Cluster Serving:

Distributed and Automated Model Inference on Big Data Streaming Frameworks

Authors: Jiaming Song, Dongjie Shi, Qiyuan Gong, Lei Xia, Jason Dai

Outline

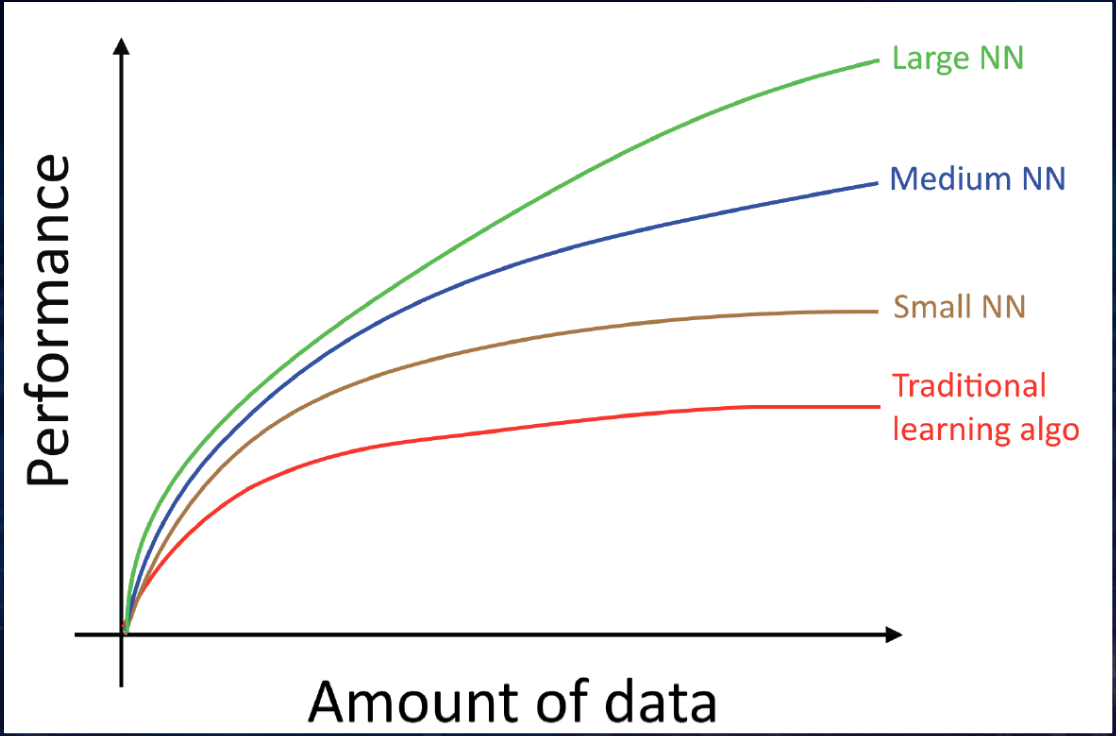
Challenges AI productions facing

Integrated Big Data and AI pipeline

Scalable online serving

Cross-industry end-to-end use cases

Big Data & Model Performance



“Machine Learning Yearning”, Andrew Ng, 2016

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

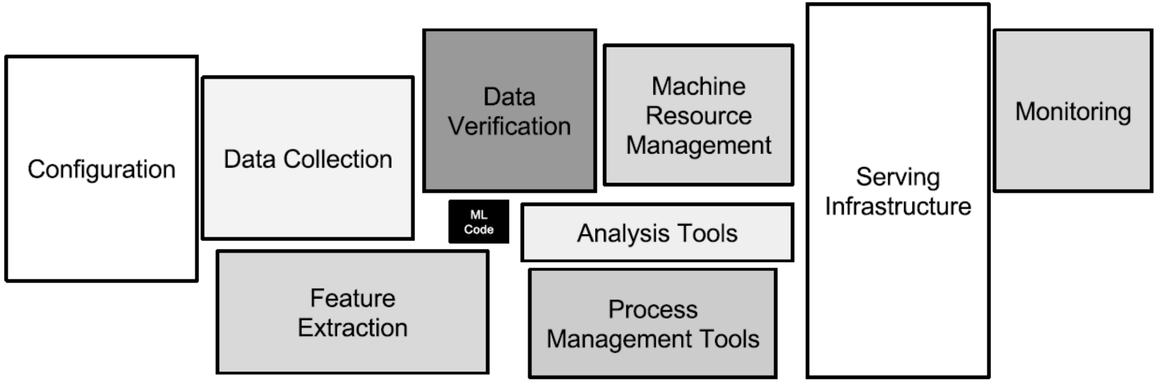


Figure 1: Only a small fraction of real-world ML systems is composed of the ML code, as shown by the small black box in the middle. The required surrounding infrastructure is vast and complex.

