GOTC

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

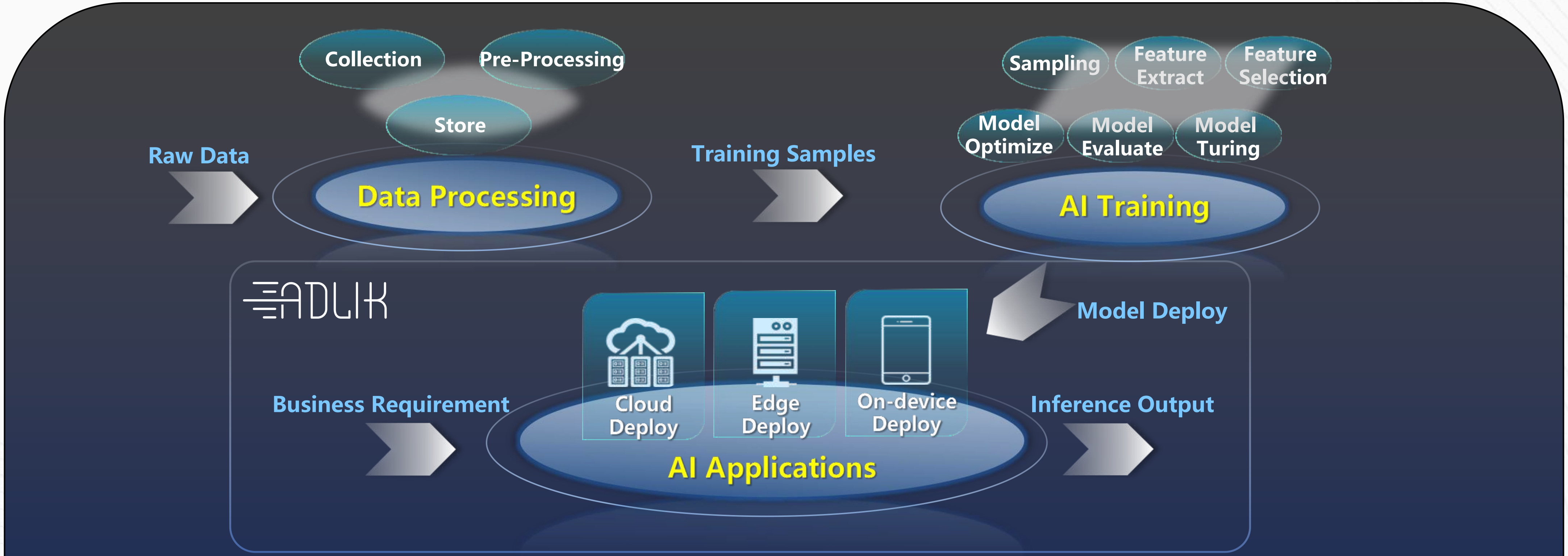
OPEN SOURCE , OPEN WORLD

「AI、大数据与数字经济论坛」专场

本期议题：Adlik对深度学习模型推理优化的实践

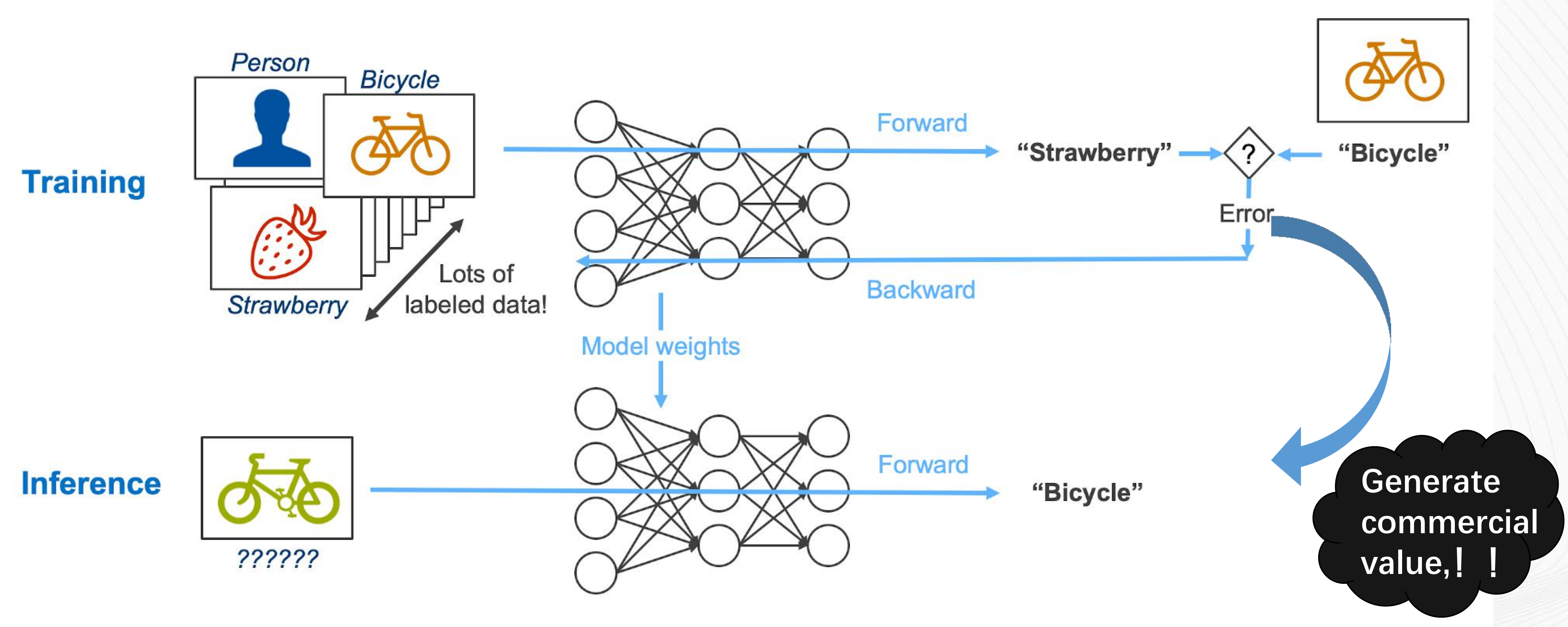
刘涛 2021年07月10日

Background: Three Big Stages in Machine Learning Pipeline



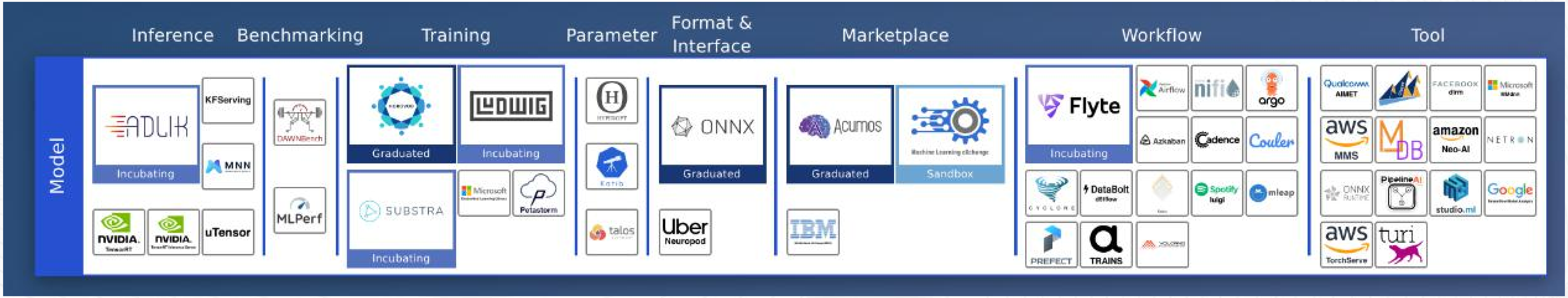
There are still many technical difficulties (both deployment and inferernce) in AI application stage that Adlik can help to solve.

Background: Inference



What's Adlik

- Adlik [ædlik], a toolkit for accelerating deep learning inference on specific hardware.
- Support several kinds of hardwares.
- Collaborate with existing inference solutions with unified entrance.
- An open source project of LFAI and code hosted on GitHub. https://github.com/Adlik

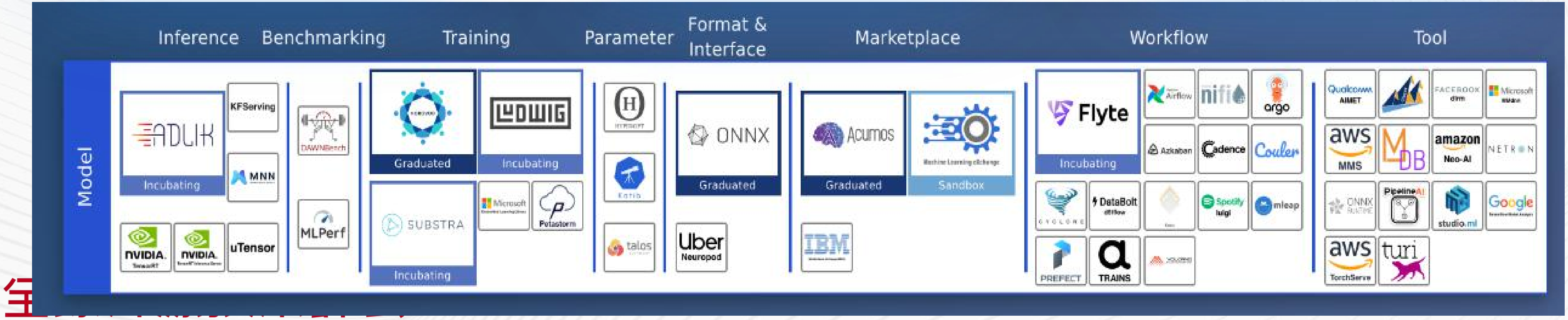


What's Adlik

Adlik [ædlik], a toolkit for accelerating deep learning inference on specific hardware.

- Support several kinds of hardwares.
- Collaborate with existing inference solutions with unified entrance.

An open source project of LFAI and code hosted on GitHub. https://github.com/Adlik



Why Using Adlik

Efficient

- Directly using training framework to do inference will be inefficient.
- Meet performance requirements (latency、throughput).

