BigDL 2.0

Seamlessly scale E2E distributed AI from laptop to cluster

Guoqiong Song Intel
Jennie Wang Intel

RAY SUMMIT 22

Content

- What is BigDL?
- What does BigDL have?
- How does BigDL leverage Ray?
- Real world use cases

BigDL 2.0

Seamlessly scale end-to-end, distributed AI applications

Domain Specific Toolkits

- **PPML** – Privacy Preserving Machine Learning
- **Chronos** – Time Series
- **Friesian** – Recommendation System

End-to-End Distributed AI Pipelines

- **Nano** – Integration and Abstraction of Modern CPU Accelerations
- **Orca** – E2E Distributed AI Pipeline (TensorFlow / PyTorch / OpenVINO / Ray)
- **DLlib** – Distributed Deep Learning Library for Apache Spark

Laptop | K8s | Apache Hadoop/Spark | Ray | Cloud

BigDL 2.0 (https://github.com/intel-analytics/BigDL/) combines the *original BigDL* and *Analytics Zoo* projects

* *"BigDL 2.0: Seamless Scaling of AI Pipelines from Laptops to Distributed Cluster", 2022 Conference on Computer Vision and Pattern Recognition* (CVPR 2022)
* *"BigDL: A Distributed Deep Learning Framework for Big Data", in Proceedings of ACM Symposium on Cloud Computing 2019* (SOCC'19)

BigDL: End-to-End Distributed AI Pipelines

- DLlib: Distributed Deep Learning Library for Spark
- **Orca: Seamlessly Scale E2E AI Pipeline from Laptop to Distributed Cluster**
- **Nano: Automatic Integration of Modern CPU Accelerations for AI**

BigDL-Orca: Building End-to-End Distributed AI Pipeline

E2E, distributed, in-memory pipeline

Data Ingestion → Feature Transformation → Model Training & Evaluation → Model Optimizaiton & Inference

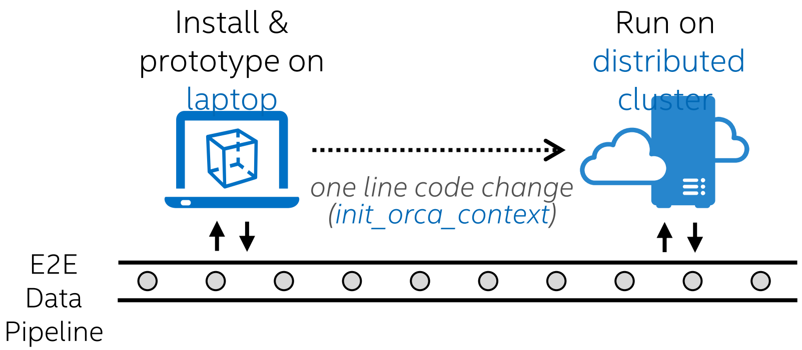
HIVE, Parquet, TFRecord, elasticsearch, amazon S3, redis, ...

