PyCon China 2020

PyConChina 2020
PYTHON 中国开发者大会 2020

Python For Good

Project Orca: Easily scaling Python AI pipelines on big data platforms

Kai Huang

Software Engineer, Intel Corporation

Agenda

- Background
  - AI on Big Data
  - Challenges and Motivations
- Project Orca
  - Distributed training pipeline
  - API design
- Burger King Use Case
- Conclusion

AI on Big Data

Distributed, High-Performance

Deep Learning Framework

for Apache Spark*

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

Unified Analytics + AI Platform

for distributed TensorFlow*, Keras*, and PyTorch* on Apache Spark*/Flink* & Ray

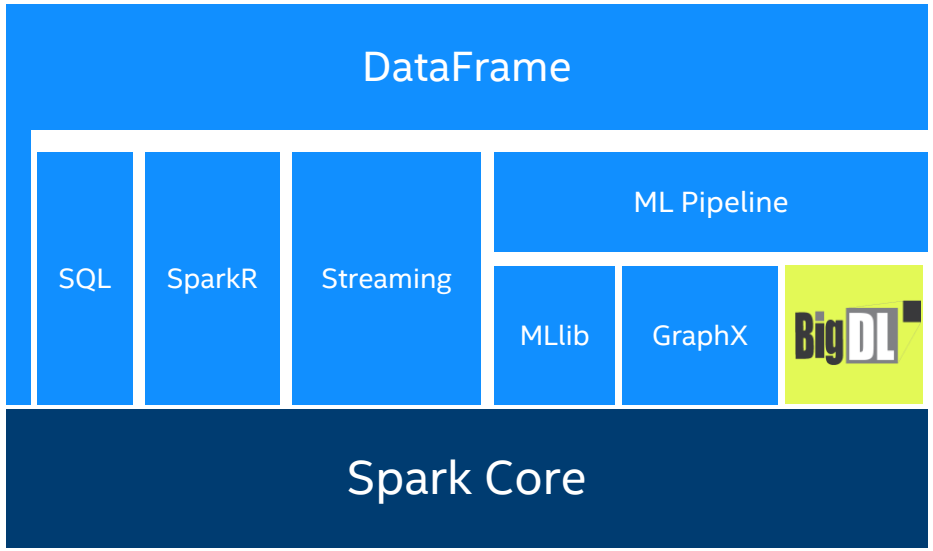
https://github.com/intel-analytics/analytics-zoo

Accelerating Data Analytics + AI Solutions At Scale

BigDL

Bringing Deep Learning to Big Data Platforms

- ***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., TensorFlow, Keras, PyTorch, 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/

Analytics Zoo

Unified Data Analytics and AI Platform for distributed TensorFlow, Keras and PyTorch on Apache Spark/Flink & Ray

ANALYTICS ZOO

Models & Algorithms

- Recommendation
- Time Series
- Computer Vision
- NLP

Automated ML Workflow

- AutoML for Time Series
- Automatic Cluster Serving

Integrated Analytics & AI Pipelines

- Distributed TensorFlow & PyTorch on Spark
- RayOnSpark
- Spark DataFrames & ML Pipelines for DL
- InferenceModel

Compute Environment

- Laptop
- K8s Cluster
- Hadoop Cluster
- Spark Cluster

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

Powered by oneAPI

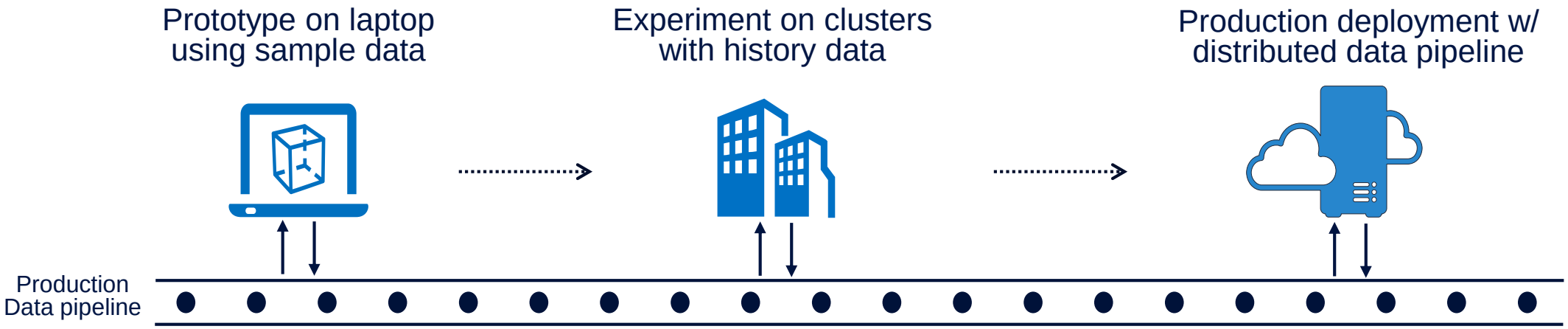
https://github.com/intel-analytics/analytics-zoo https://analytics-zoo.github.io/