Convenient

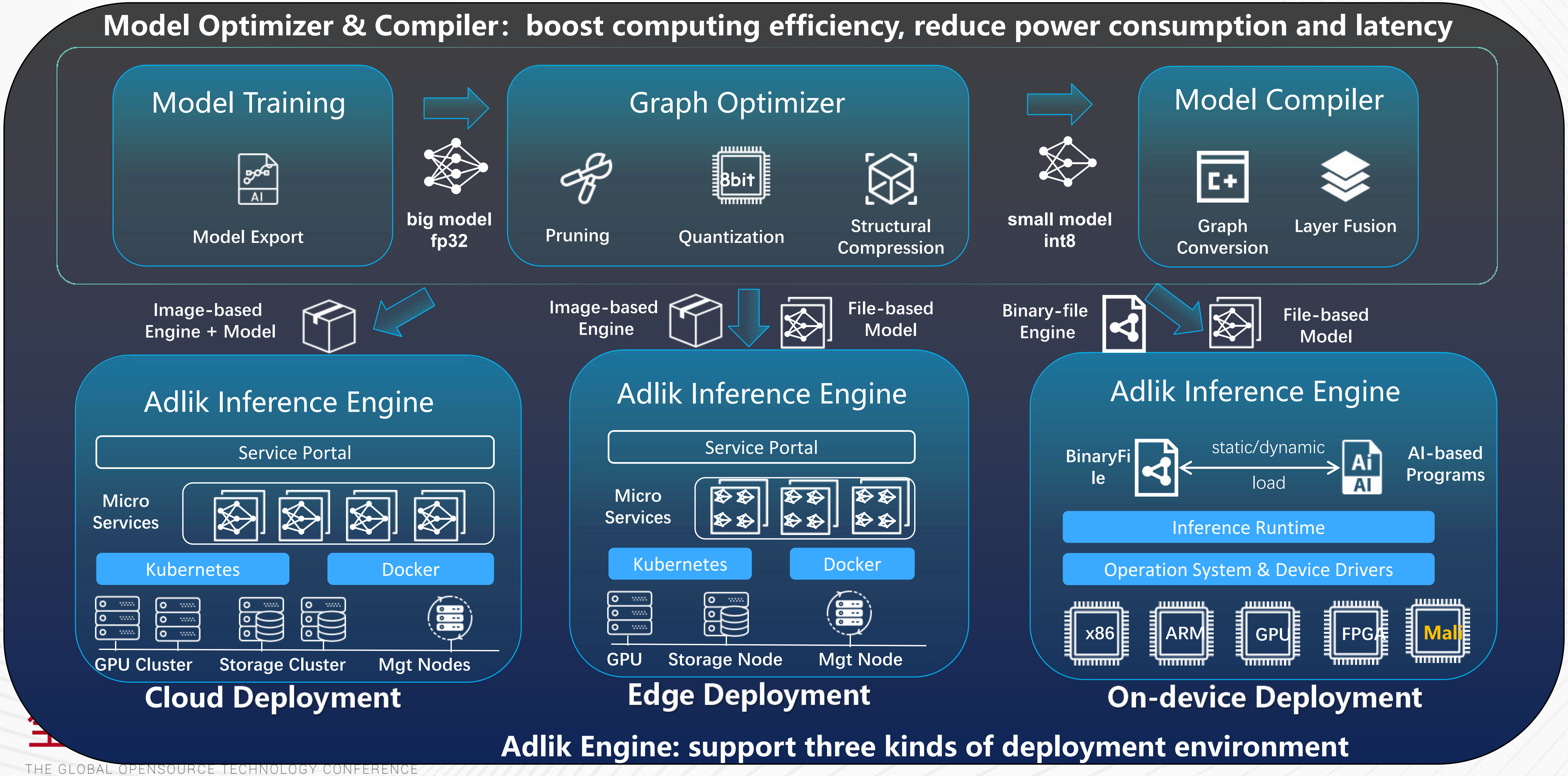
- Convenient to use in different deployment scenario and specific hardware.
- Easy for user to choose correct inference params to get ideal performance in specific hardware.

Portable

- Adaptive for different hardwares.
- Uniform interface for model compiler and optimizer.
- Unified inference interface and model management.

Adlik Architecture

GOTC



Model Optimizer & Compiler: boost computing efficiency, reduce power consumption and latency

Model Training

Model Export

big model fp32

Graph Optimizer

Pruning

Quantization

Structural Compression

small model int8

Model Compiler

Graph Conversion

Layer Fusion

Image-based Engine + Model

Image-based Engine

File-based Model

Binary-file Engine

File-based Model

Adlik Inference Engine

Service Portal

Micro Services

Kubernetes

Docker

GPU Cluster

Storage Cluster

Mgt Nodes

Cloud Deployment

Adlik Inference Engine

Service Portal

Micro Services

Kubernetes

Docker

GPU

Storage Node

Mgt Node

Edge Deployment

Adlik Inference Engine

BinaryFile

static/dynamic load

AI-based Programs

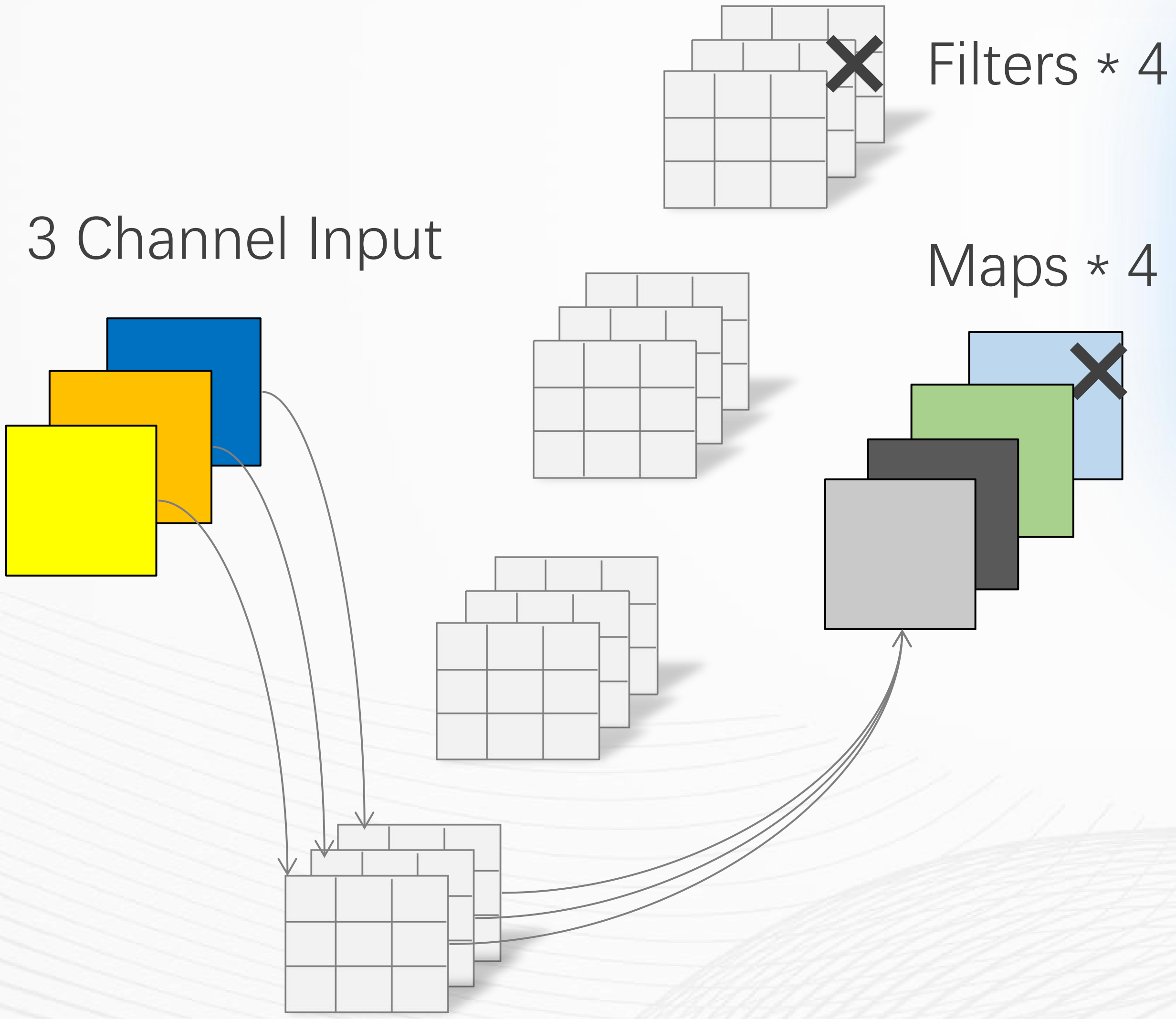
Inference Runtime

Operation System & Device Drivers

On-device Deployment

Adlik Engine: support three kinds of deployment environment

Adlik Feature: Model Optimizer, Pruning

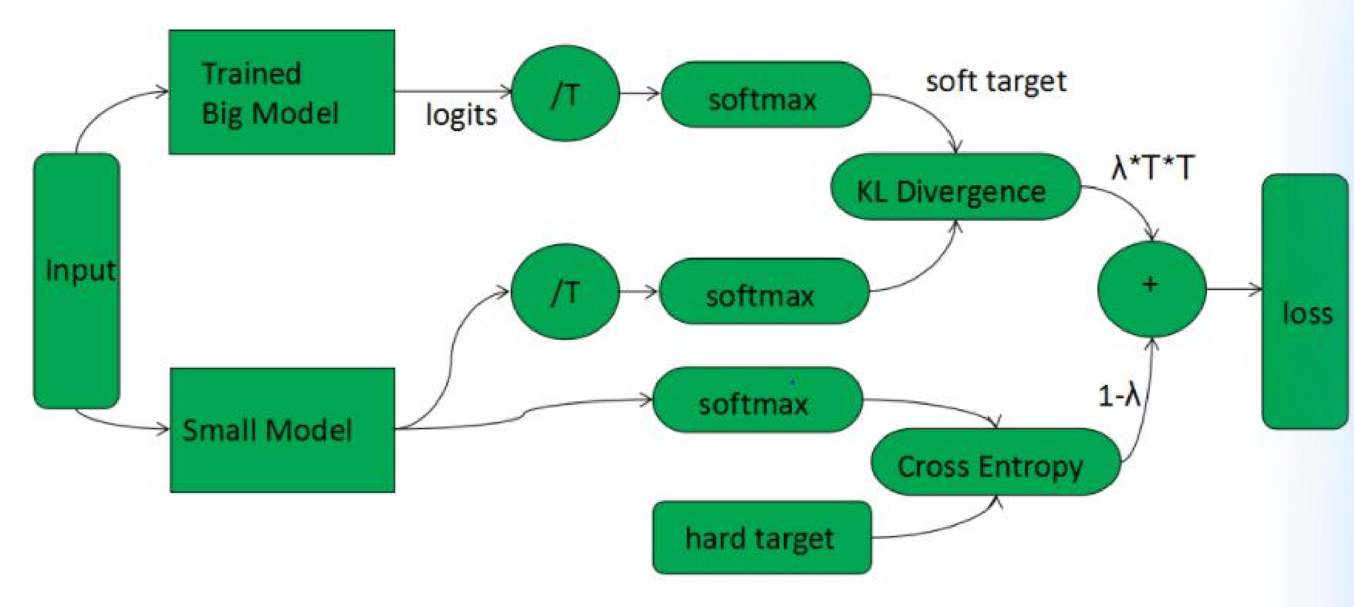


- Supporting multi-nodes and multi-GPU pruning and tuning.
- Supporting channel pruning and filter pruning, reducing the number of parameters and flops.

