray)]** → **Train(python model)**



Distributed Evaluation and Prediction



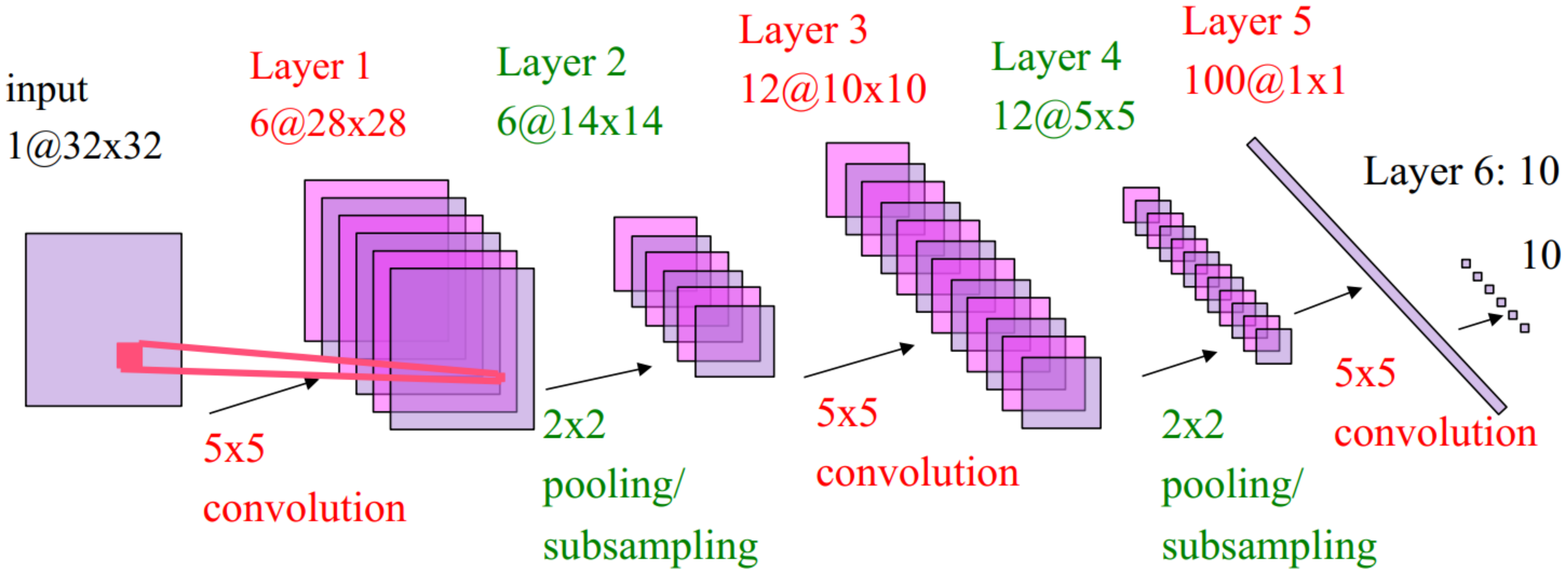
Model Quantization

Quantize the model to get higher speed

```
model = ...

quantizedModel = model.quantize()
```

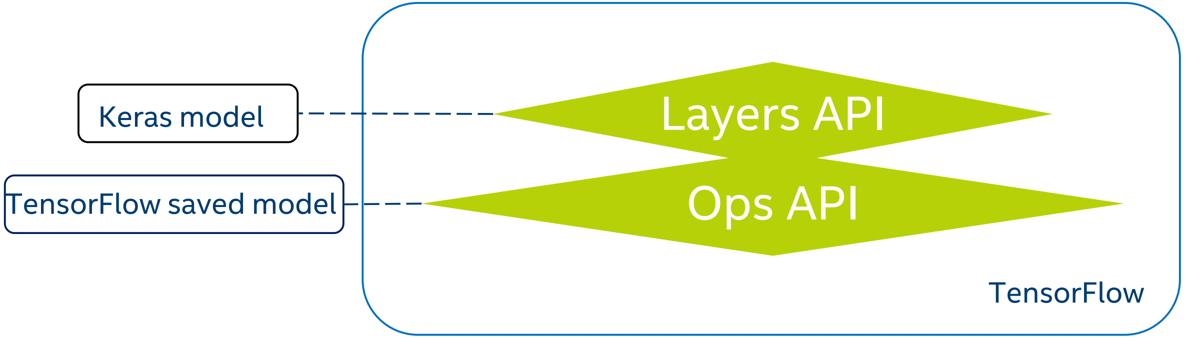
Lenet5



input
1@32x32
Layer 1
6@28x28
Layer 2
6@14x14
Layer 3
12@10x10
Layer 4
12@5x5
Layer 5
100@1x1
Layer 6: 10
10
5x5
convolution
2x2
pooling/
subsampling
5x5
convolution
2x2
pooling/
subsampling
5x5
convolution

