7. - Huawei MindSpore AI Development Framework

7.1 - MindSpore Overview

MindSpore logo link

Gilvan Maia

10 Setembro, 2020

Instituto Universidade Virtual

Universidade Federal do Ceará

Disclaimer

The following content is heavily based on HCIA-AI Course material by Huawei Technologies Co., Ltd., authored by Zhang Luxiao and Yuan Meng. Distribution is not allowed.

Index

- Introduction
- MindSpore Architecture
- MindSpore Key Features

Introduction

Introducing MindSpore

MindSpore is a new open source deep learning framework, licensed under the Apache-2.0 License

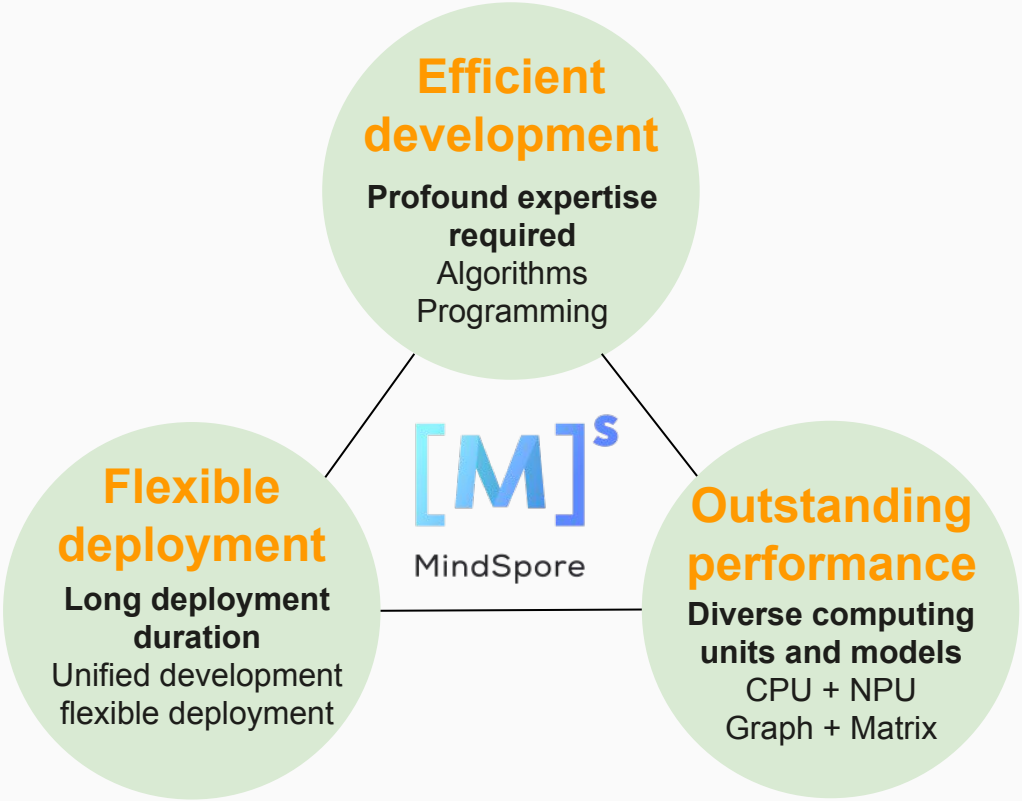
Can be used for mobile, edge, and cloud scenarios

Designed for efficiency of execution, friendly for data scientists and algorithm engineers, native support for Huawei's Ascend AI processors, and software hardware co-optimization.

The current stable release is 0.7.0

A global AI open source community that aims to provide advances on the AI software/hardware application ecosystem

Vision and Value



Supported Platforms

Hardware Platform	Operating System
Ascend910	Ubuntu-x86
	Ubuntu-aarch64
	EulerOS-x86
	EulerOS-aarch64
GPU CUDA 10.1	Ubuntu-x86
CPU	Ubuntu-x86
	Ubuntu-aarch64
	Windows-x86