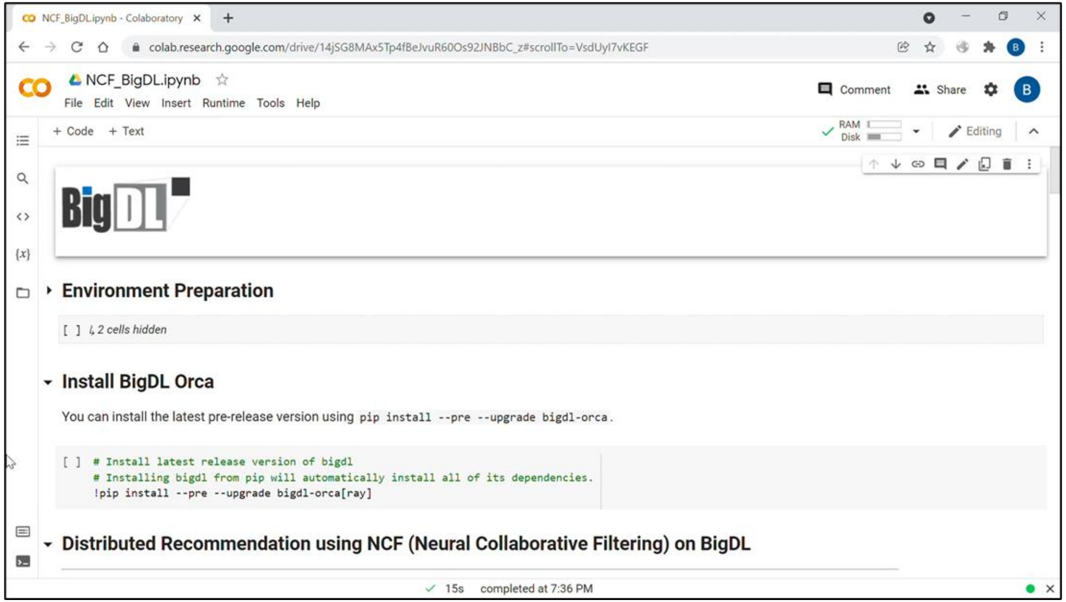
Distributed Data Processing
Spark Dataframe
Ray Dataset
(Distributed) TensorFlow Dataset
(Distributed) PyTorch DataLoader
(Distributed) Python Libraries

Distributed Training
TensorFlow
Keras
PyTorch
MXNet
Horovod
Ray

Distributed Inference
TensorFlow
PyTorch
OpenVINO
ONNX Runtime
INC
Ray

BigDL-Orca: Seamless Scaling from Laptop to Distributed Cluster

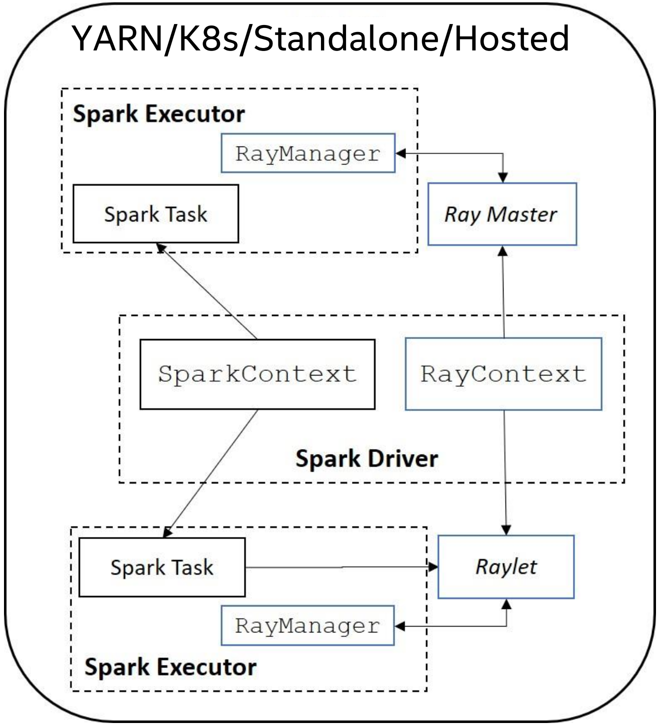




https://bigdl.readthedocs.io/en/latest/doc/UseCase/xshards-pandas.html

Seamless Pipeline between Spark and Ray using Orca

RayOnSpark



Connecting Spark Dataframe & Ray Dataset

```
...

#spark dataframe processing
df = spark.read.csv(...)
train_df = data_process(df)

#connecting spark dataframe to ray dataset
from bigdl.orca.data \
   import spark_df_to_ray_dataset
train_data = spark_df_to_ray_dataset(train_df)

#xgboost on ray
from xgboost_ray import RayDMatrix, train
dtrain = RayDMatrix(train_data, ...)
bst = train(config, dtrain, ...)
```