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

Outline

Challenges AI productions facing

Integrated Big Data and AI pipeline

Scalable online serving

Cross-industry end-to-end use cases

Integrated Big Data Analytics and AI

Seamless Scaling from Laptop to Distributed Big Data

- **Easily prototype end-to-end pipelines that apply AI models to big data**
- **“Zero” code change from laptop to distributed cluster**
- **Seamlessly deployed on production Hadoop/K8s clusters**
- **Automate the process of applying machine learning to big data**

AI ON BIG DATA

BigDL

Distributed, High-Performance
Deep Learning Framework
for Apache Spark*

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

ANALYTICS ZOO

Unified Analytics + AI Platform
for TensorFlow*, PyTorch*, Keras*, BigDL, OpenVINO, Ray* and Apache Spark*

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

Seamless Scaling from Laptop to Distributed Big Data

Analytics Zoo

Unified Data Analytics and AI Platform

Models & Algorithms
- Recommendation
- Time Series
- Computer Vision
- NLP

ML Workflow
- AutoML
- Automatic Cluster Serving

End-to-end Pipelines
- Distributed TensorFlow* & PyTorch* on Spark*
- RayOnSpark
- Spark* Dataframes & ML Pipelines for DL
- InferenceModel

Compute Environment
- Laptop
- K8S* Cluster
- Hadoop* Cluster
- Cloud

- DL Frameworks (TF*/PyTorch*/OpenVINO*/…)
- Distributed Analytics (Spark*/Flink*/Ray*/…)
- Python Libraries (Numpy/Pandas/sklearn/…)