Motivations for Project Orca

- Most AI projects start with a Python notebook running on a single laptop; however, one usually needs to go through a mountain of pains to scale it to handle larger data set in a distributed fashion.
- Gap between deep learning frameworks and big data systems.
- Challenge to prepare the Python environment on each node without modifying the cluster.

Objectives for Project Orca

Seamless Scaling from Laptop to Distributed Big Data Clusters



- 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.

Project Orca

We develop *Project Orca* in Analytics Zoo based on Spark and Ray to allow users to easily scale out single node Python notebook across large clusters, by providing:

- Data-parallel preprocessing for Python AI (supporting common Python libraries such as Pandas, Numpy, PIL, TensorFlow Dataset, PyTorch DataLoader, etc.)
- Sklearn-style APIs for transparently distributed training and inference (supporting TensorFlow, PyTorch, Keras, MXNet, Horovod, etc.)

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

https://analytics-zoo.github.io/master/#Orca/overview/

Ray

Ray is a fast and simple framework for building and running distributed applications.

- Ray Core provides easy Python interface for parallelism by using remote functions and actors.

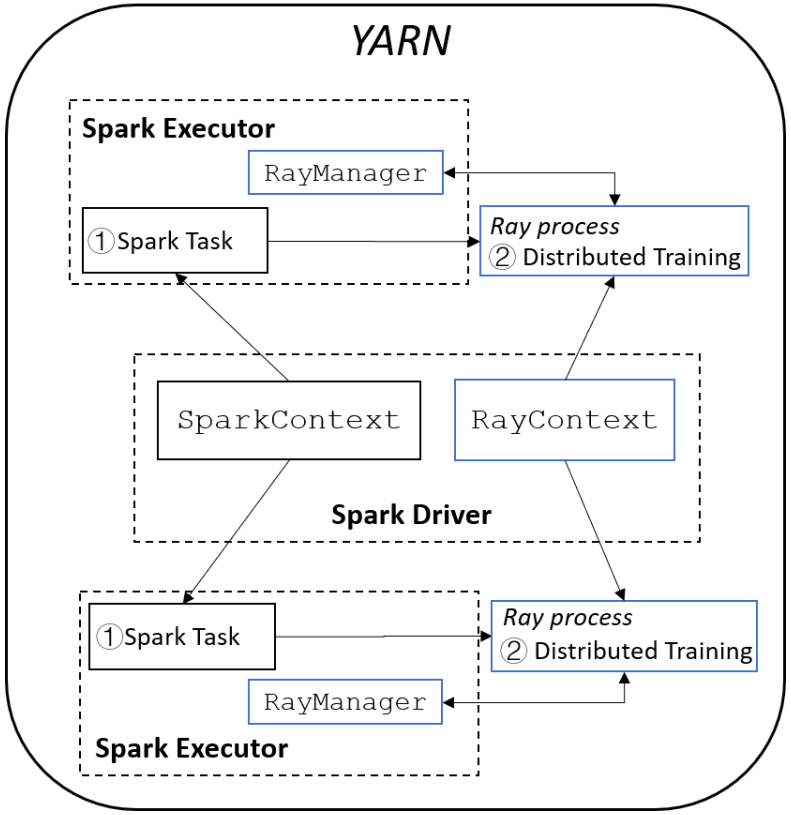
Ray is packaged with several high-level libraries to accelerate machine learning workloads.

- Tune: Scalable Experiment Execution and Hyperparameter Tuning
- RLlib: Scalable Reinforcement Learning
- RaySGD: Distributed Training Wrappers for TensorFlow and PyTorch
- https://github.com/ray-project/ray/

Distributed Training Pipeline

We use *RayOnSpark* to seamlessly integrate Ray applications into Spark data processing pipelines.

- Runtime cluster environment preparation.
- Create a *SparkContext* on the drive node and use Spark to perform data related tasks.
- *RayContext* on Spark driver launches Ray across the cluster.
- Use Ray to implement a lightweight shim layer for deep learning frameworks to make deployment on big data clusters easy.
- The worker on each node takes the local data partitions of Spark from the plasma object store used by Ray.