ResNet-50	Top-1	Parameters	Size
baseline	76.19%	25.61M	99MB
pruned	75.50%	17.43M	67MB

ResNet-50	MACs	Inference speed
baseline	$5.10*10^7$	7.2 pcs/s
pruned	$3.47*10^7$	9.57 pcs/s

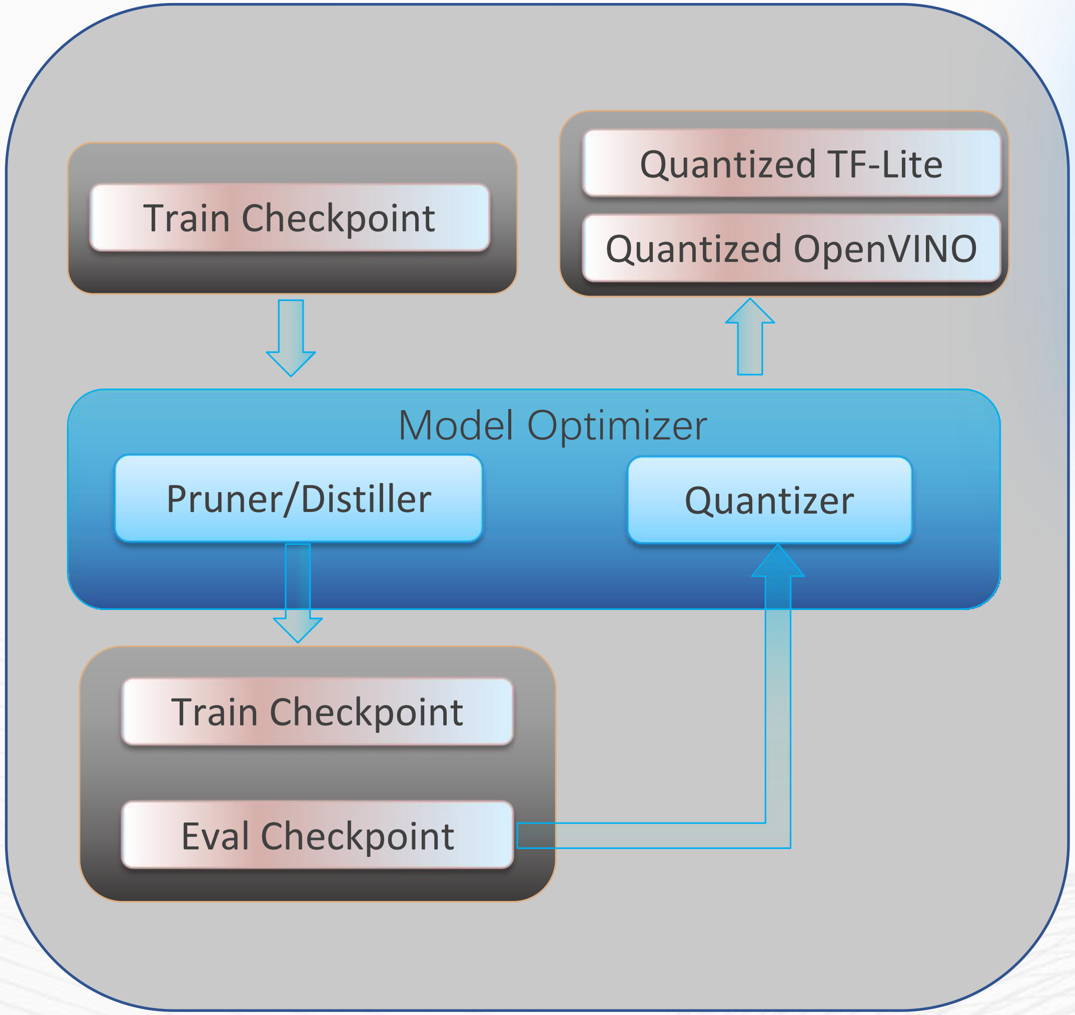
Adlik Feature: Model Optimizer, Knowledge Distillation



Reduce the scale of the small model, and decrease the number of parameters and flops.

Increase the performance of the small model.

Adlik Feature: Model Optimizer



- Supporting combined distillation, which greatly improves the accuracy of the model
- Supporting 8-bit Calibration Quantization. Quantizing process needs only a small batch of datasets and few minutes.

	Params	Flops	Accuracy	Size
ResNet-50	25610152	3899M	76.174%	99M
+ pruned(72.8%)	6954152	1075M	72.28%	27M
+ distill	6954152	1075M	76.39%	27M
+ quantize			75.938%	7.1M

Model Optimizer Result: 7.1/99 = 7.2%

Adlik Feature: Model Optimizer

Inference Benchmark Result:

- Based on MLPerf SingleStream Mode

ResNet-50	FP32	INT8	FP32_pruned	INT8_pruned
Latency(ms)	6.74	2.82	3.32	1.34

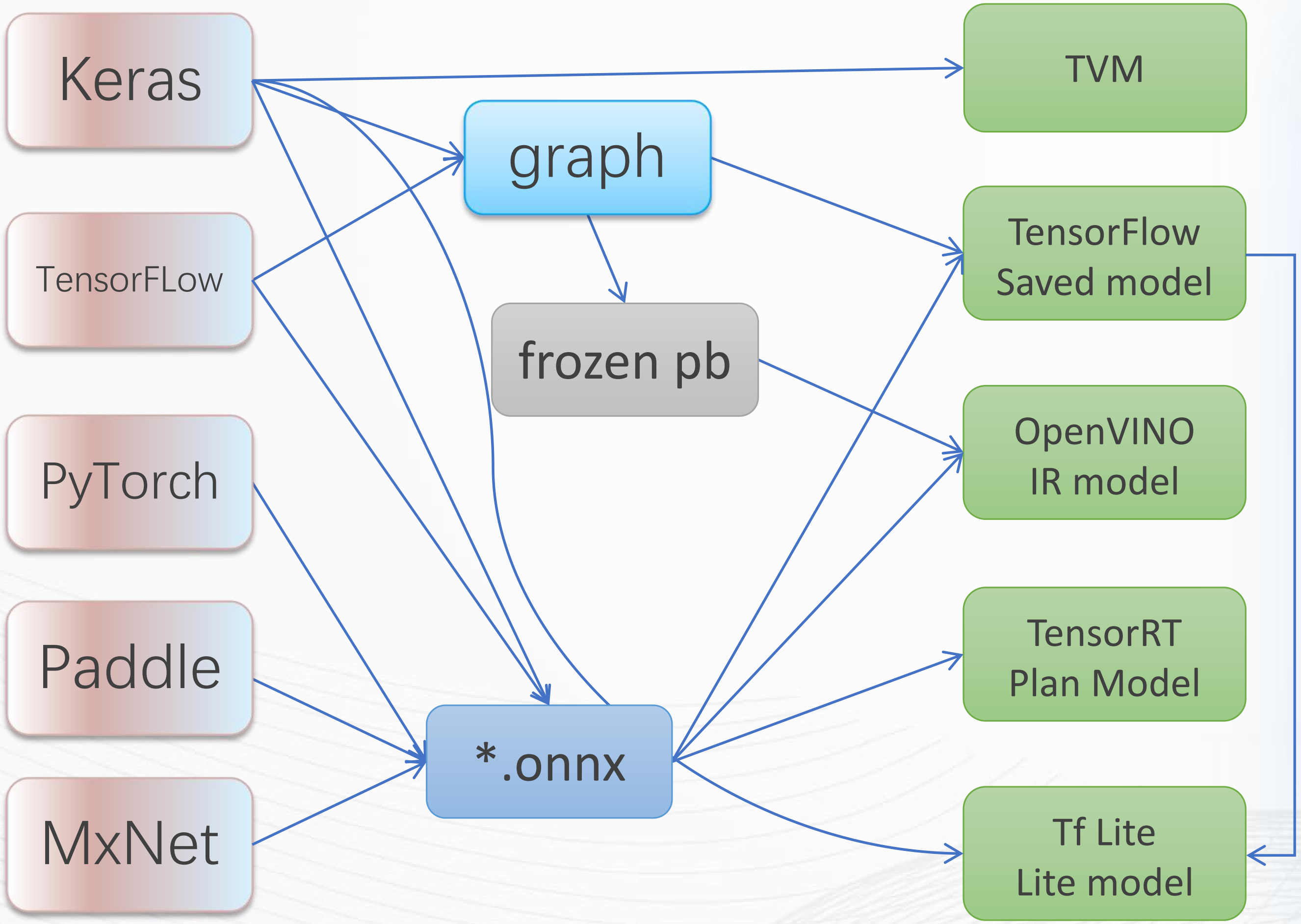
Batch size: 1, ZXCLOUD R5300 G4; Intel(R) Xeon(R) Platinum 8260 CPU @2.40GHz

- Based on OpenVINO Benchmark

	ResNet-50	FP32	INT8	FP32_pruned	INT8_pruned
Async Mode	Latency(ms)	22.56	6.35	6.63	2.09
	FPS	526.83	1863.60	1782.49	5685.45
Sync Mode	Latency(ms)	5.24	1.82	2.45	1.28
	FPS	190.73	549.93	408.03	781.56