End-to-End Distributed AI Pipeline Example

```
#1. Initialize OrcaContext
sc = init_orca_context("local", init_ray_on_spark=True)

#2. Distributed data processing using Spark Dataframe
raw_df = spark.read.format("csv").load(data_source_path) \
  .select("Cardholder Last Name", "Cardholder First Initial", \
          "Amount", "Vendor", "Year-Month") \
  ...
#3. Building model using TensorFlow
import tensorflow as tf
model = tf.keras.models.Model(inputs=input, outputs=output)
model.compile(optimizer='rmsprop',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
#4. Distributed training on Orca
from zoo.orca.learn.tf.estimator import Estimator
est = Estimator.from_keras(model, model_dir=args.log_dir)
est.fit(data=trainingDF, batch_size=batch_size, epochs=max_epoch, \
   feature_cols=['features'], label_cols=['labels'], ...)
```

1 Distributed Data processing

Spark Dataframe Ray Dataset	Distributed Python APIs in Orca		
	TensorFlow Dataset, PyTorch DataLoader	PyData (pandas, sklearn, numpy, ...)	PyImage (pillow, opencv, ...)

2 ML/DL Model

Standard TensorFlow/PyTorch APIs for building models

3 Distributed Training (& Inference)

sklearn-style APIs for transparently distributed training & inference

https://github.com/Mastercard/udap-analytic-zoo-examples

Orca AutoML: Distributed HyperParameter Tuning Pipeline using Ray-tune

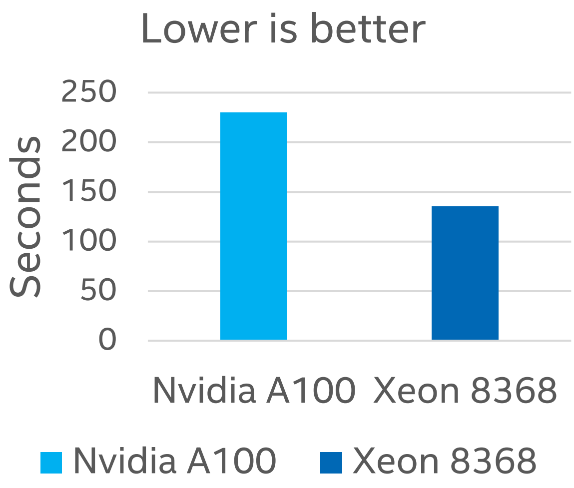
```
from bigdl.orca import init_orca_context
sc = init_orca_context(...,init_ray_on_spark=True)

from bigdl.orca.automl.xgboost import
AutoXGBRegressor
auto_est = AutoXGBRegressor(…)

from bigdl.orca.automl import hp
search_space = {
  "n_estimators": hp.grid_search([50, 100]),
  "max_depth": hp.choice([2, 4, 6])}

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

AutoXGBoost



1.7x Speedup for AutoXGBoost

https://medium.com/intel-analytics-software/scalable-autoxgboost-using-analytics-zoo-automl-30d576cb138a

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

BigDL-Nano: Automatic Integration of Modern CPU Accelerations for TensorFlow & PyTorch

Accelerations	Tuning / Configuration / Tool	Action Needed
Better Memory Allocator	tcmalloc, jemelloc	Download, set environment variables correctly
Proper environment variables	Intel OpenMP (OMP/KMP)	Install, set environment variables correctly
AVX512, BF16	IPEX, Intel Optimized TensorFlow, oneDNN	Install, change some code to use the extensions
Multi-processing	Torch distributed, TF Distributed Strategy, Horovod, etc.	Change some code to use the functionality
Inference Engine	OpenVINO, ONNX Runtime, etc.	Install, Export your model to onnx file, use ort/IE API
Quantization	INC, NNCF/POT	Install, quantize through tool APIs and save/load the quantized model
…	…	…

Nano: Automatic & Transparent Integration of Many IA-Specific Accelerations

```python
# define model
import fastface as ff
model = ff.FaceDetector.build(arch="lffd", config="slim",
                              preprocess=preprocess,
                                     hparams=hparams)
model.add_metric("average_precision",
             ff.metric.AveragePrecision(iou_threshold=0.5))