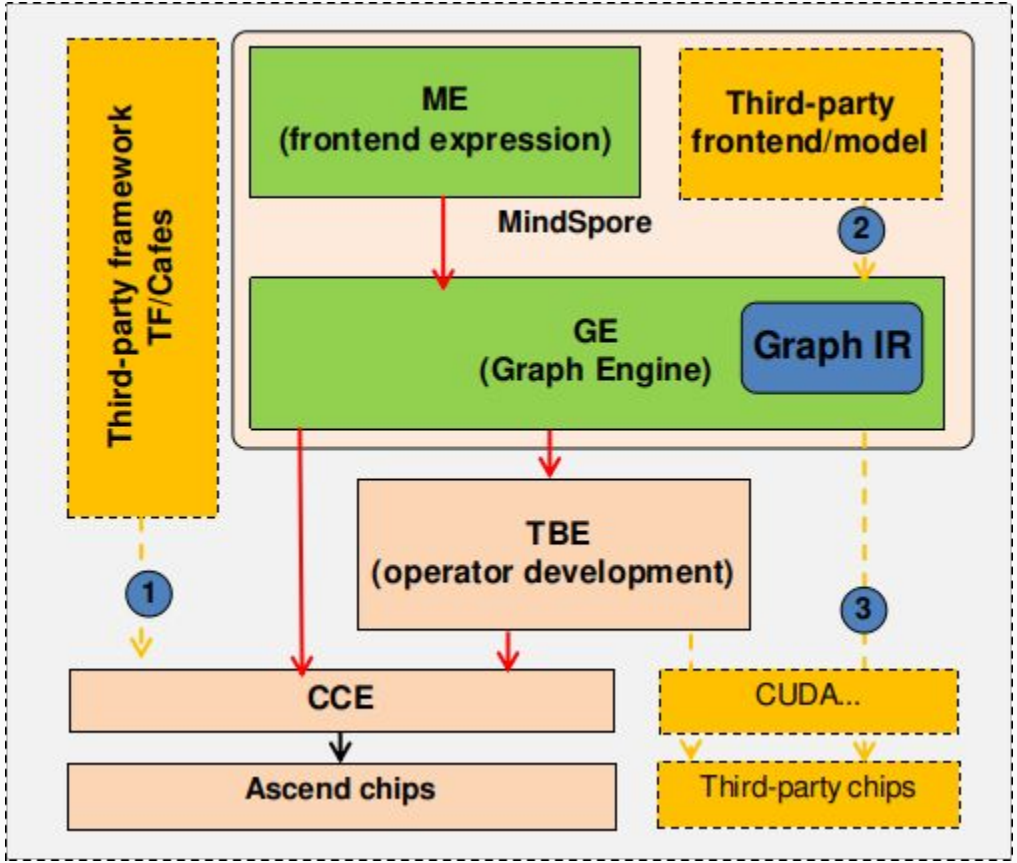
Binary Releases

Hardware Platform	Status
CPU	devel/runtime
GPU CUDA 10.1	devel/runtime
Ascend	coming soon

Docker Images

MindSpore Architecture

Easy Development and Efficient Execution



1. Similarly to open-source frameworks in the industry, MindSpore preferentially serves self-developed chips and cloud services.

2. It supports upward interconnection with third-party frameworks and ecosystems through Graph IR, including training frontends and inference models. Developers can expand the capability of MindSpore.

3. MindSpore also supports interconnection with third-party chips and helps developers increase MindSpore application scenarios and expand the AI ecosystem.

Usability Goals

Automatic differential programming and original mathematical expression

- Auto diff: operator-level automatic differential
- Auto parallel: automatic parallelism
- Auto tensor: automatic generation of operators
- Semi-auto labeling: semi-automatic data labeling

Core Architecture

MindSpore

Unified APIs for all scenarios

Auto differ | Auto parallelism | Auto tuning

MindSpore Intermediate Representation (IR) for computational graph

On-device execution | Pipeline parallelism | Deep graph optimization

Device-edge-cloud co-distributed architecture (deployment, scheduling, communications, etc.)