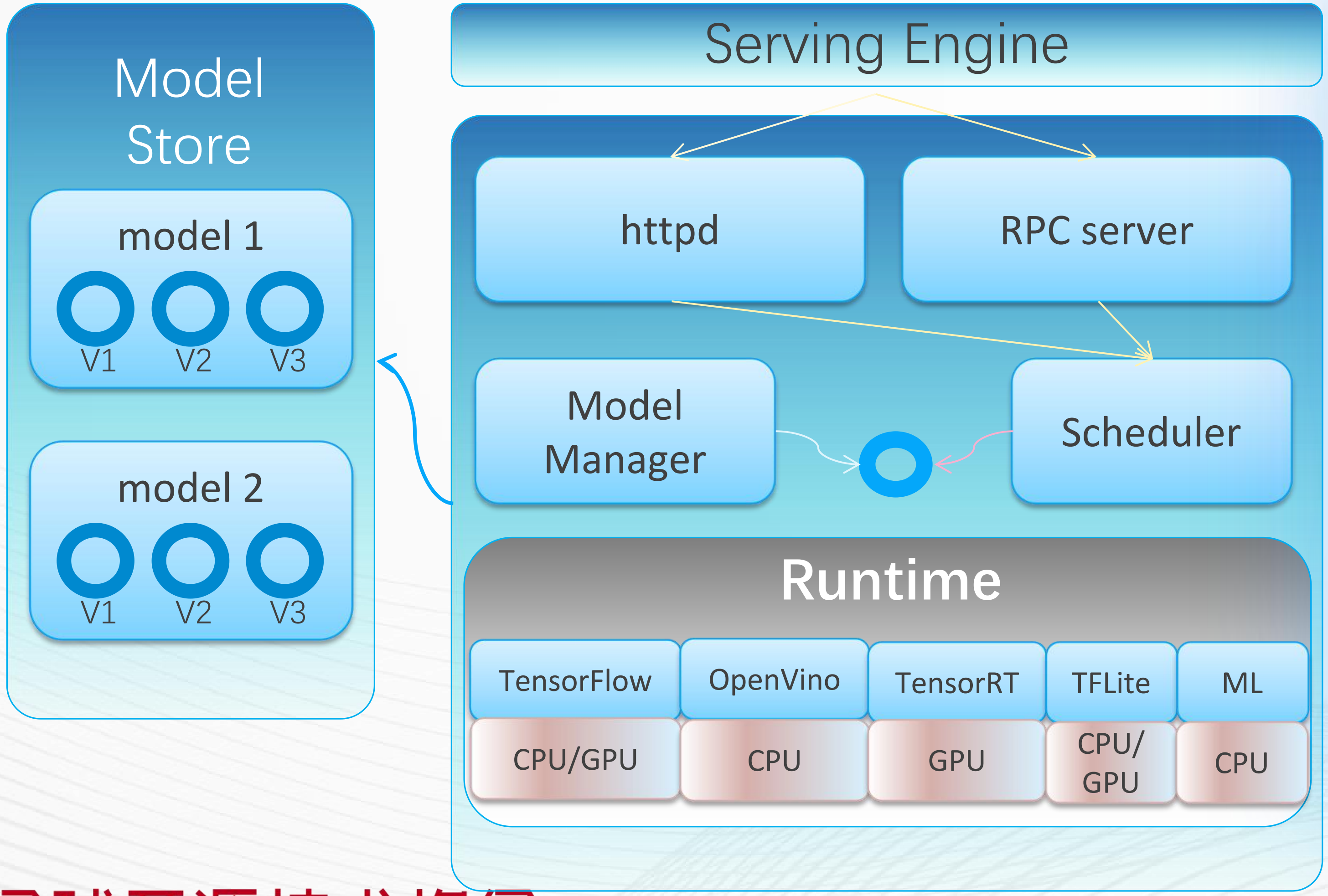
Batch size: 1, ZXCLOUD R5300 G4; Intel(R) Xeon(R) Platinum 8260 CPU @2.40GHz

Adlik Feature: Model Compiler



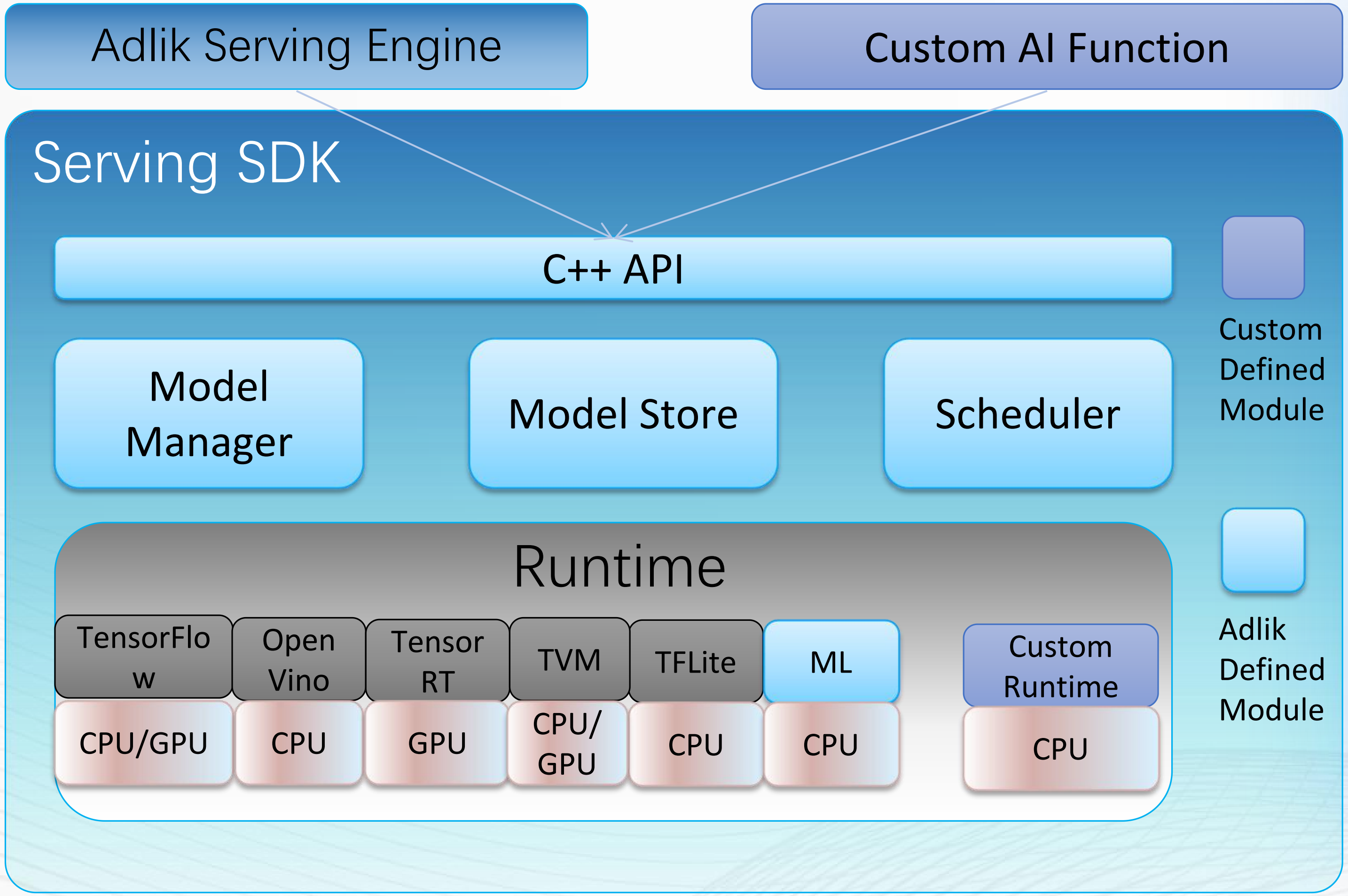
- Support several original trained model formats and target runtime formats with unified compiling request.
- Support DAG generation for end-to-end compilation of models with different representation.
- Support model quantization for TfLite, TensorRT, OpenVINO.

Adlik Feature: Adlik Inference Engine



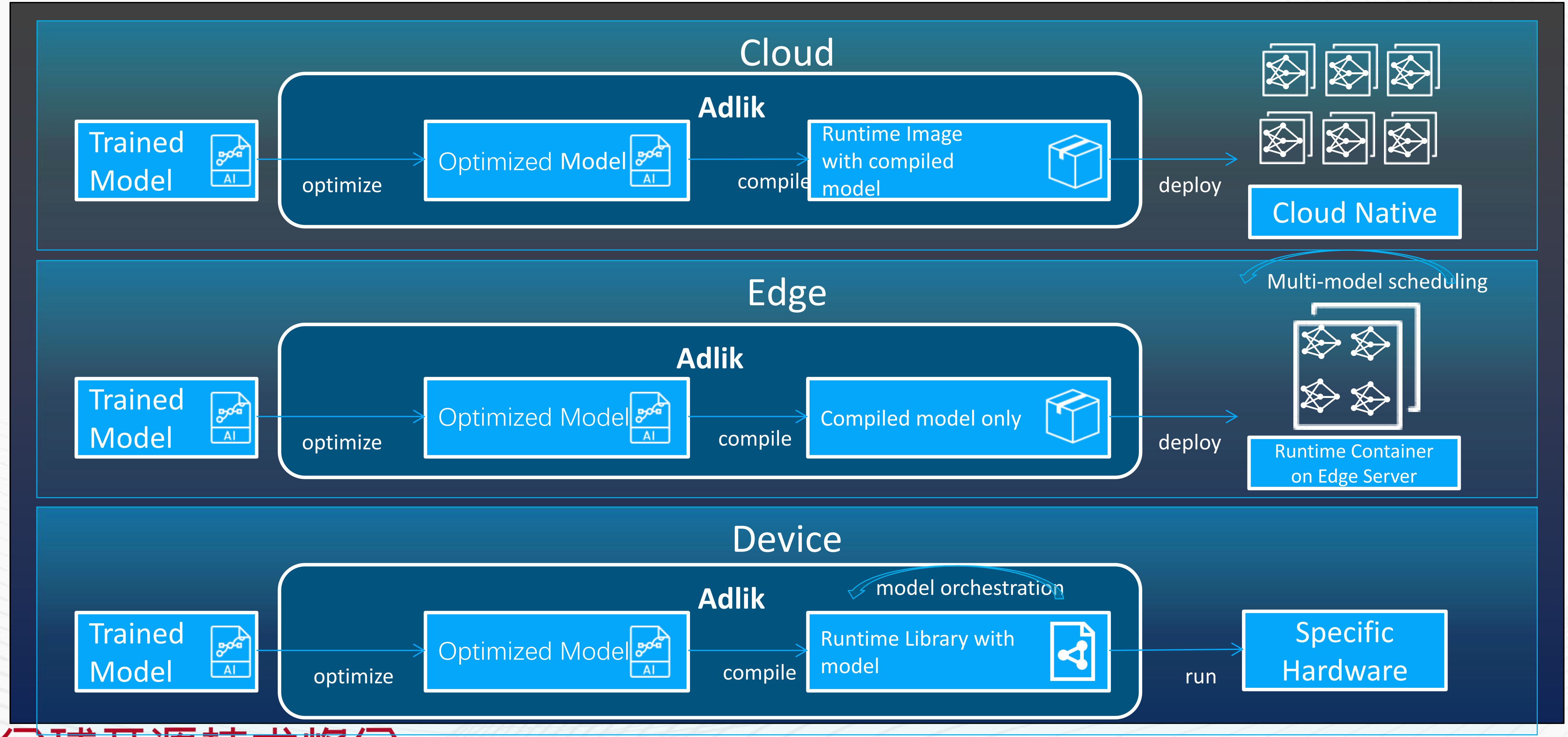
- Model upload, upgrade, versioning, inference and monitoring
- Unified inference interface
- Unified management and scheduling of multi-runtime, multi-model and multi-instance
- Supporting custom-defined runtime
- Supporting ML runtime