Keras Support

- Keras 1.2.2
- Load Keras Model
- Keras-like API



Load Keras model

```python
from keras.applications import ResNet50
keras_model = ResNet50(weights="imagenet")
# Load a Keras definition
bmodel = DefinitionLoader.from_kmodel(keras_model)
# Dump weights from kears model to BigDL
WeightLoader.load_weights_from_kmodel(bmodel, keras_model)
```

```python
model = Model.load_keras(json_path=None, hdf5_path=None, by_name=False)
```

Keras-like API

```
input1 = Input((28, 28, 1))
reshape = Reshape((1, 28, 28))(input1)
conv1 = Convolution2D(6, 5, 5, activation="tanh", name="conv1_5x5")(reshape)
pool1 = MaxPooling2D()(conv1)
conv2 = Convolution2D(12, 5, 5, activation="tanh", name="conv2_5x5")(pool1)
pool2 = MaxPooling2D()(conv2)
flatten = Flatten()(pool2)
fc1 = Dense(100, activation="tanh", name="fc1")(flatten)
fc2 = Dense(class_num, activation="softmax", name="fc2")(fc1)
return Model(input1, fc2)
```

Caffe Support

Load caffe model

```
model = Net.load_caffe_model(caffe.prototxt, caffe.model)
```

Load Caffe Model Weights to Predefined BigDL Model

```
model = Net.load_caffe(bigdlModel, caffe.prototxt,
caffe.model, match_all=True)
```

Notebook

https://github.com/zhichao-li/tzoo/tree/master/notebooks/part1

Cloud & Big Data Platforms

Running BigDL, Deep Learning for Apache Spark, on AWS* (Amazon* Web Service)
https://aws.amazon.com/blogs/ai/running-bigdl-deep-learning-for-apache-spark-on-aws/

BigDL on Alibaba* Cloud E-MapReduce*
https://yq.aliyun.com/articles/73347

BigDL on CDH* and Cloudera* Data Science Workbench*
http://blog.cloudera.com/blog/2017/04/bigdl-on-cdh-and-cloudera-data-science-workbench/

BigDL Spark deep learning library VM now available on Microsoft* Azure* Marketplace https://azure.microsoft.com/en-us/blog/bigdl-spark-deep-learning-library-vm-now-available-on-microsoft-azure-marketplace/

BigDL in KMR* Service of Kingsoft* Cloud
https://docs.ksyun.com/read/latest/33/_book/bigDL.html

Using BigDL in IBM* Data Science Experience
https://medium.com/ibm-data-science-experience/using-bigdl-in-data-science-experience-for-deep-learning-on-spark-f1cf30ad6ca0

Using BigDL for deep learning with Apache Spark and Google* Cloud Dataproc*
https://cloud.google.com/blog/big-data/2018/04/using-bigdl-for-deep-learning-with-apache-spark-and-google-cloud-dataproc

Intel's BigDL on Databricks*
https://databricks.com/blog/2017/02/09/intels-bigdl-databricks.html

BigDL Shipped in Cray* Urika-XC* Analytics Software Suite
https://www.cray.com/blog/scalable-deep-learning-bigdl-urika-xc-software-suite/

BigDL @ JD.COM

Problem

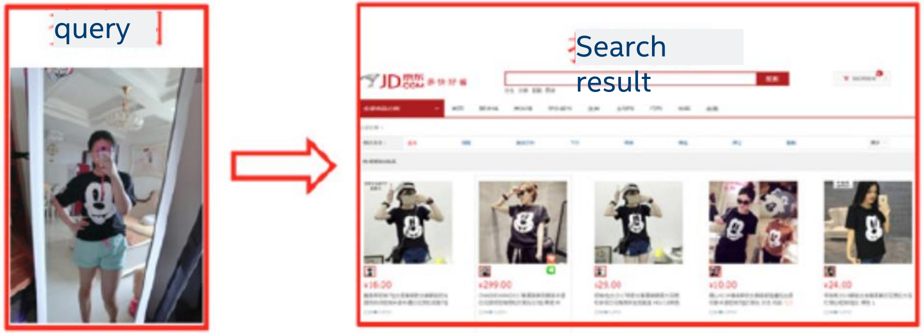
Large-scale image feature extraction

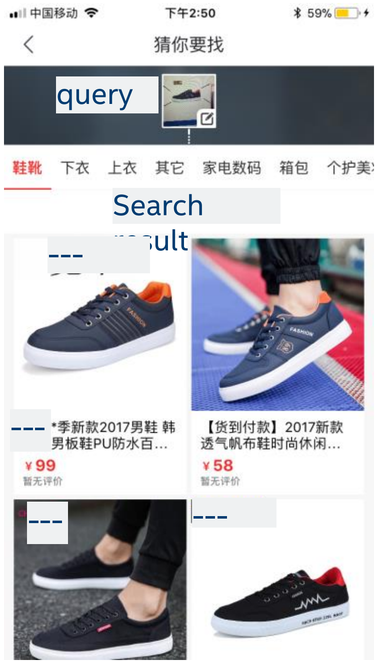
- Object detect (remove background, optional)
- Feature extraction