# define train_loader
train_loader = ff.dataset.FDDBDataset(
    phase="train", transforms=train_transforms
).get_dataloader(batch_size=8, shuffle=True, num_workers=8)

# define Trainer
import pytorch_lightning as pl
trainer = pl.Trainer(...)
```

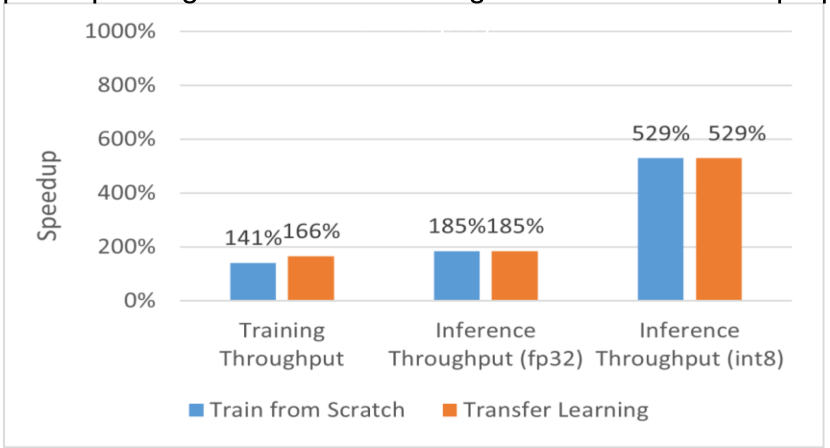
```python
# define Trainer
from bigdl.nano.pytorch.trainer import Trainer
trainer = pl.Trainer(num_processes=4, use_ipex=True, ...)
```

```python
# start training
trainer.fit(model, train_dataloader=train_loader)
```

BigDL-Nano: Automatic Integration of Modern CPU Accelerations for TensorFlow & PyTorch

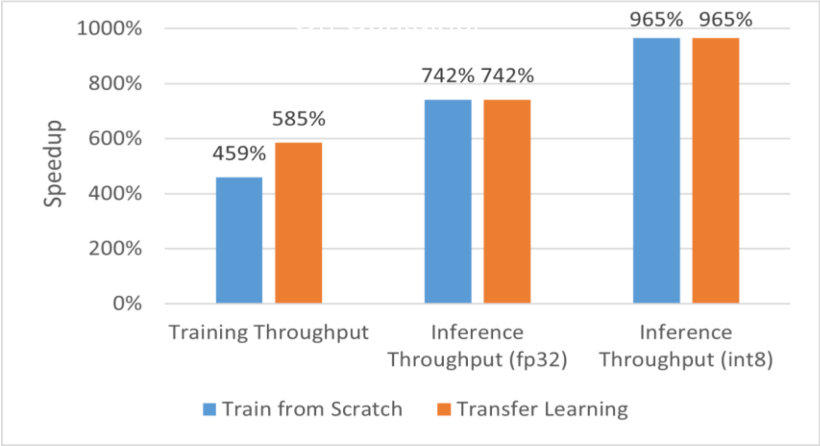
- Substantial speedup
  - Up to ~5.8x training speedup, and ~9.6x inference speedup

Speedup of BigDL-Nano for training and inference on Laptop



laptop - a single 8-core Intel(R) Core (TM) i7-11800H CPU @ 2.30GHz, 12G Memory, Ubuntu 20.04 OS

Speedup of BigDL-Nano for training and inference on Container



container - a docker container with 28 cores in a single socket Intel(R) Xeon(R) Platinum 8380H CPU @ 2.90GHz, 192G memory, Ubuntu 16.04 OS

RAY SUMMIT 22

BigDL 2.0: Seamless Scaling of AI Pipelines from Laptops to Distributed Cluster", CVPR 2022

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

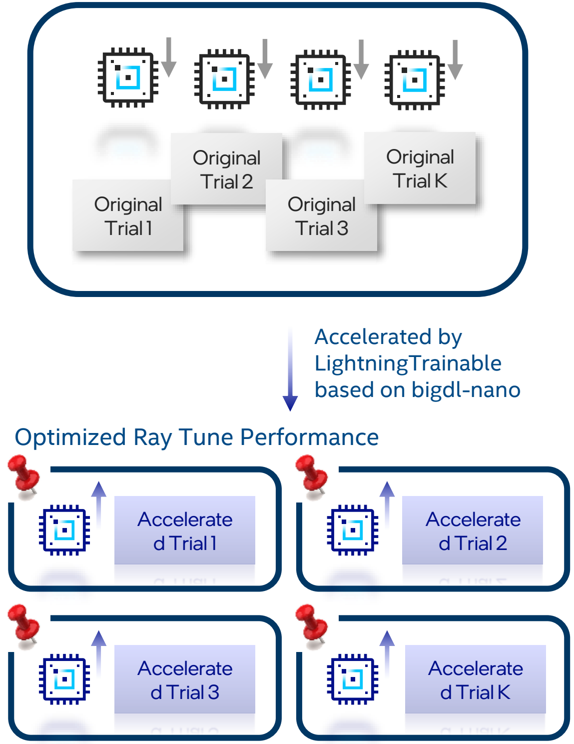
BigDL-Nano: acceleration on Ray-tune