Powered by oneAPI

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

Outline

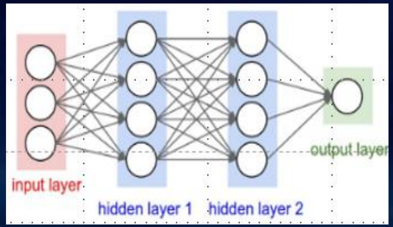
Challenges AI productions facing

Integrated Big Data and AI pipeline

Scalable online serving

Cross-industry end-to-end use cases

What’s Serving

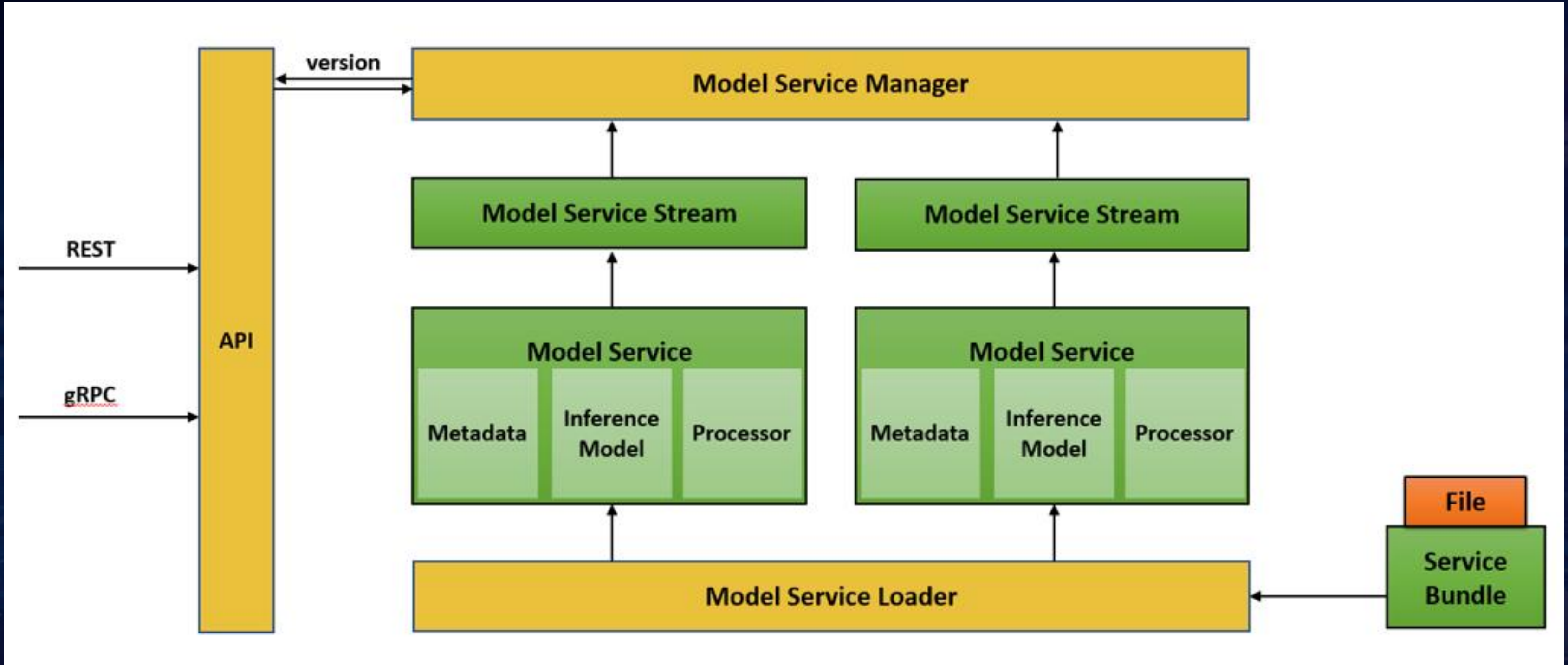


model

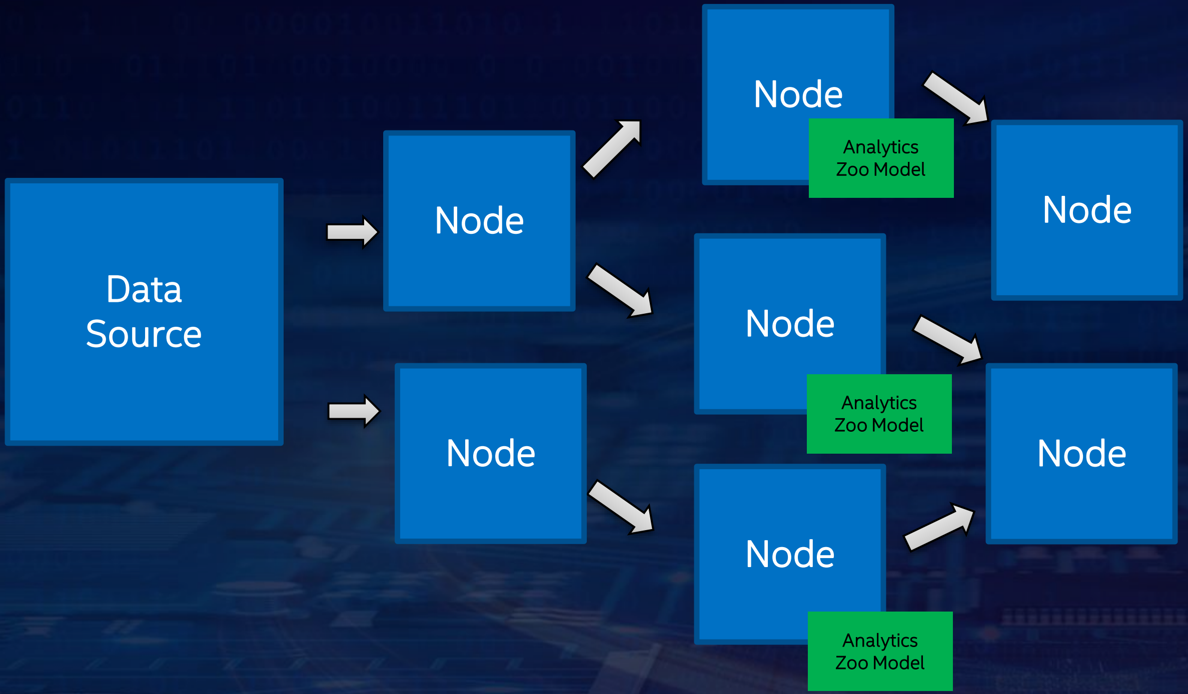
Input Data → Preprocessing → Inference → Postprocessing → Result