Application

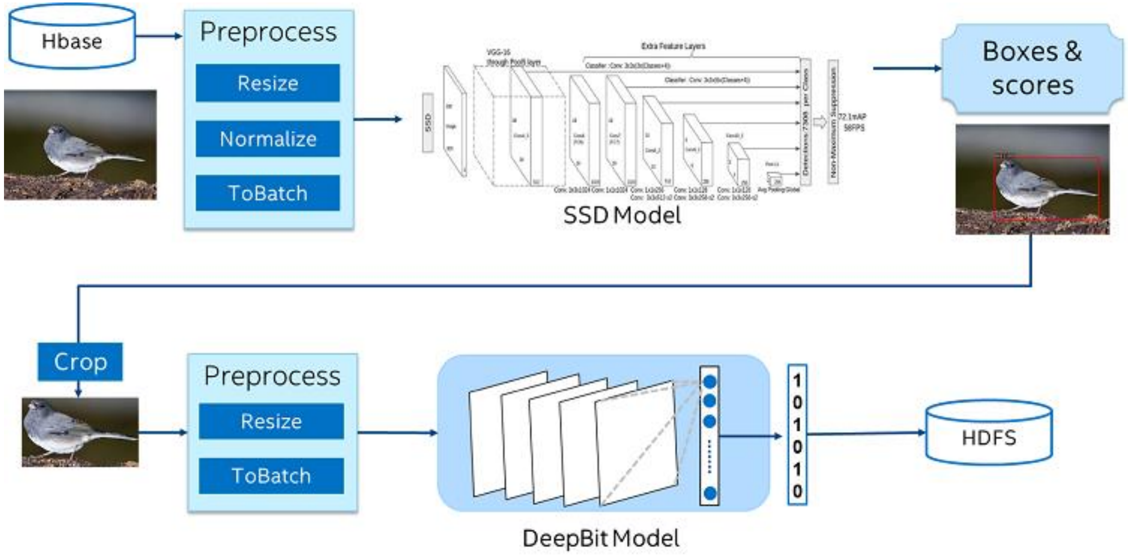
- Similar image search
- Image Deduplication