Project Orca API

- Minimum code changes and learning efforts are needed to scale the Python AI application from single node to big data clusters.

```
from zoo.orca import init_orca_context
import zoo.orca.data.pandas

# init_orca_context unifies SparkContext and RayContext
sc = init_orca_context(cluster_mode="yarn", num_nodes, cores, memory)

# Data loading and preprocessing.
shards = zoo.orca.data.pandas.read_csv(path)  #or read_json
data = shards.transform_shard(preprocess_func)
# Can also directly use Spark RDD/DataFrame, TensorFlow Dataset, PyTorch DataLoader, etc as data.
```

Project Orca API

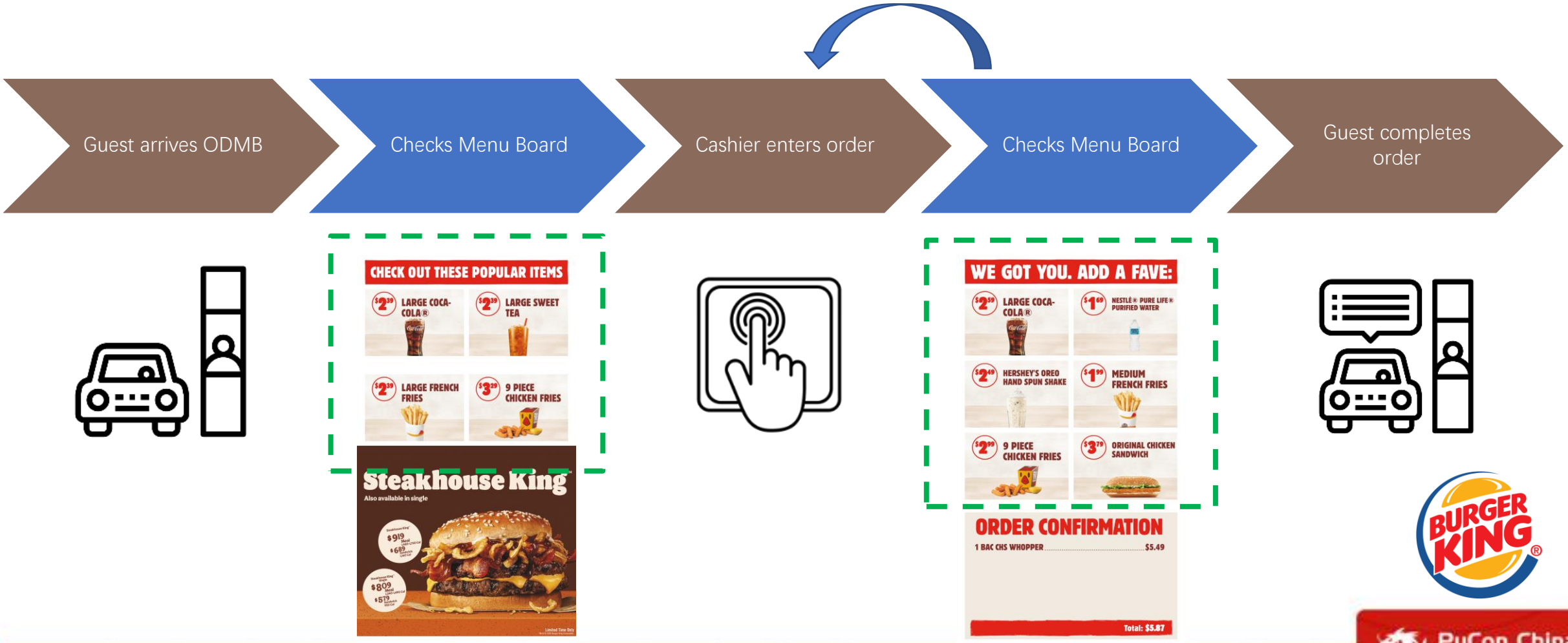
- The entire pipeline runs on a single cluster. No extra data transfer needed.


```
from zoo.orca.learn.pytorch import Estimator

estimator = Estimator.from_torch(model, optimizer, loss, ...)

estimator.fit(data, val_data, batch_size, epochs, ...)
```

```
from zoo.orca.learn.tf import Estimator

# For tf.keras users
estimator = Estimator.from_keras(compiled_keras_model, ...)

# For graph users
estimator = Estimator.from_graph(inputs, outputs, labels,
                                 outputs, loss, optimizer, ...)

estimator.fit(data, val_data, batch_size, epochs, ...)
```