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

Example of Classical Web Serving



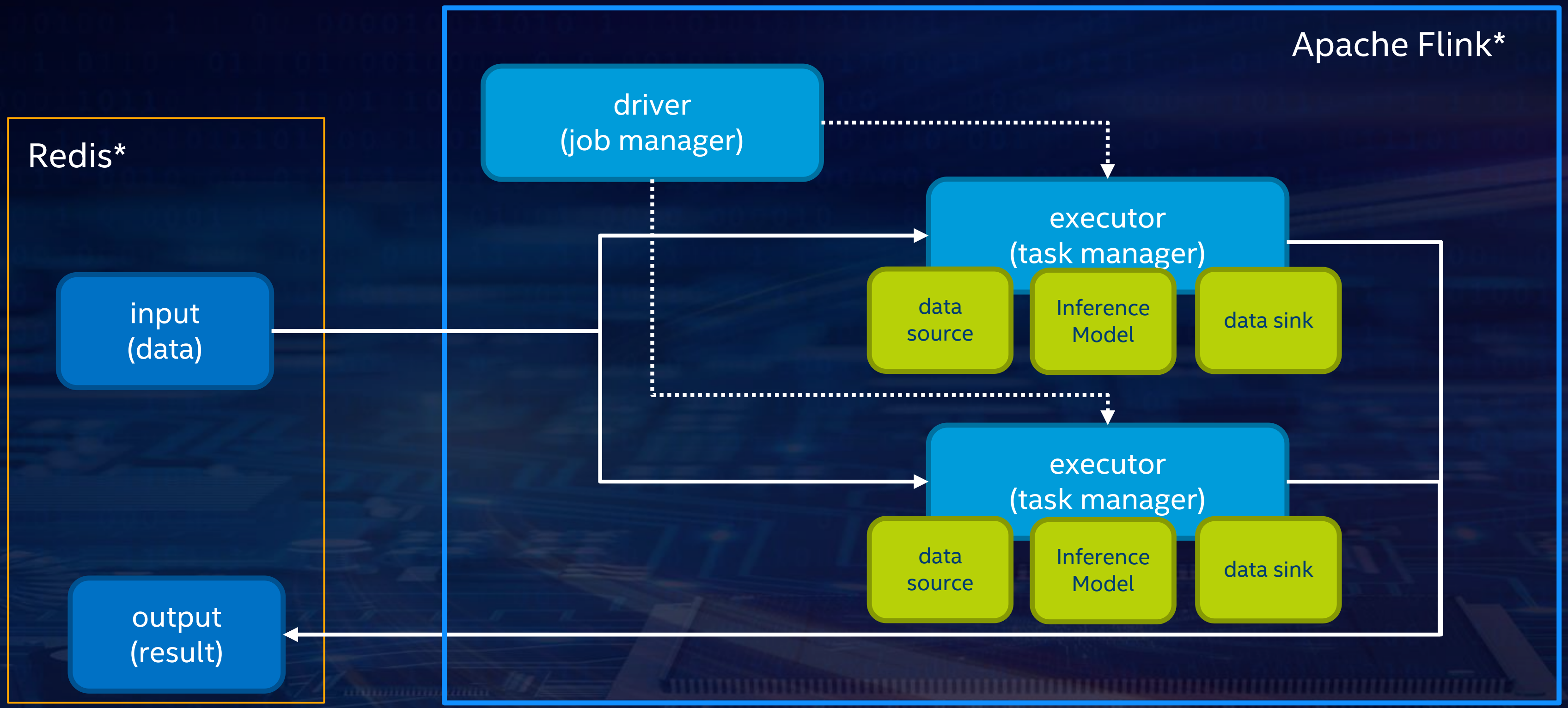
Distributed Model Serving



Distributed model serving in Flink*, Spark*, Kafka*, Storm*, etc

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

Architecture of Main Version of Cluster Serving



Version based on Spark* Streaming is also supported.