Adlik Feature: Adlik Serving SDK



- C++ API
- Supporting custom defined runtime
- Supporting custom defined Ops
- Supporting model orchestration
- Easy for users to expand their own runtime

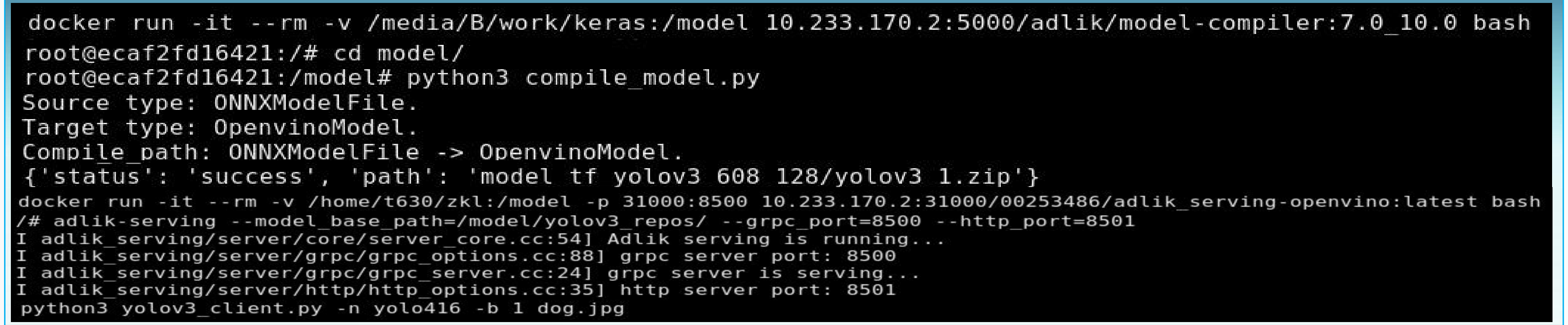
Using Adlik to Deploy Models in Cloud/Edge/Device



Usecase: Adlik in Cloud Native

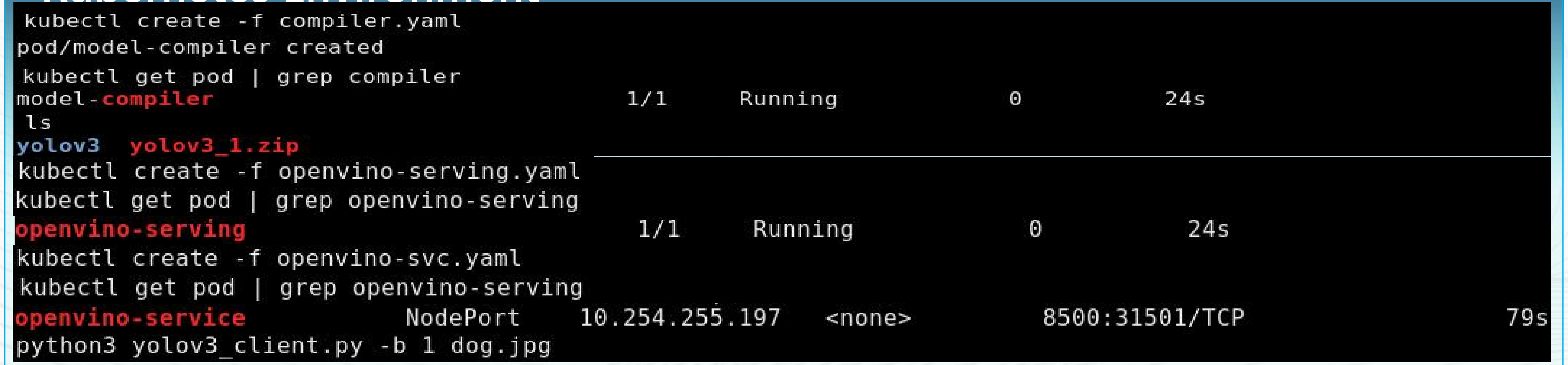
GOTC

Docker Environment

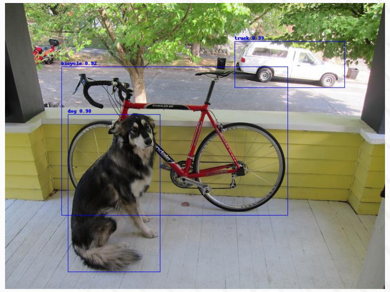


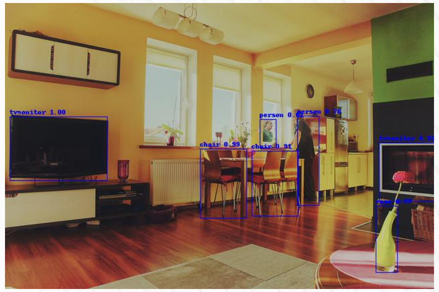
```
 docker run -it --rm -v /media/B/work/keras:/model 10.233.170.2:5000/adlik/model-compiler:7.0_10.0 bash
root@ecaf2fd16421:/# cd model/
root@ecaf2fd16421:/model# python3 compile_model.py
Source type: ONNXModelFile.
Target type: OpenvinoModel.
Compile_path: ONNXModelFile -> OpenvinoModel.
{'status': 'success', 'path': 'model tf yolov3 608 128/yolov3 1.zip'}
docker run -it --rm -v /home/t630/zkl:/model -p 31000:8500 10.233.170.2:31000/00253486/adlik_serving-openvino:latest bash
/# adlik-serving --model_base_path=/model/yolov3_repos/ --grpc_port=8500 --http_port=8501
I adlik_serving/server/core/server_core.cc:54] Adlik serving is running...
I adlik_serving/server/grpc/grpc_options.cc:88] grpc server port: 8500
I adlik_serving/server/grpc/grpc_server.cc:24] grpc server is serving...
I adlik_serving/server/http/http_options.cc:35] http server port: 8501
python3 yolov3_client.py -n yolo416 -b 1 dog.jpg
```

Kubernetes Environment



```
kubectl create -f compiler.yaml
pod/model-compiler created
kubectl get pod | grep compiler
model-compiler                          1/1     Running            0            24s
ls
yolov3  yolov3_1.zip
kubectl create -f openvino-serving.yaml
kubectl get pod | grep openvino-serving
openvino-serving                         1/1     Running            0            24s
kubectl create -f openvino-svc.yaml
kubectl get pod | grep openvino-serving
openvino-service          NodePort    10.254.255.197    <none>          8500:31501/TCP                 79s
python3 yolov3_client.py -b 1 dog.jpg
```