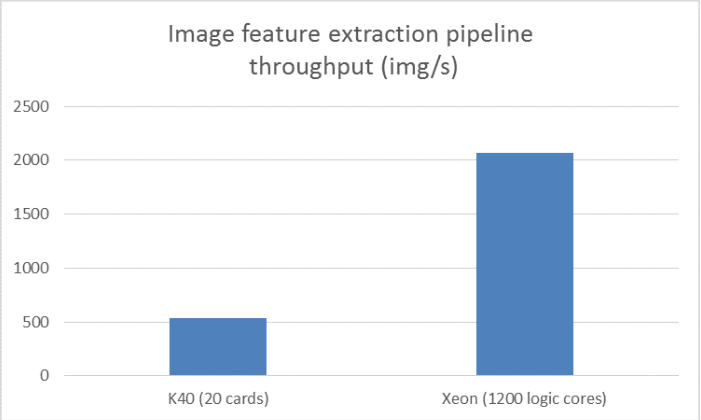
Similar image search





Object Detection and Image Feature Extraction in JD

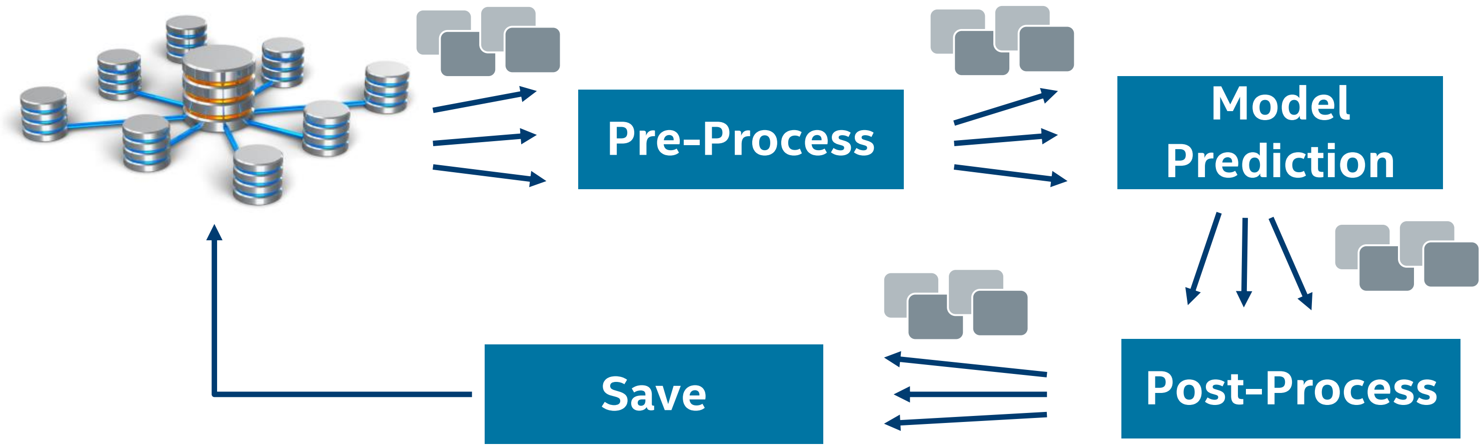




- Reuse existing Hadoop/Spark clusters for deep learning with no changes (image search, IP protection, etc.)
- Efficiently scale out on Spark with superior performance (*3.83x* speed-up vs. GPU severs) as benchmarked by JD

http://mp.weixin.qq.com/s/xUCkzbHK4K06-v5qUsaNQQ
https://software.intel.com/en-us/articles/building-large-scale-image-feature-extraction-with-bigdl-at-jdcom

Build a distributed image prediction pipeline on Spark using BigDL



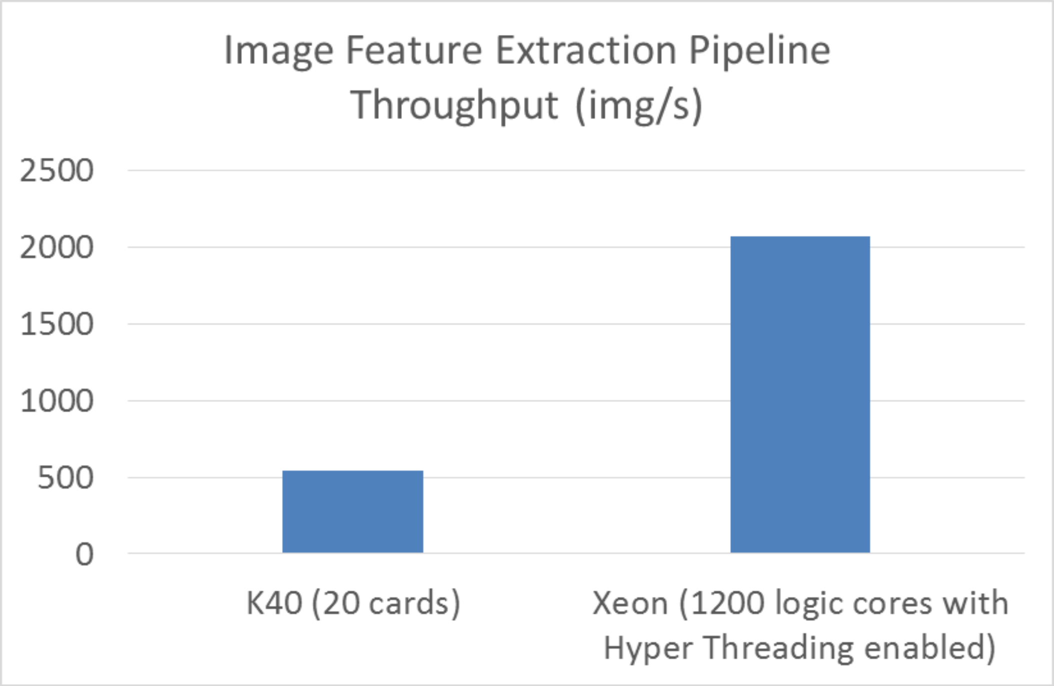

```
val distImageFrame = ImageFrame.read(folder, sc) -> preprocessor
val model = Module.loadModule(path)
model.predict(distImageFrame)
distImageFrame.save(outPath)
```