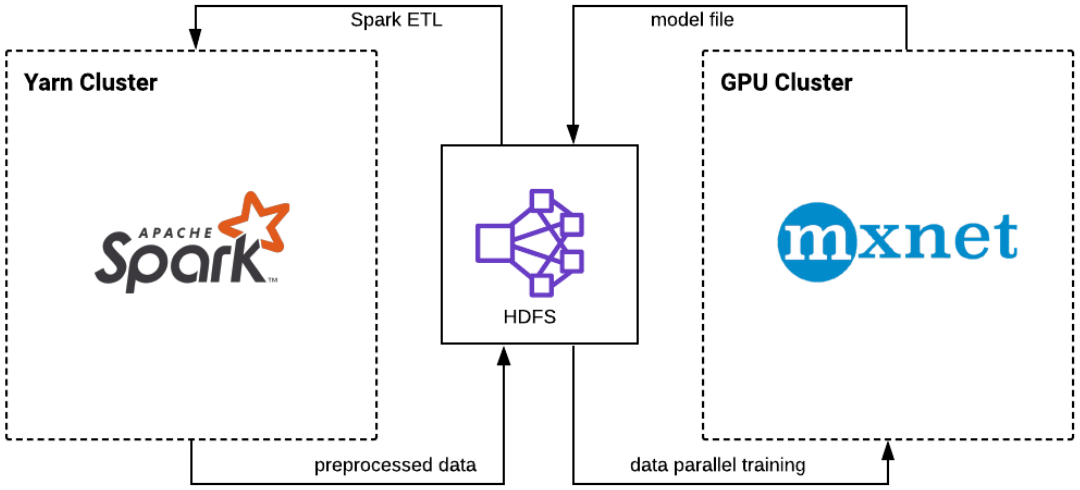
Recommendation System at Burger King

- Drive-thru, mobile app and web browse recommendation use cases.

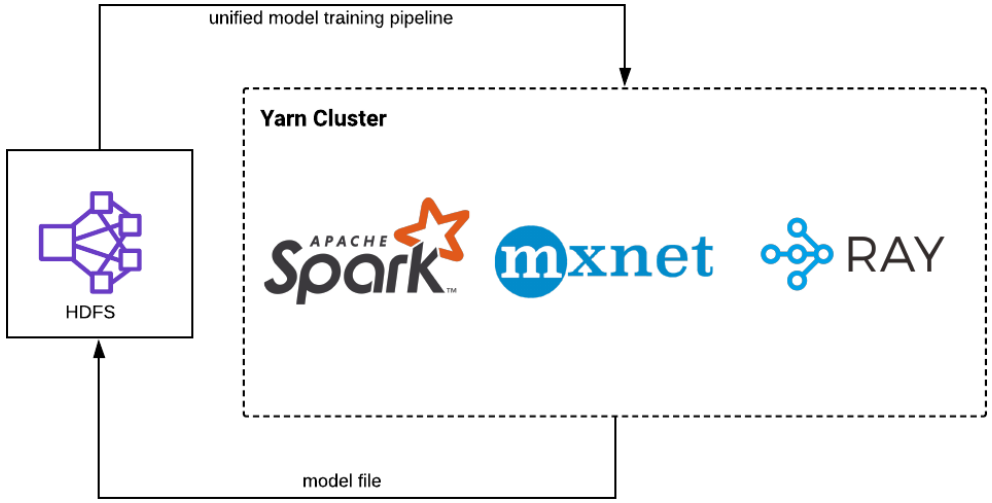


Recommendation System at Burger King

Previous



Current



Recommendation System at Burger King

- Burger King performs Spark ETL tasks first, followed by distributed MXNet training.
- Similar to RaySGD, MXNet Estimator in Project Orca implements a lightweight shim layer around native MXNet modules for easy deployment on YARN cluster.
- Project Orca eliminates the extra data transfer and cluster management overhead.

```
from zoo.orca import init_orca_context
from zoo.orca.learn.mxnet import Estimator

# init_orca_context unifies SparkContext and RayContext
sc = init_orca_context(cluster_mode="yarn", num_nodes, cores, memory)
# Use sc to load data and do data preprocessing.

mxnet_estimator = Estimator(train_config, model=txt, loss=SoftmaxCrossEntropyLoss(),
                            metrics=[mx.metric.Accuracy(), mx.metric.TopKAccuracy(3)])

mxnet_estimator.fit(data=train_rdd, validation_data=val_rdd, epochs=…, batch_size=…)
```

Conclusion

- Project Orca makes scaling Python AI pipelines from single node to large clusters easy.

- More information for Analytics Zoo at:

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

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

- More information for Burger King use case at:

https://arxiv.org/abs/2010.06197

https://medium.com/riselab/context-aware-fast-food-recommendation-at-burger-king-with-rayonspark-2e7a6009dd2d

PyCon China 2020

Python For Good

PyConChina 2020 | PYTHON 中国开发者大会 2020

THANK YOU

ANALYTICS ZOO