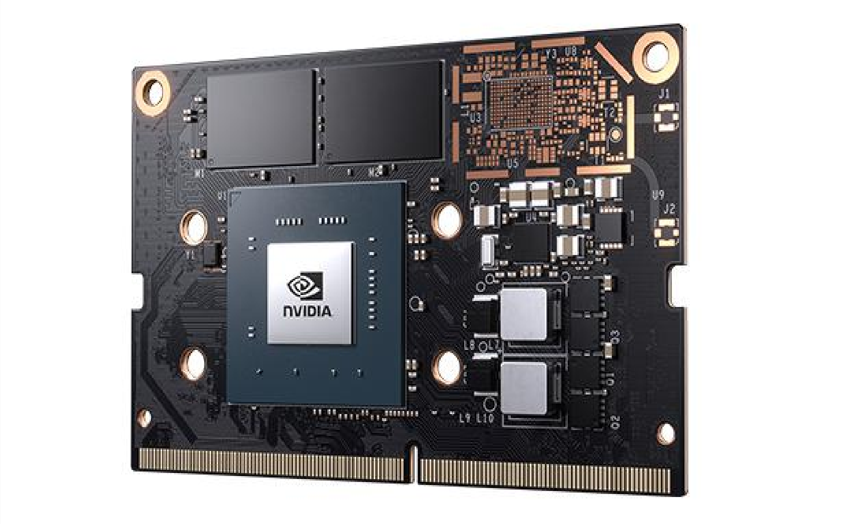




全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

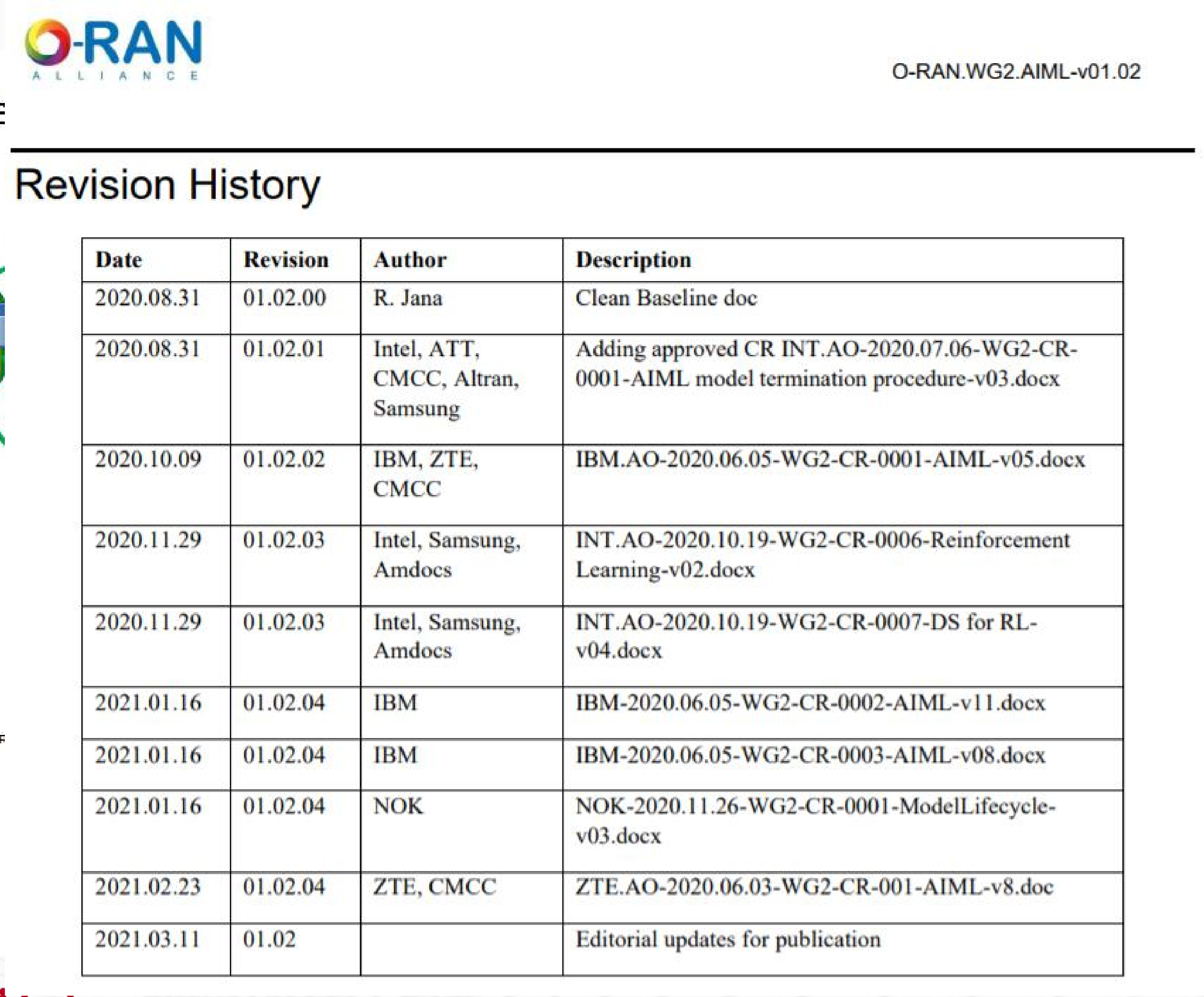
Usecase: Adlik used in embedded device



- Deploy Adlik inference engine in Jetson Nano and Raspberrt Pi.
- Use Adlik optimizer to quantize Resnet-50, Inception V3, and compile it to TfLite model format.
- In device, we read test images locally and run inference test by calling Adlik inference interface.

Usecase: Adlik for O-RAN

GOTC

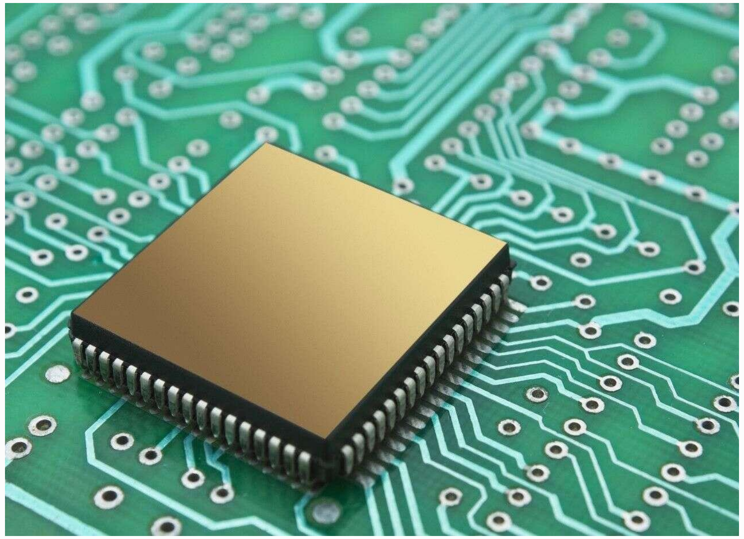

O-RAN ALLIANCE

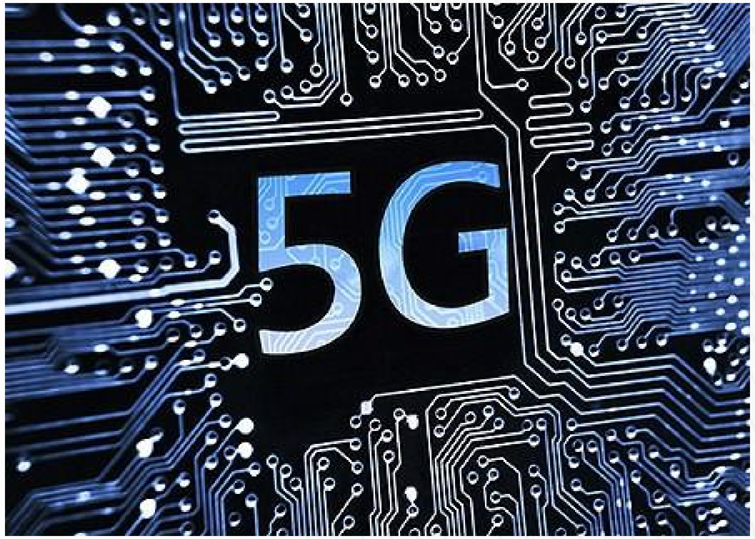
O-RAN.WG2.AIML-v01.02

Revision History

Date	Revision	Author	Description
2020.08.31	01.02.00	R. Jana	Clean Baseline doc
2020.08.31	01.02.01	Intel, ATT, CMCC, Altran, Samsung	Adding approved CR INT.AO-2020.07.06-WG2-CR-0001-AIML model termination procedure-v03.docx
2020.10.09	01.02.02	IBM, ZTE, CMCC	IBM.AO-2020.06.05-WG2-CR-0001-AIML-v05.docx
2020.11.29	01.02.03	Intel, Samsung, Amdocs	INT.AO-2020.10.19-WG2-CR-0006-Reinforcement Learning-v02.docx
2020.11.29	01.02.03	Intel, Samsung, Amdocs	INT.AO-2020.10.19-WG2-CR-0007-DS for RL-v04.docx
2021.01.16	01.02.04	IBM	IBM-2020.06.05-WG2-CR-0002-AIML-v11.docx
2021.01.16	01.02.04	IBM	IBM-2020.06.05-WG2-CR-0003-AIML-v08.docx
2021.01.16	01.02.04	NOK	NOK-2020.11.26-WG2-CR-0001-ModelLifecycle-v03.docx
2021.02.23	01.02.04	ZTE, CMCC	ZTE.AO-2020.06.03-WG2-CR-001-AIML-v8.doc
2021.03.11	01.02		Editorial updates for publication

Challenge in AI Inference





- Fast Inference Speed
- Lightweight
- AI in 5G, Fast and Lightweight

Adlik Practice: Model Graph Optimization

BN Fold

before

$$z = W * x + b$$

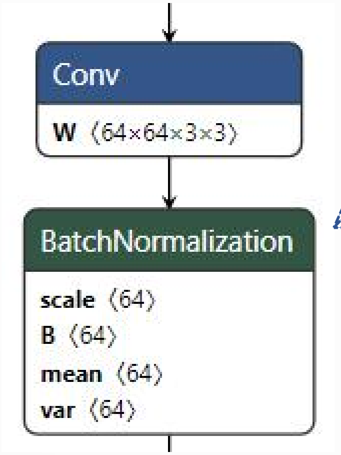
$$\text{out} = \gamma \cdot \frac{z-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta$$

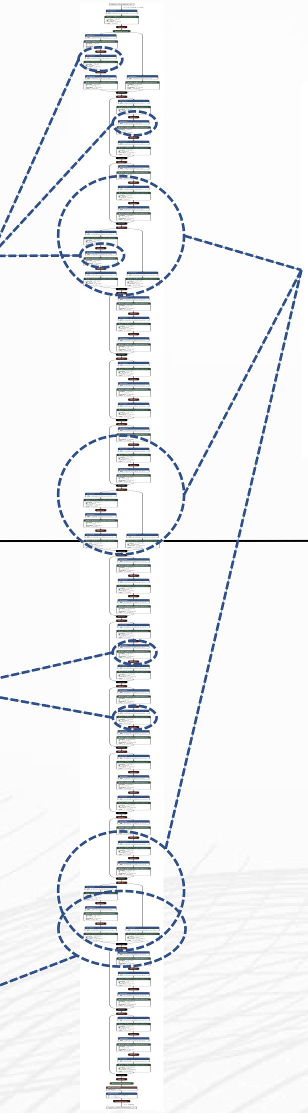
$$\downarrow$$

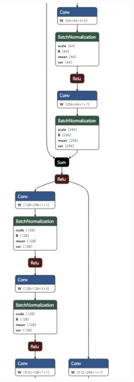
after

$$w_{\text{fold}} = \gamma \cdot \frac{W}{\sqrt{\sigma^2+\epsilon}}$$

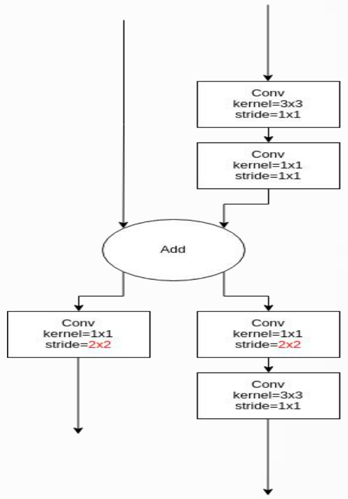
$$b_{\text{fold}} = \gamma \cdot \frac{b-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta$$