Easy development:

AI Algorithm As Code

Efficient execution:

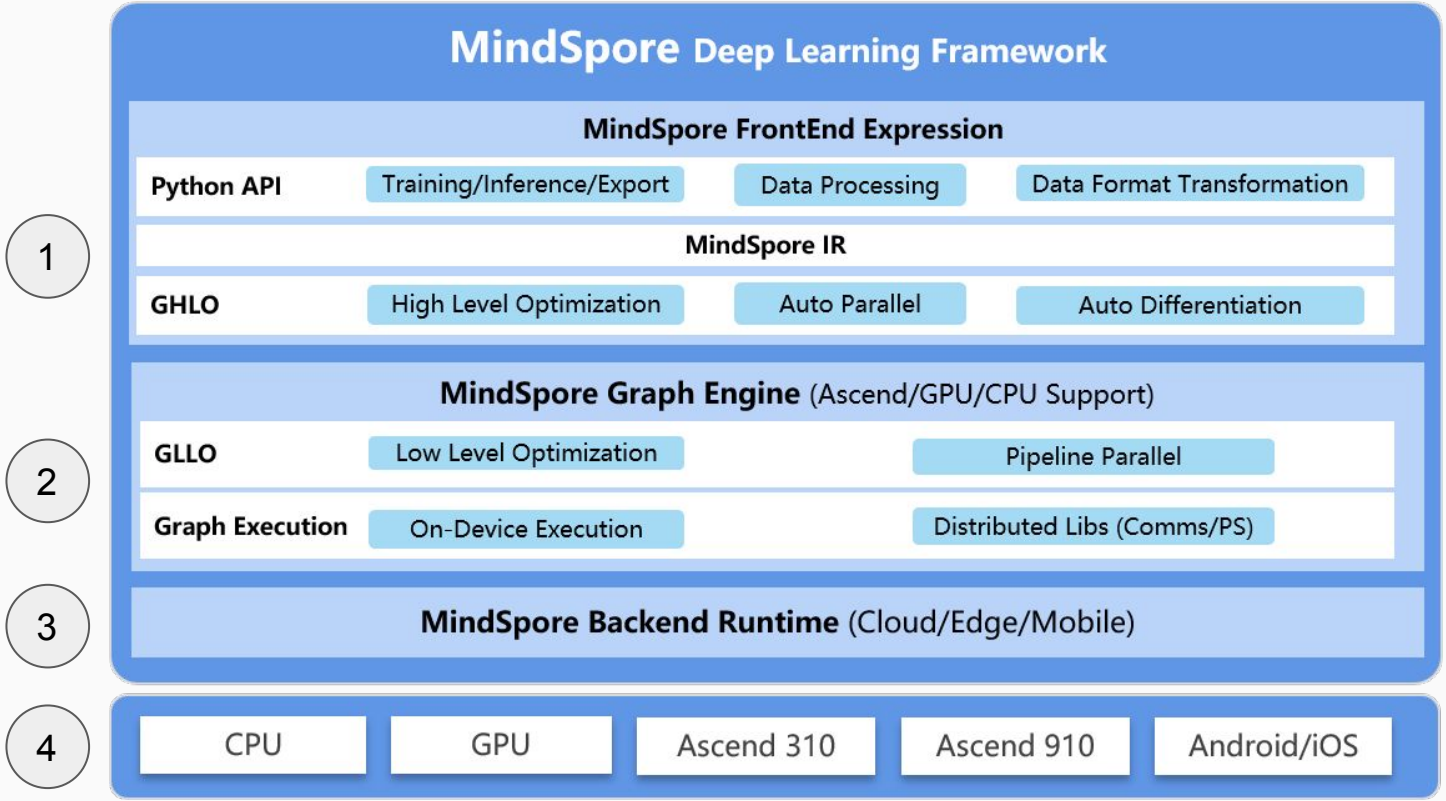
Optimized for Ascend

GPU support

Flexible deployment: on-demand cooperation across all scenarios

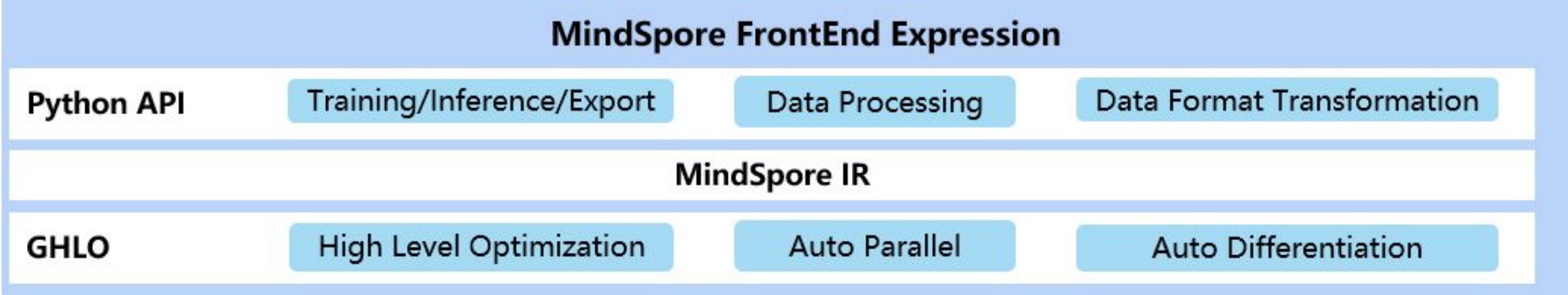
Processors: Ascend, GPU, and CPU

Architecture Overview



The MindSpore framework consists of the following layers: **Frontend Expression**, **Graph Engine**, and **Backend Runtime**

Frontend Expression Layer

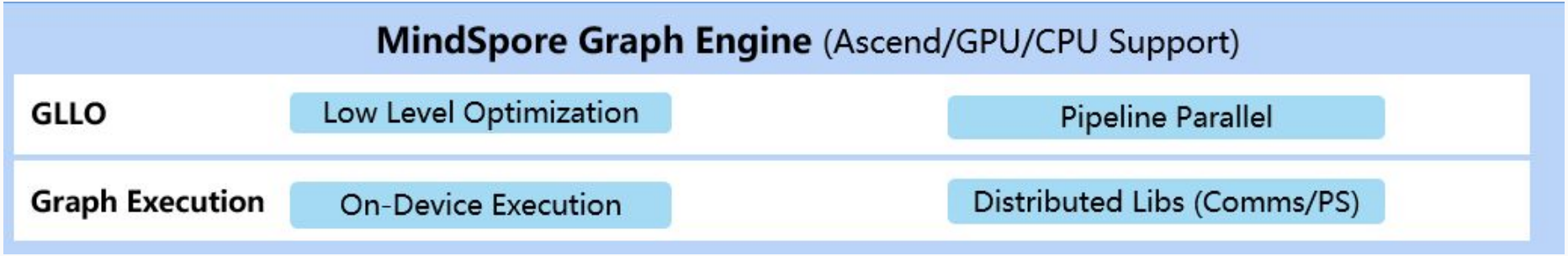


Python APIs provide users with a unified API for model training, inference, and export, and a unified API for data processing and format transformation.

MindSpore IR provides unified intermediate representations, based on which MindSpore performs pass optimization.

Graph High Level Optimization (GHLO) includes optimization irrelevant to hardware (such as dead code elimination), auto parallel, and auto differentiation.

Graph Engine Layer