```
class MyTrainable(LightningTrainable):
    cpu_binding = True
    def create_model(self, config):
        return PL_MODEL(config)
    def configrate_trainer(self):
        return {"max_epochs": 2, ...}

analysis = tune.run(MyTrainable, ...,
                    reuse_actor=True)
```

Speedup of BigDL-Nano	Ray Tune default	Ray Tune multi-core
Ray Tune + BigDL-Nano	4.34x	1.20x
Ray Tune + BigDL-Nano + core binding	5.74x	1.59x

- Cat vs dog using ResNet 50, 40 cores requested, 5 actors is used
- Known issue: reuse_actor may not be valid on some cases (~5%)



https://github.com/ray-project/ray/pull/25654

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

BigDL: Domain-Specific Toolkits

- BigDL PPML (Privacy Preserving ML & Data Analytics)
- Project Chronos: Scalable Time Series Toolkit
- Project Friesian: E2E Recommender Toolkit

BigDL PPML (Privacy Preserving ML)

Secure & Trusted Big Data and AI,

Trusted Big Data & AI Apps
- Trusted SQL & Dataframe
- Trusted ML
- Trusted DL
- Trusted FL (Federated Learning)

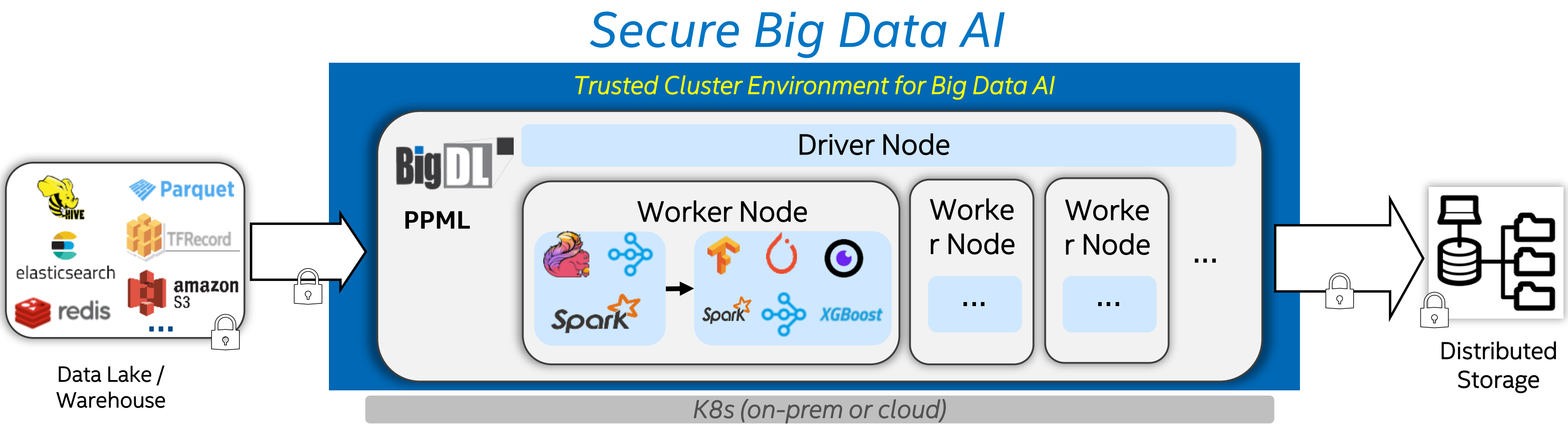
E2E Distributed Pipeline
- **Orca** Distributed AI (TensorFlow/ PyTorch/OpenVINO/Ray) Pipeline