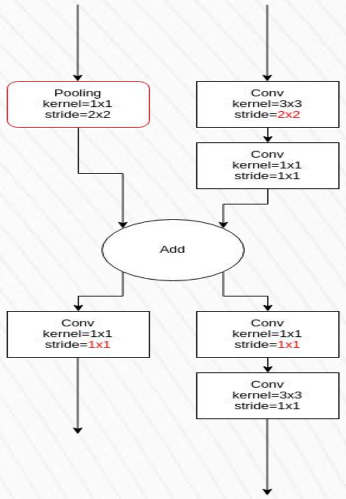


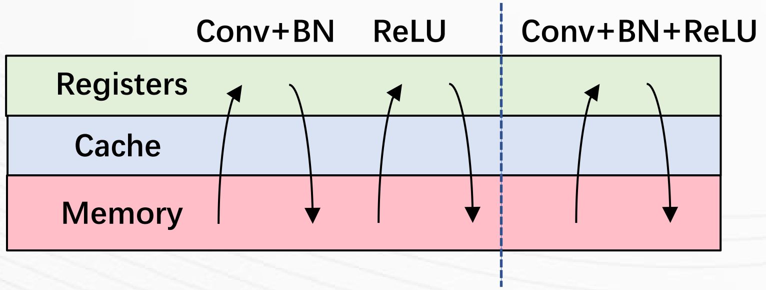


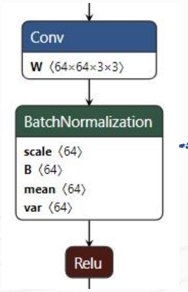


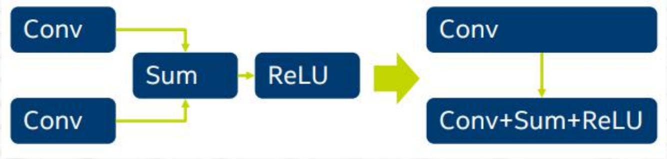
Stride Optimization（Resnet-specific）

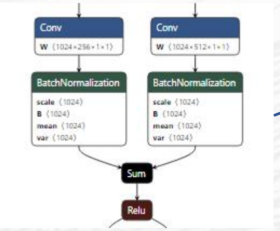












Layer Fusion

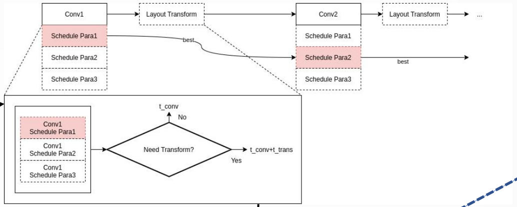
	Inference Latency(ms)	Improvment
Benchmark	12.09	-
Constant Fold (Conv+BN)	9.87	18.39%
Layer Fusion	7.81	20.85%
Stride Optimization	6.7	14.24%

ZXCLOUD R5300 G4; Intel(R) Xeon(R) Platinum 8260 CPU @2.40GHz

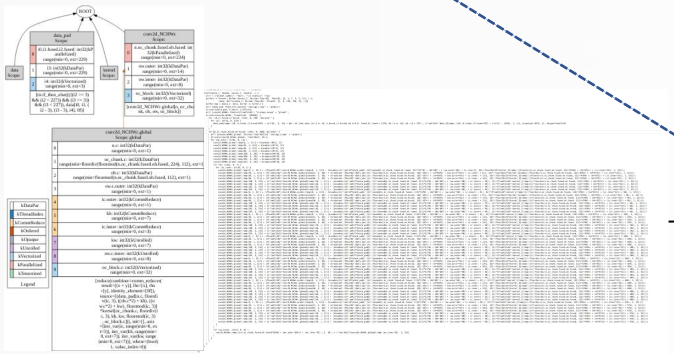
Adlik Practice: Operator Schedule Optimization

Step1: Schedule parameter optimization for single op

Step 2：Schedule parameter optimization in graph view



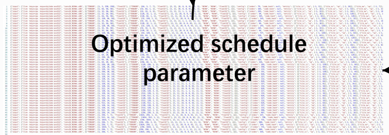
- Generate schedule based on schedule parameters and template
- Execute the program and collect performance information using VTune (Hostpot,/Memory Consumption/Memory Access)



Operator Schedule Template

Schedule Template

Schedule Parameters Optimization Algorithm

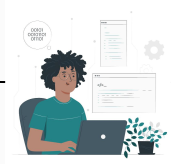


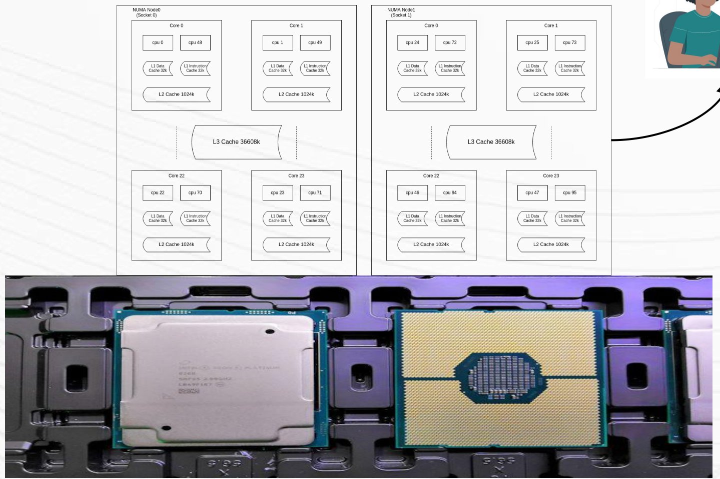
Optimized schedule parameter

Schedule Parameters Search

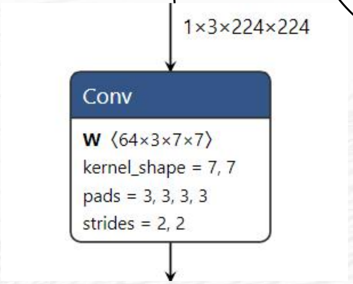
Schedule Template Design

Schedule Performance Analysis

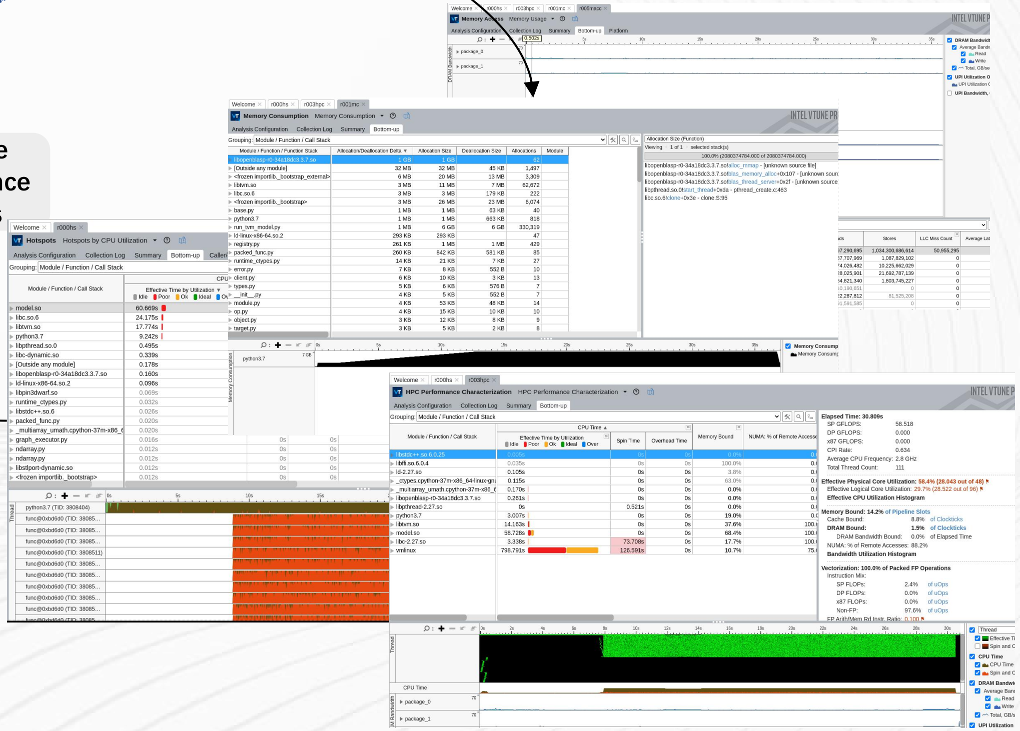




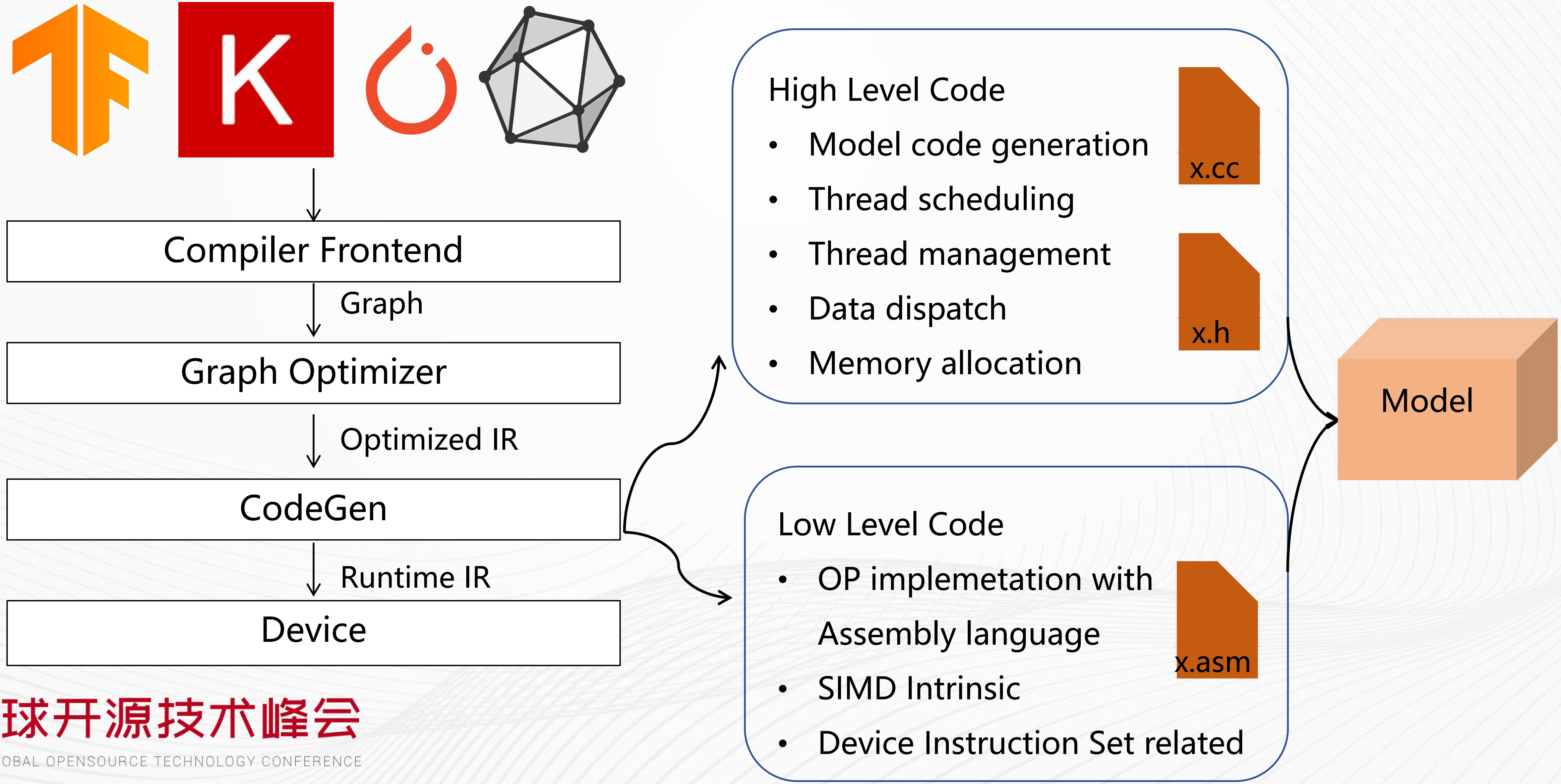
CPU Micro Architecture



operator workload



Adlik Practice: Compiling Process



Compiler Frontend

Graph

Graph Optimizer

Optimized IR

CodeGen

Runtime IR

Device

High Level Code

- Model code generation
- Thread scheduling
- Thread management
- Data dispatch
- Memory allocation