Pipeline Correctness

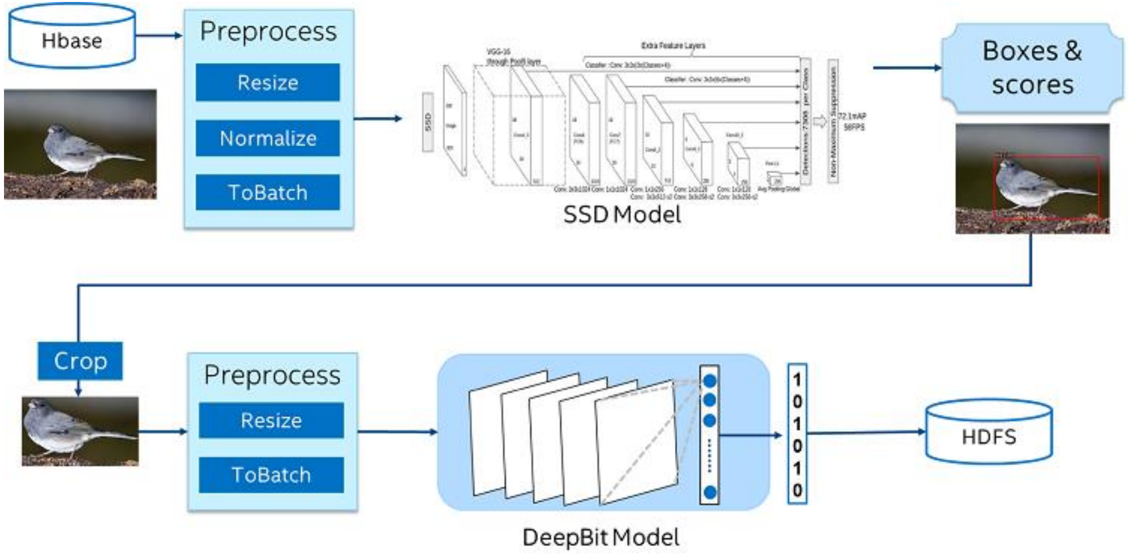
Almost same as Caffe GPU

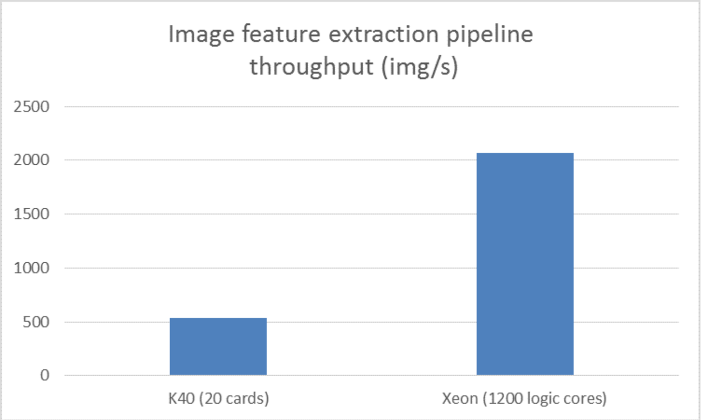
Element-wise error < 0.001%

3.83x Speed up compared to GPU solution



Object Detection and Image Feature Extraction in JD





- Reuse existing Hadoop/Spark clusters for deep learning with no changes (image search, IP protection, etc.)
- Efficiently scale out on Spark with superior performance (*3.83x* speed-up vs. GPU severs) as benchmarked by JD

http://mp.weixin.qq.com/s/xUCkzbHK4K06-v5qUsaNQQ
https://software.intel.com/en-us/articles/building-large-scale-image-feature-extraction-with-bigdl-at-jdcom