Advantages of Analytics Zoo Cluster Serving

Ease of Deployment

One container with all dependencies & leverage existed YARN/K8S cluster

Wide Range Deep Learning model support

Tensorflow*, Caffe*, OpenVINO*, Pytorch*, BigDL*

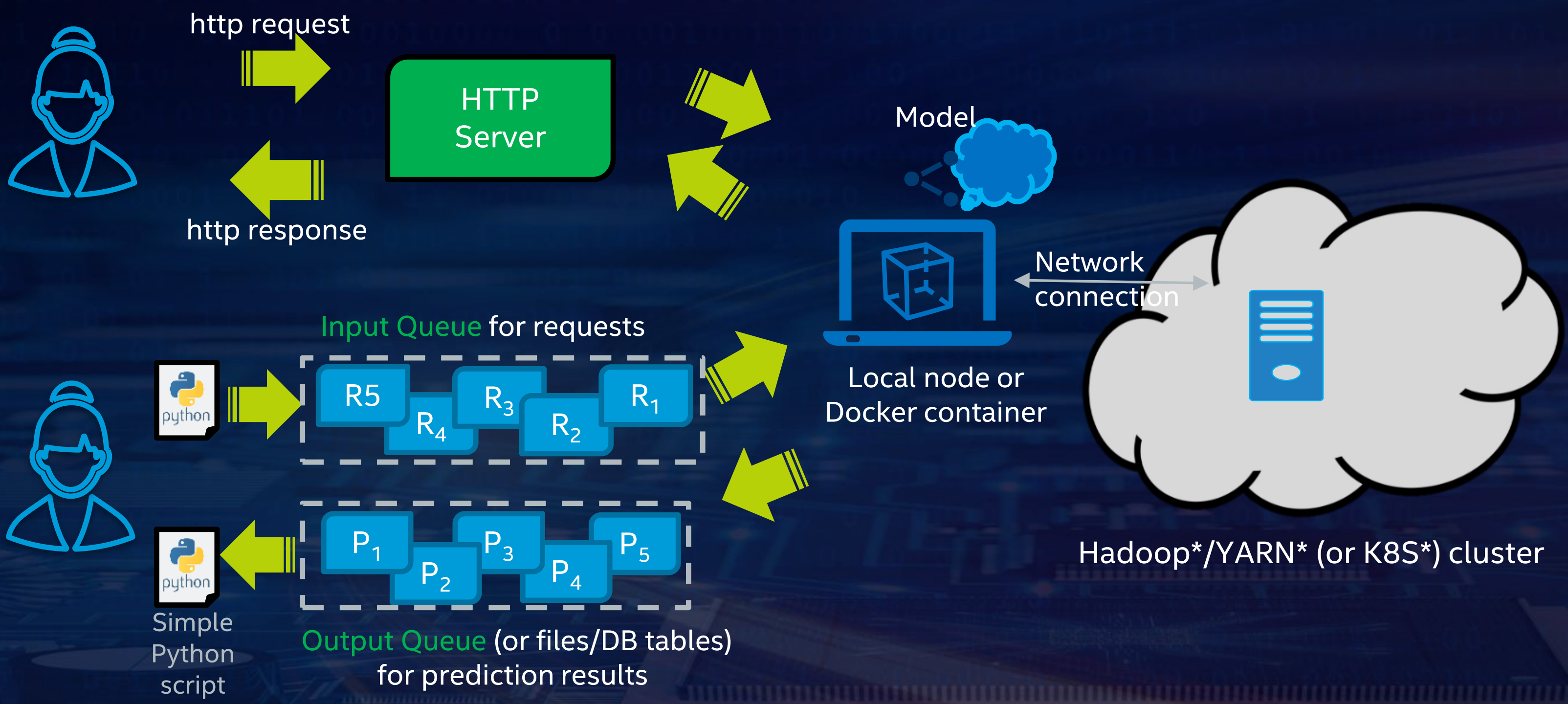
Low Latency

Continuous Streaming pipeline is supported by Apache Flink* and Spark*

High Throughput & Scalability

Optimization of multithread control, and could easily scale out to clusters

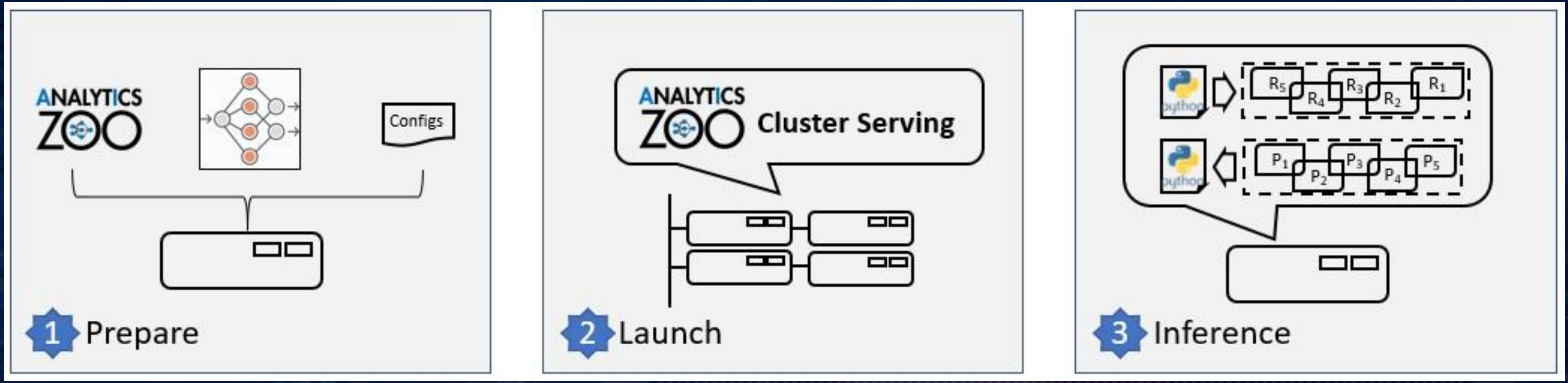
Data pipeline User Perspective



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

Cluster Serving Workflow Overview

1. Install and prepare Cluster Serving environment on a local node
2. Launch the Cluster Serving service
3. Distributed, real-time (streaming) inference



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

Very Quick Start

Start docker container

```
#docker run -itd --name cluster-serving --net=host intelanalytics/zoo-cluster-serving:0.7.0
```

Log into container

```
#docker exec -it cluster-serving bash
```

Start Serving

```
#cluster-serving-start
```

https://github.com/intel-analytics/analytics-zoo/blob/master/docs/docs/ClusterServingGuide/ProgrammingGuide.md

API Introductions

http sync API

data are represented by json format
call http post method to enqueue your data into pipeline
http API is compatible with TFServing*

pub-sub python async API

data are represented by ndarray
call python method to enqueue your data into pipeline