- **DLlib** Distributed Deep Learning Framework for Apache Spark

Library and Framework
- Apache Spark
- Apache Flink
- XGBoost
- Ray
- TensorFlow
- PyTorh
- OpenVINO

Secure Execution Layer
- Secure Storage I/O
- Secure Network I/O
- Secure Data Alignment
- Secure Parameter Sync
- SGX SDK
- Crypto
- LibOS
- Key Mgmt
- Attestation
- Homomorphic Encryption

Intel SGX on kubernetes

BigDL PPML: Trusted Big Data AI



- Standard, distributed AI applications on encrypted data
- Hardware (Intel SGX/TDX) protected computation (and memory)
- End-to-end security enabled for the entire workflow
  - *Provision and attestation of "trusted cluster environment" on K8s (of SGX nodes)*
  - *Secrete key management through KMS for distributed data decryption/encryption*
  - *Secure distributed compute and communication (via SGX, encryption, TLS, etc.)*

BigDL-Chronos

Application framework for scalable time series analysis w/ AutoML

70+ algorithms for distributed time series processing (rolling, scaling, etc.)

TSDataset
with tunable parameters

Model
with tunable parameters

10+ built-in time series models (Prophet, TCN, Seq2Seq, etc.)

Search space

SearchEngine

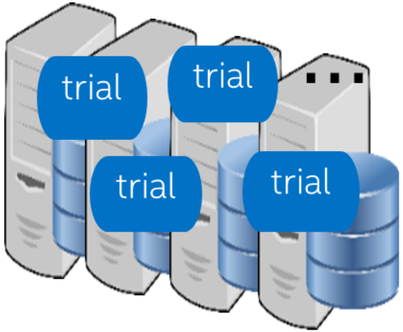