User-Merchant Propensity Modeling in MasterCard

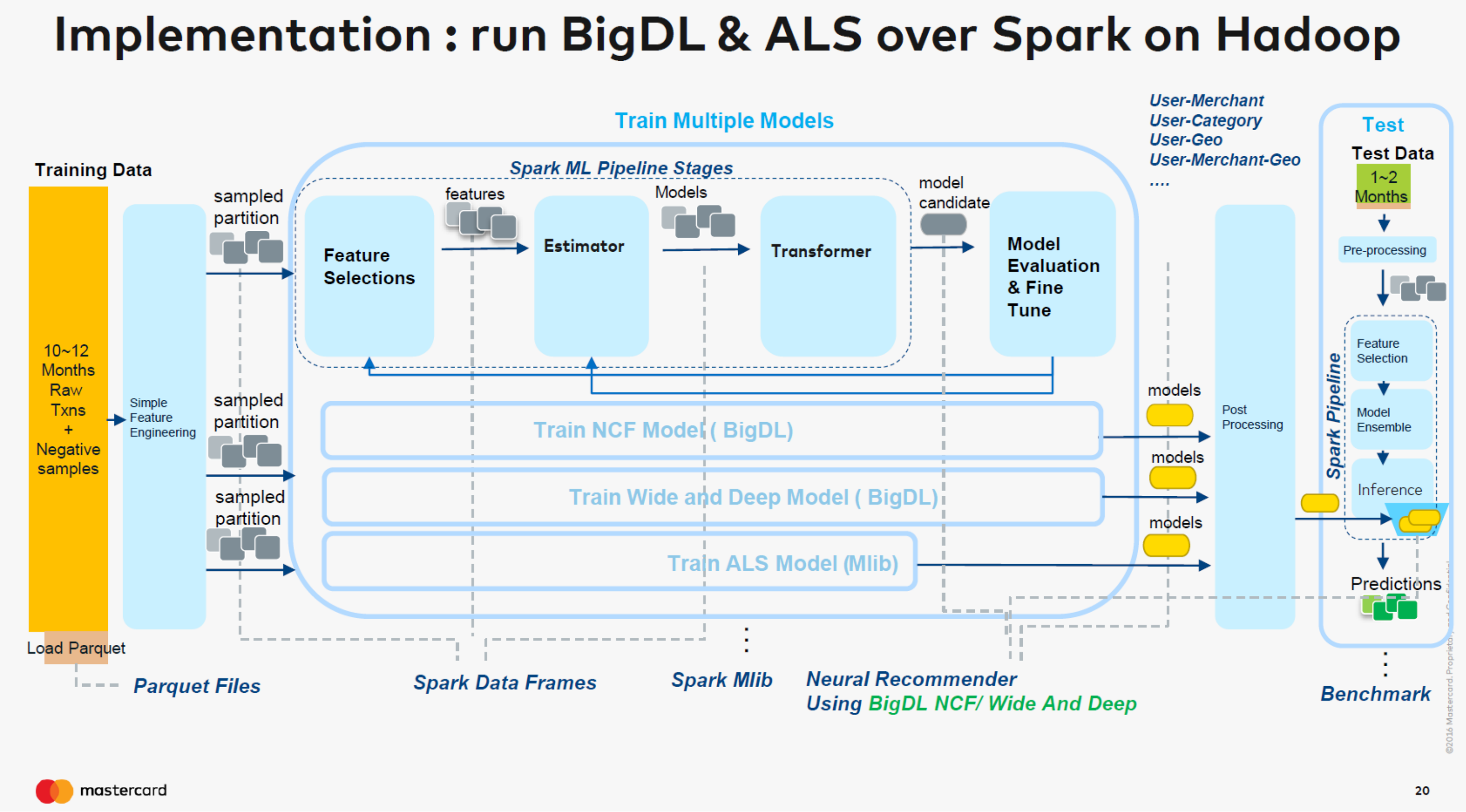
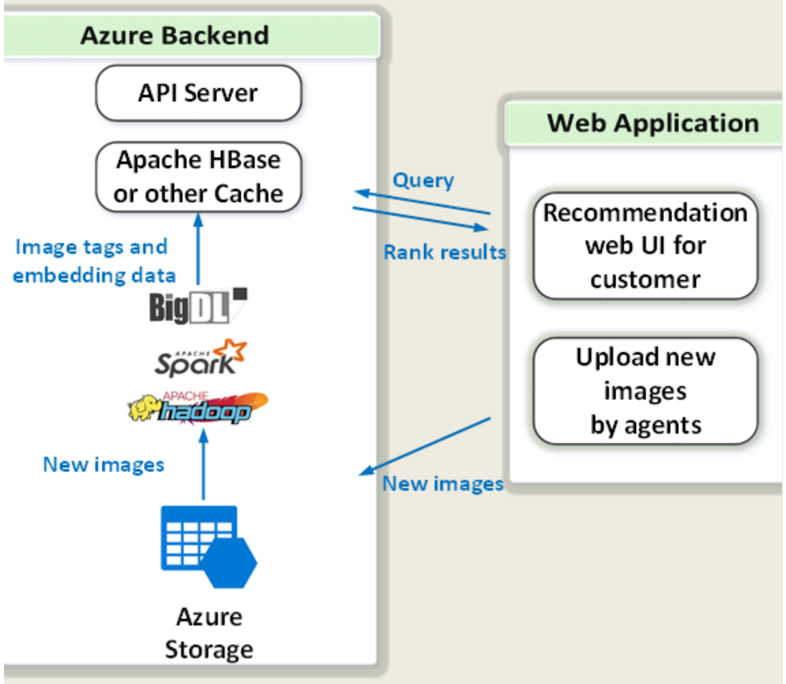
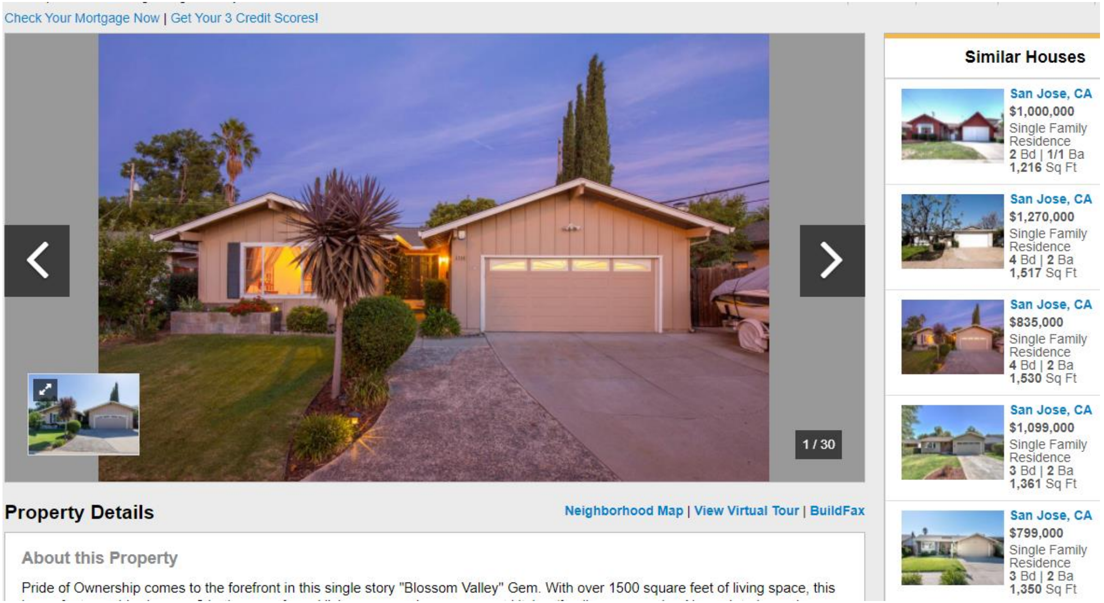