API Introductions - HTTP

http API

data are represented by json format

Support

- scalars
- tensors
- sparse tensors
- image encodings

```
curl -d \
 '{
  "instances" : [ {
    "intScalar" : 12345,
    "floatScalar" : 3.14159,
    "stringScalar" : "hello, world. hello, arrow.",
    "intTensor" : [ 7756, 9549, 1094, 9808, 4959, 3831, 3926, 6578, 1870, 1741 ],
    "floatTensor" : [ 0.6804766, 0.30136853, 0.17394465, 0.44770062, 0.20275897, 0.32762378, 0.45966738, 0.30405
    "stringTensor" : [ "come", "on", "united" ],
    "intTensor2" : [ [ 1, 2 ], [ 3, 4 ], [ 5, 6 ] ],
    "floatTensor2" : [ [ [ 0.2, 0.3 ], [ 0.5, 0.6 ] ], [ [ 0.2, 0.3 ], [ 0.5, 0.6 ] ] ],
    "stringTensor2" : [ [ [ [ "come", "on", "united" ], [ "come", "on", "united" ], [ "come", "on", "united" ],
  }, {
    "intScalar" : 12345,
    "floatScalar" : 3.14159,
    "stringScalar" : "hello, world. hello, arrow.",
    "intTensor" : [ 7756, 9549, 1094, 9808, 4959, 3831, 3926, 6578, 1870, 1741 ],
    "floatTensor" : [ 0.6804766, 0.30136853, 0.17394465, 0.44770062, 0.20275897, 0.32762378, 0.45966738, 0.30405
    "stringTensor" : [ "come", "on", "united" ],
    "intTensor2" : [ [ 1, 2 ], [ 3, 4 ], [ 5, 6 ] ],
    "floatTensor2" : [ [ [ 0.2, 0.3 ], [ 0.5, 0.6 ] ], [ [ 0.2, 0.3 ], [ 0.5, 0.6 ] ] ],
    "stringTensor2" : [ [ [ [ "come", "on", "united" ], [ "come", "on", "united" ], [ "come", "on", "united" ],
  } ]
}' \
-X POST http://host:port/predict
```