x.cc

x.h

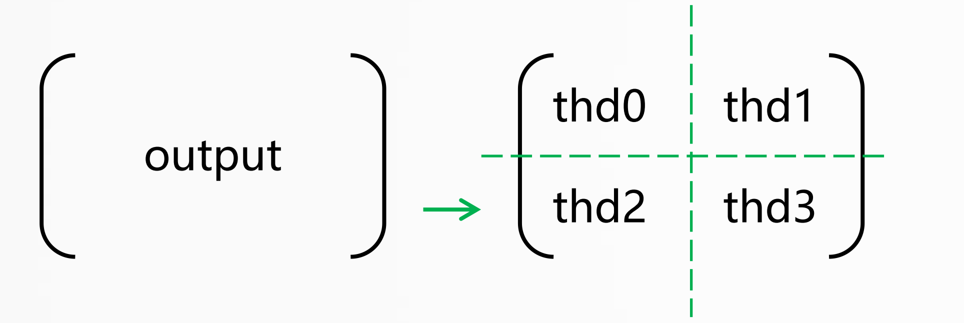
Low Level Code

- OP implemetation with Assembly language
- SIMD Intrinsic
- Device Instruction Set related

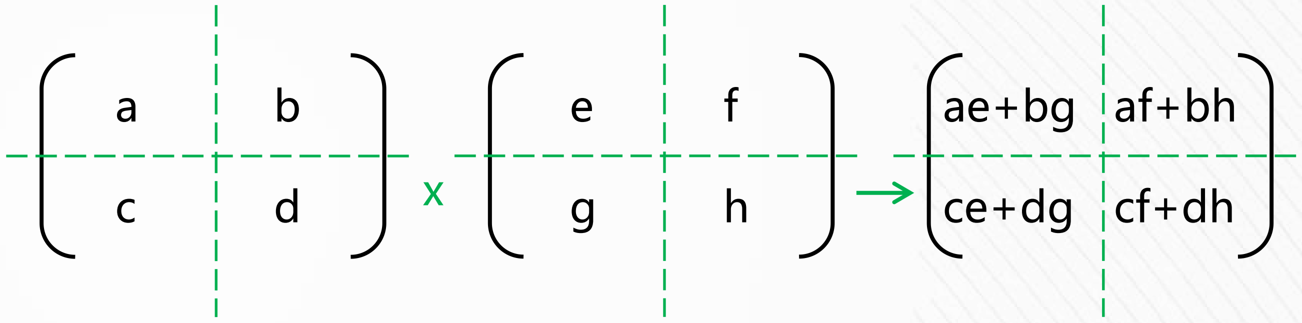
x.asm

Model

Adlik Practice: OP design (Dense)

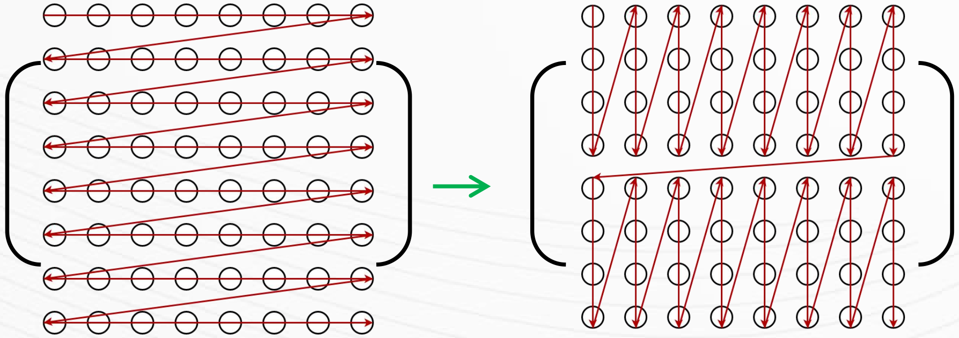


Parallelization

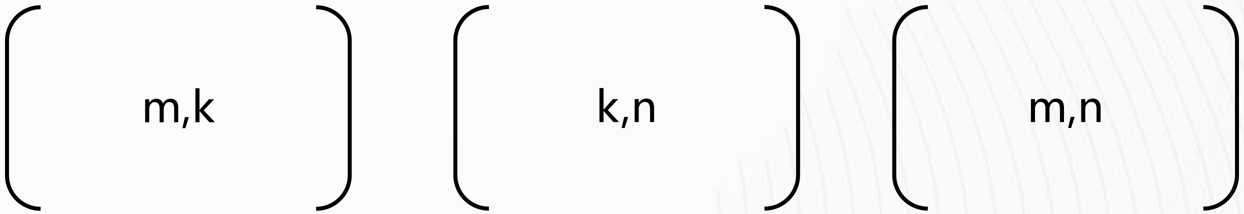


Blocking

Layout Reorder



SIMD（Computing block）

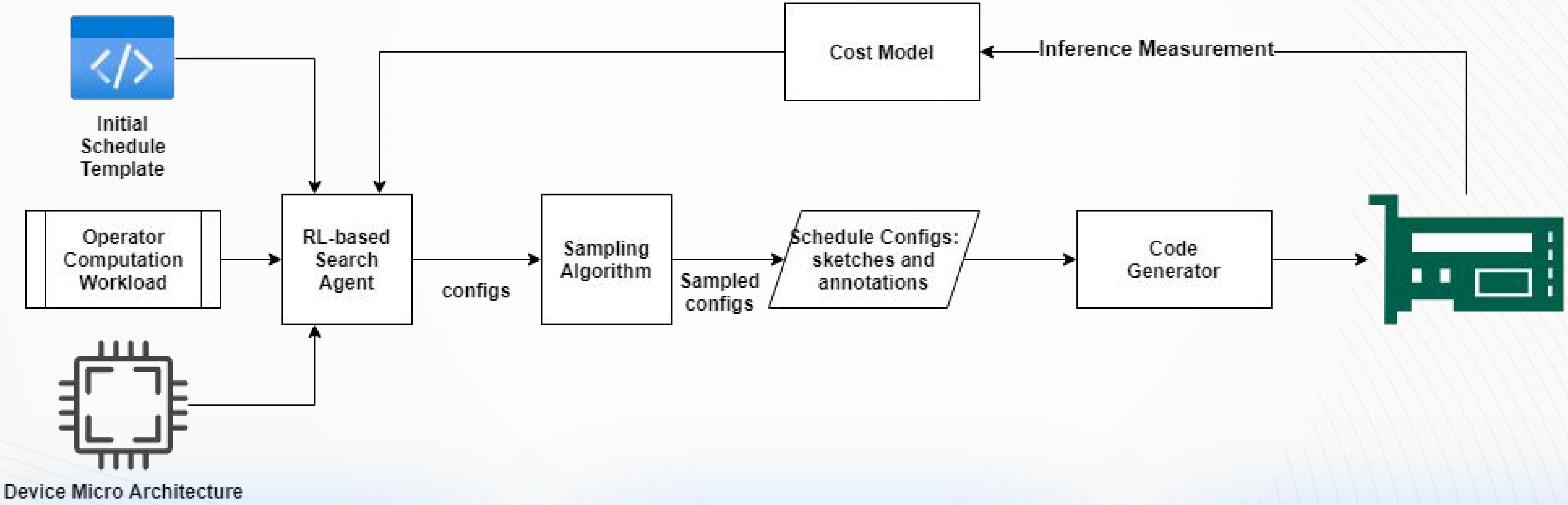


m：4X/8Y/16Z的倍数；n：由X/Y/Zmm个数和实际算法设计决定。

Benchmark test by google/benchmark

Thread Number	1	2	4	8
Improvement（vs oneDNN）	6.5%	5%	7%	5%

Adlik Practice: RL-based Schedule AutoScheduler (Ongoing)


Cost Model
Inference Measurement
Initial Schedule Template
Operator Computation Workload
RL-based Search Agent
configs
Sampling Algorithm
Sampled configs
Schedule Configs: sketches and annotations
Code Generator
Device Micro Architecture

Based on Ansor
(a.k.a TVM auto scheduler)

Agent generates new config based on:
1. Micro-architecture of device
2. Init/last episode configs
3. Operator workload

A cost model is trained to accelerate config evaluation

Adlik Development Status

- ***Released Version 0.1.0 (Antelope): 2020.6***
  - ***Model Optimizer***
  - ***Model Compiler***
  - ***Inference Engine***
  - ***Benchmark Test Framework***

- ***Released Version 0.2.0 (Bear): 2020.11***
  - ***Provide new compiler framework.***
  - ***Support hybrid scheduling of ML and DL inference jobs.***
  - ***Support image based deployment of Adlik compiler and inference engine in cloud native environment.***
  - ***Benchmark test for ResNet-50, Inception V3, Yolo V3 and Bert.***

- ***Released Version 0.3.0 (Cheetah): 2021.6***
  - ***Model compiler with PaddlePaddle/MXNet/Caffe supported***
  - ***Specific optimization for YOLO v4 and Resnet50 v1/v2***
  - ***TVM/OpenVINO/TFLite/TensorRT/TensorFlow runtime integrated***
  - ***Paddle models supported in benchmark test framework***

Adlik Development Status

- ***Community Activity :***
  - Routine TSC meetings.
  - Stable cooperation with CMCC, Unicom, AIIA.
  - Submit CR in ORAN community, introduce Adlik into ORAN framework.
  - Cooperation intention with PaddlePaddle community.

GOTC

THANKS

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

谢谢