Image Similarity Search for MLSListings

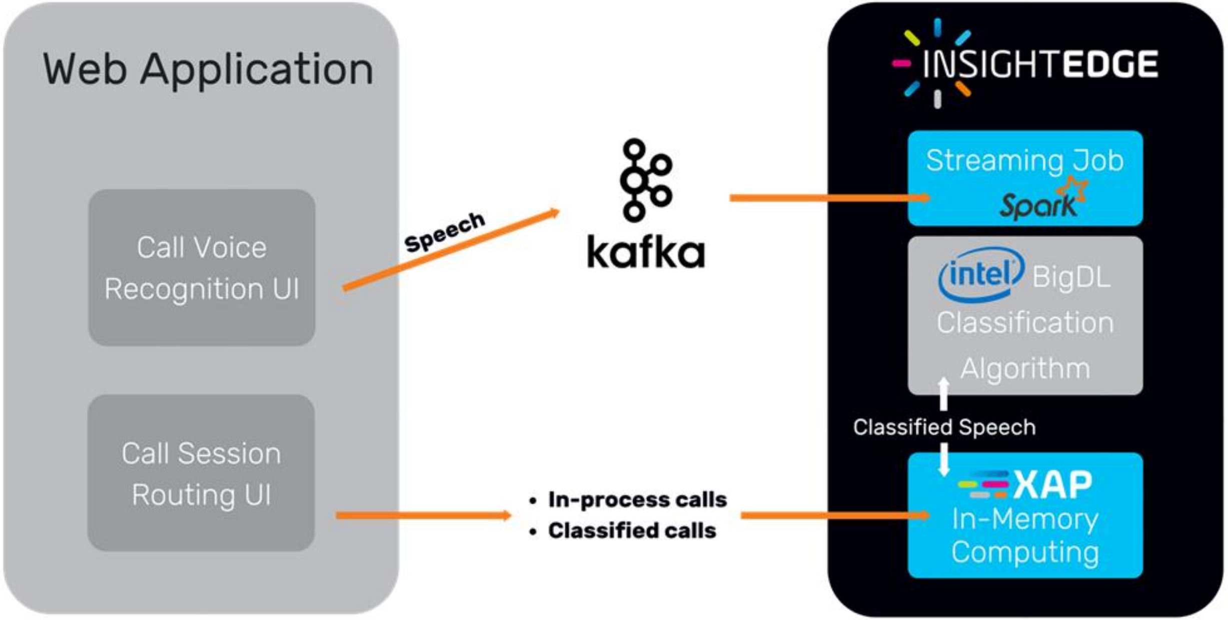


MLSlistings built image-similarity based house recommendations using BigDL on Microsoft Azure