API Introductions - Pub-sub

Python API

data are represented by python objects

Support

- scalars
- tensors
- sparse tensors
- image encodings

```
from zoo.serving.client import InputQueue
import numpy as np
input_api = InputQueue()
t1 = np.array([1,2])
t2 = np.array([[1,2], [3,4]])
input_api.enqueue('my-instance', img={"path": 'path/to/image'}, tensor1=t1, tensor2=t2)
```

Outline

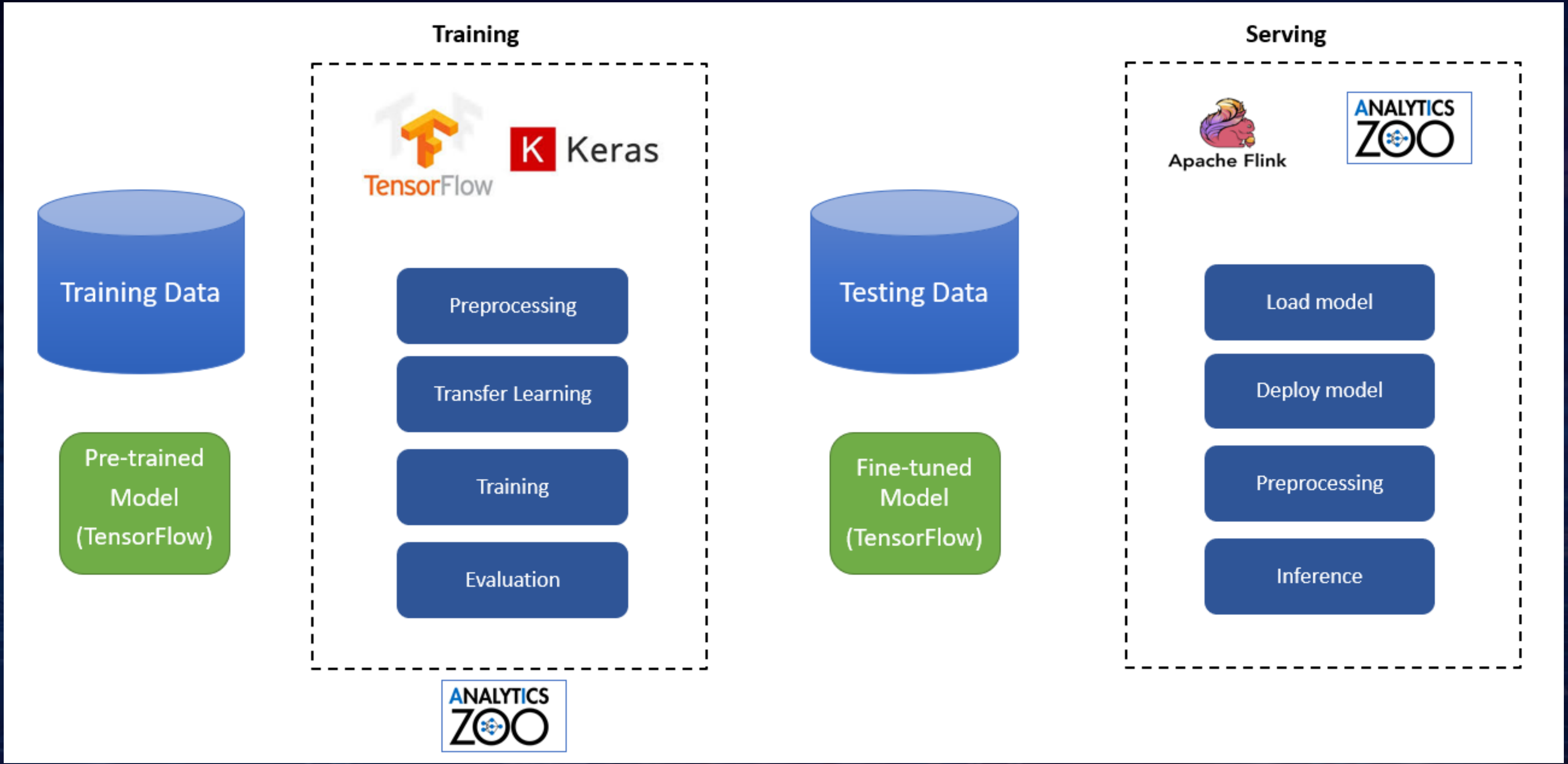
Challenges AI productions facing

Integrated Big Data and AI pipeline

Scalable online serving

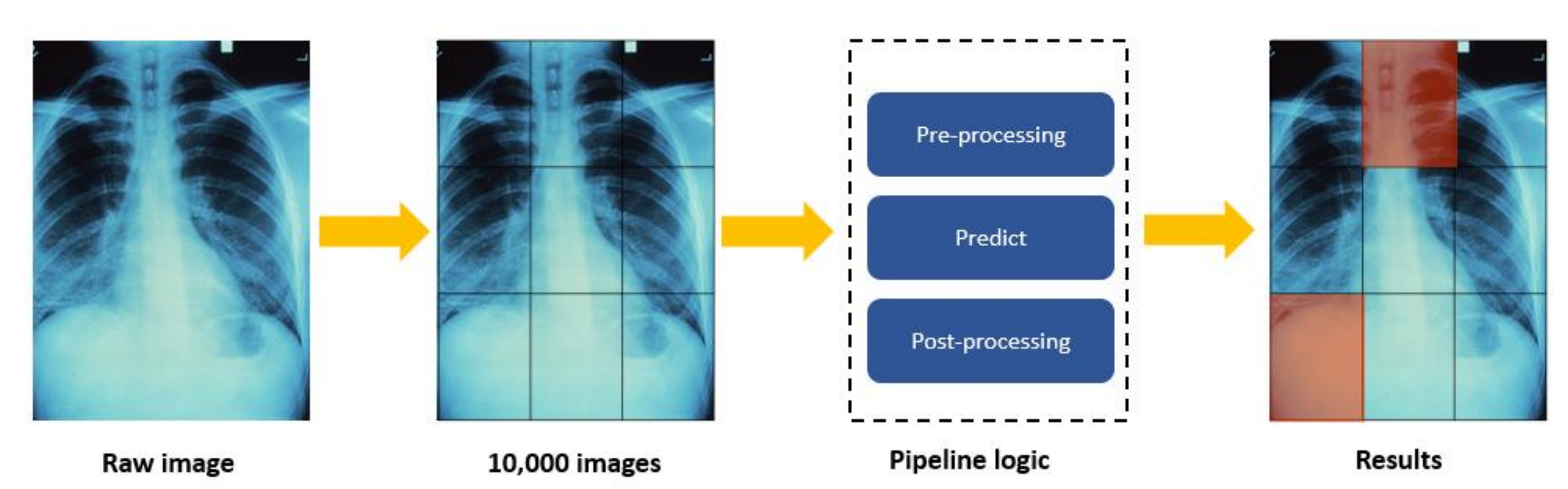
Cross-industry end-to-end use cases

Garbage classification on Tianchi Competition



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

Medical Imaging Analysis



Bottleneck:
Preprocessing, inference, up to 1-2 hours per large piece

https://en.wikipedia.org/wiki/X-ray

End-to-End Big Data and AI Pipelines

Seamless Scaling from Laptop to Production

ANALYTICS ZOO

Unified Analytics + AI Platform

Distributed TensorFlow*, Keras*, PyTorch* & BigDL on Apache Spark*

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

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

intel®

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.