trial jobs

best model /parameters

Pipeline

configured with best parameters/model

Each trial runs a different combination of hyper parameters



Ray Tune

RayOnSpark

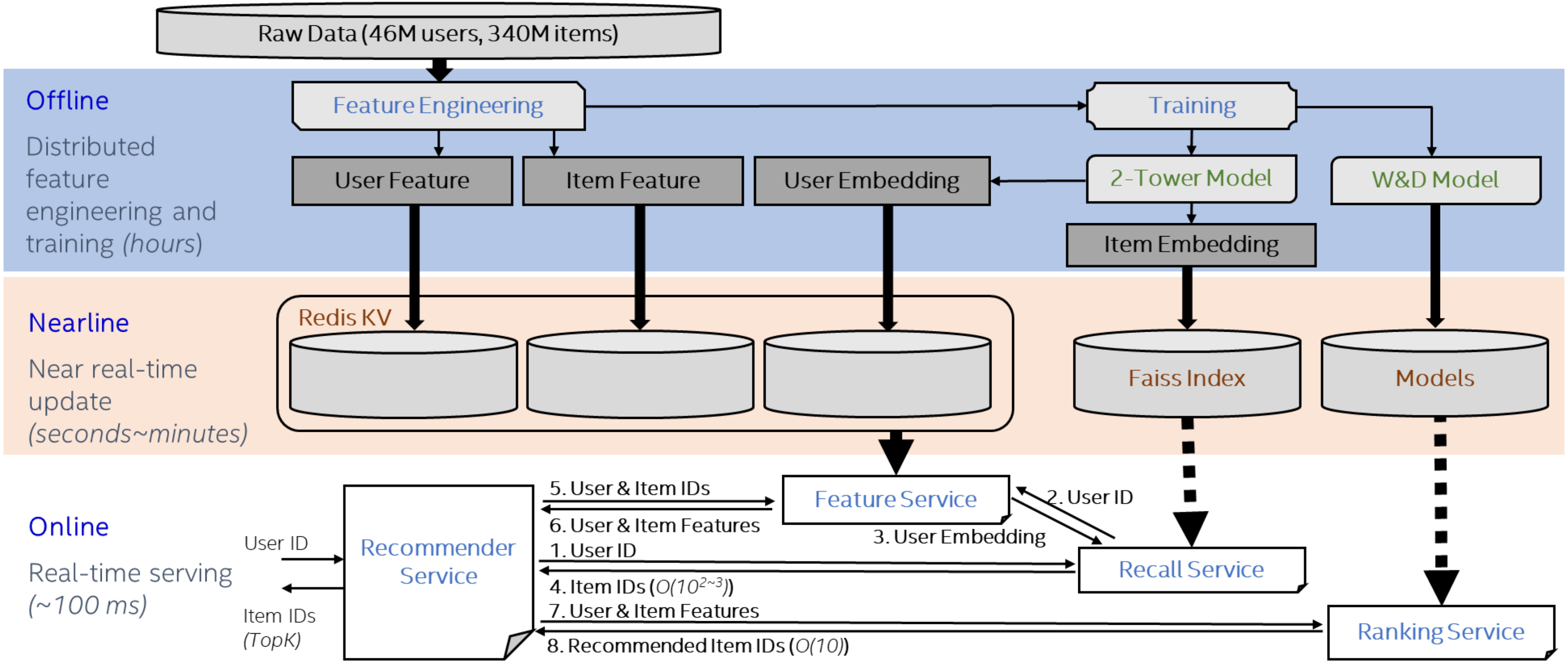
**Joint-collaboration with NEX/NPG for Network AI, CESG/ESS China for AIOps*

* *"Scalable AutoML for Time Series Forecasting using Ray", USENIX OpML'20*

RAY SUMMIT 22

BigDL-Friesian

Application framework for large-scale E2E recommender solution



RAY SUMMIT 22

BigDL-real world End to end Deep Learning examples

CONSUMER

HEALTH

FINANCE

RETAIL

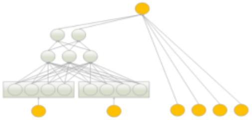
MANUFACTURING

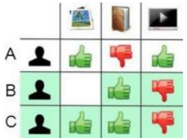
INFRASTRUCTURE

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

“AI at Scale” in Mastercard with BigDL

Transactions → Data ETL → Distributed Training / Tuning → DL Models → Distributed Inference

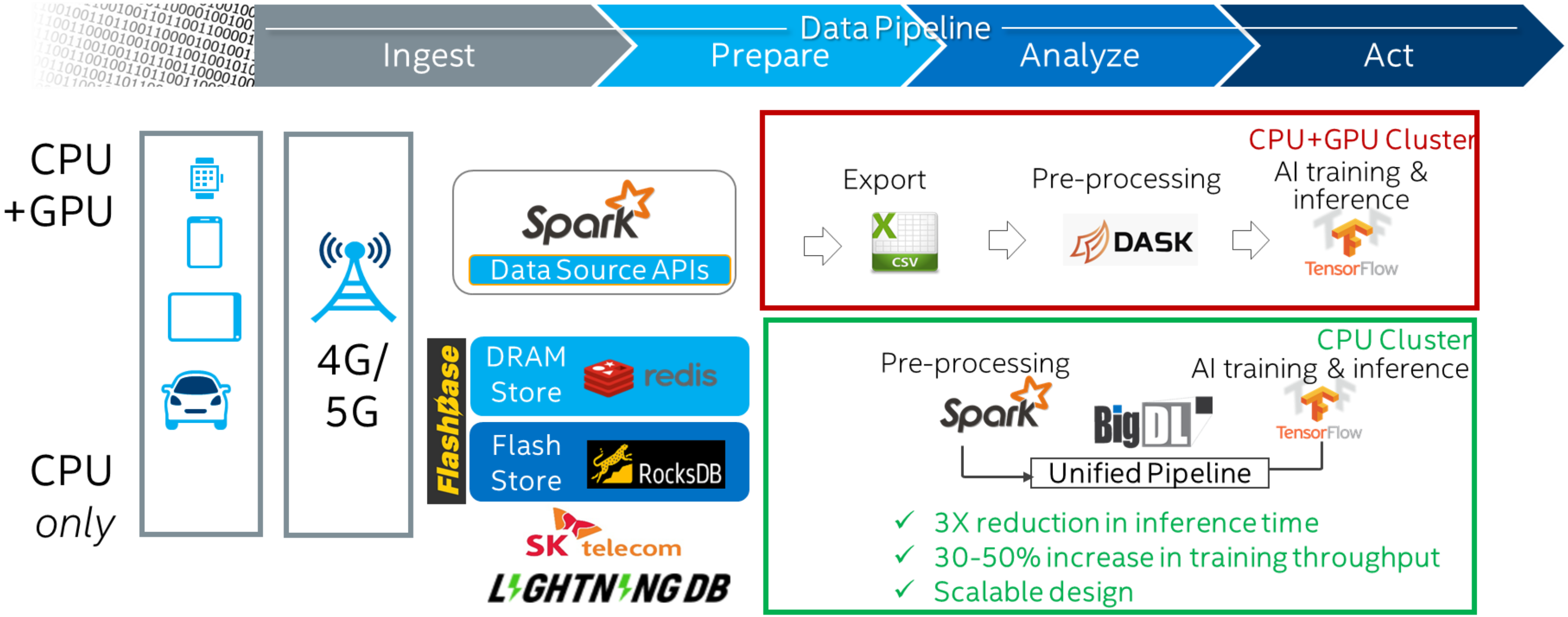




- Building distributed AI applications directly on Enterprise Data Warehouse platform
- Supporting up to 2.2 billion users, 100s of billions of records, and distributed training on several hundred Intel Xeon servers

https://www.intel.com/content/www/us/en/developer/articles/technical/ai-at-scale-in-mastercard-with-bigdl0.html