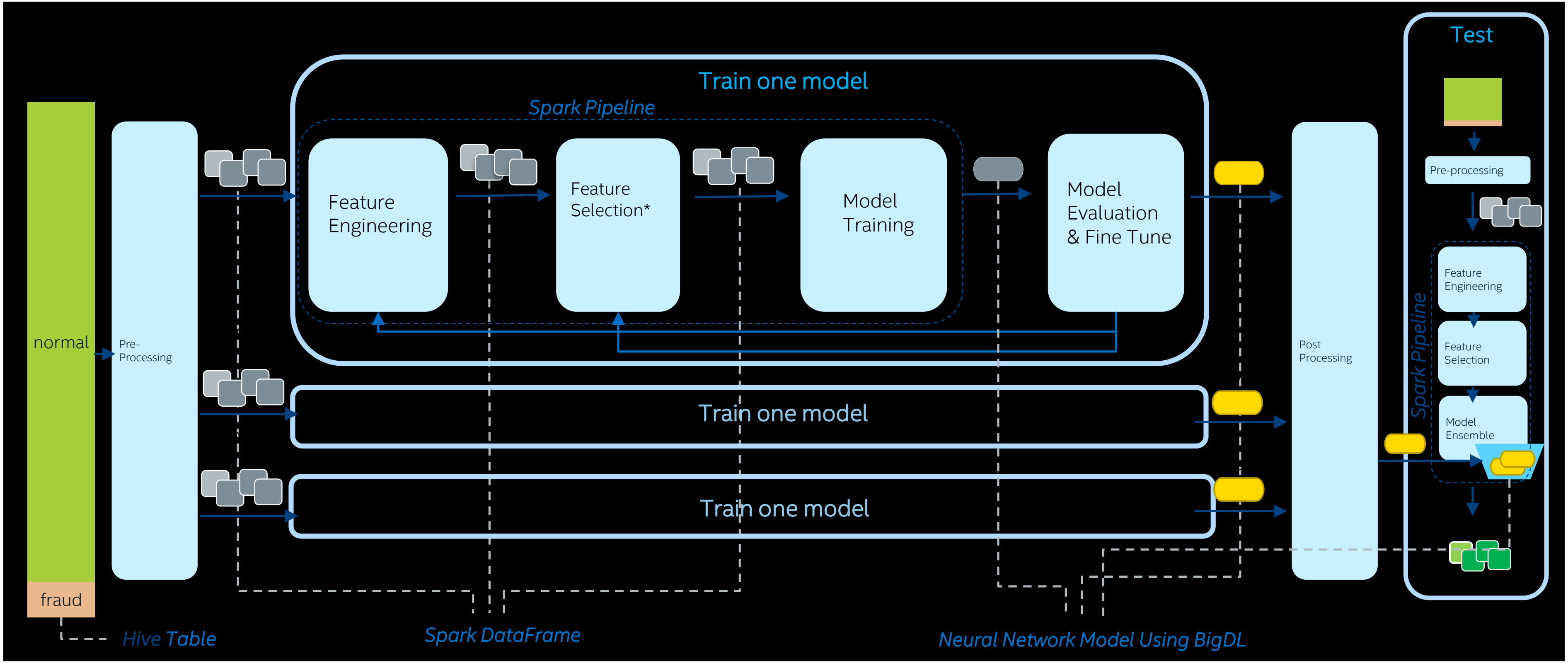


https://software.intel.com/en-us/articles/using-bigdl-to-build-image-similarity-based-house-recommendations

NLP Based Call Center Routing in GigaSpaces



Fraud Detection in UnionPay



https://mp.weixin.qq.com/s?__biz=MzI3NDAwNDUwNg==&mid=2648307335&idx=1&sn=8eb9f63eaf2e40e24a90601b9cc03d1f

Hands on

- Pre-trained ResNet
- Anomaly detection
- Recommendation (NCF wide and deep)
- VAE

https://github.com/zhichao-li/tzoo/tree/master/notebooks/part2

Software

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document 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.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Intel, Quark, VTune, Xeon, Cilk, Atom, Look Inside and the Intel logo are trademarks of Intel Corporation in the United States and other countries.

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

Risk Factors

The above statements and any others in this document that refer to plans and expectations for the first quarter, the year and the future are forward-looking statements that involve a number of risks and uncertainties. Words such as “anticipates,” “expects,” “intends,” “plans,” “believes,” “seeks,” “estimates,” “may,” “will,” “should” and their variations identify forward-looking statements. Statements that refer to or are based on projections, uncertain events or assumptions also identify forward-looking statements. Many factors could affect Intel’s actual results, and variances from Intel’s current expectations regarding such factors could cause actual results to differ materially from those expressed in these forward-looking statements. Intel presently considers the following to be the important factors that could cause actual results to differ materially from the company’s expectations. Demand could be different from Intel's expectations due to factors including changes in business and economic conditions; customer acceptance of Intel’s and competitors’ products; supply constraints and other disruptions affecting customers; changes in customer order patterns including order cancellations; and changes in the level of inventory at customers. Uncertainty in global economic and financial conditions poses a risk that consumers and businesses may defer purchases in response to negative financial events, which could negatively affect product demand and other related matters. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Revenue and the gross margin percentage are affected by the timing of Intel product introductions and the demand for and market acceptance of Intel's products; actions taken by Intel's competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel’s response to such actions; and Intel’s ability to respond quickly to technological developments and to incorporate new features into its products. The gross margin percentage could vary significantly from expectations based on capacity utilization; variations in inventory valuation, including variations related to the timing of qualifying products for sale; changes in revenue levels; segment product mix; the timing and execution of the manufacturing ramp and associated costs; start-up costs; excess or obsolete inventory; changes in unit costs; defects or disruptions in the supply of materials or resources; product manufacturing quality/yields; and impairments of long-lived assets, including manufacturing, assembly/test and intangible assets. Intel's results could be affected by adverse economic, social, political and physical/infrastructure conditions in countries where Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Expenses, particularly certain marketing and compensation expenses, as well as restructuring and asset impairment charges, vary depending on the level of demand for Intel's products and the level of revenue and profits. Intel’s results could be affected by the timing of closing of acquisitions and divestitures. Intel's results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust, disclosure and other issues, such as the litigation and regulatory matters described in Intel's SEC reports. An unfavorable ruling could include monetary damages or an injunction prohibiting Intel from manufacturing or selling one or more products, precluding particular business practices, impacting Intel’s ability to design its products, or requiring other remedies such as compulsory licensing of intellectual property. A detailed discussion of these and other factors that could affect Intel’s results is included in Intel’s SEC filings, including the company’s most recent reports on Form 10-Q, Form 10-K and earnings release.