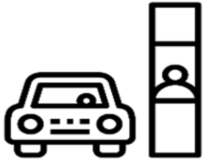
Network Quality Prediction in SK Telecom with BigDL

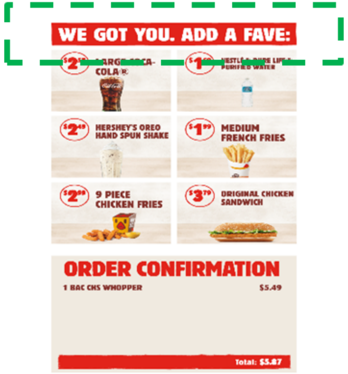


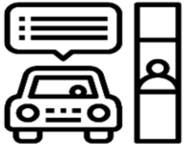
https://networkbuilders.intel.com/solutionslibrary/sk-telecom-intel-build-ai-pipeline-to-improve-network-quality

Fast Food Recommendation in Burger King with BigDL and Ray

Guest arrives ODMB → Checks Menu Board → Cashier enters order → Checks Menu Board → Guest completes order







* *https://medium.com/riselab/context-aware-fast-food-recommendation-at-burger-king-with-rayonspark-2e7a6009dd2d*
* *"Context-aware Fast Food Recommendation with Ray on Apache Spark at Burger King", Data + AI Summit Europe 2020*

Summary

- BigDL 2.0
  - E2E distributed AI pipelines (seamless scaling from laptop to cluster)
  - Domain-specific AI toolkits (PPML, Time Series, Recommender)
  - Real word use cases
- References
  - https://github.com/intel-analytics/bigdl
  - https://jason-dai.github.io/cvpr2022/
  - https://bigdl.readthedocs.io/en/latest/

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 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.