Graph Low Level Optimization (GLLO) includes hardware-related optimization and in-depth optimization related to the combination of hardware and software, such as operator fusion and buffer fusion

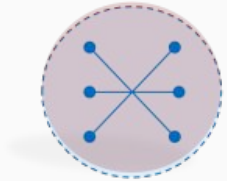
Graph Execution provides communication APIs required for offline graph execution and distributed training

Backend Runtime Layer

MindSpore Backend Runtime (Cloud/Edge/Mobile)

This layer contains the efficient running environments on the cloud, edge, and device.

MindSpore Design: Auto Differ



Technical path of automatic differential

Graph: TensorFlow

- Non-Python programming based on graphs
- Complex representation of control flows and higher-order derivatives

Operator Overloading: PyTorch

- Runtime overhead
- Backward process performance is difficult to optimize.

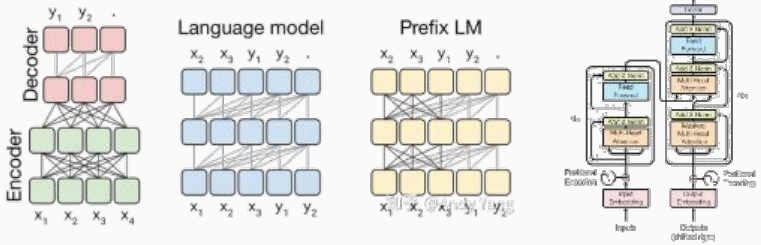
Source Code Transfer: MindSpore

- Python APIs for higher efficiency
- IR-based compilation optimization for better performance

Automatic Parallel

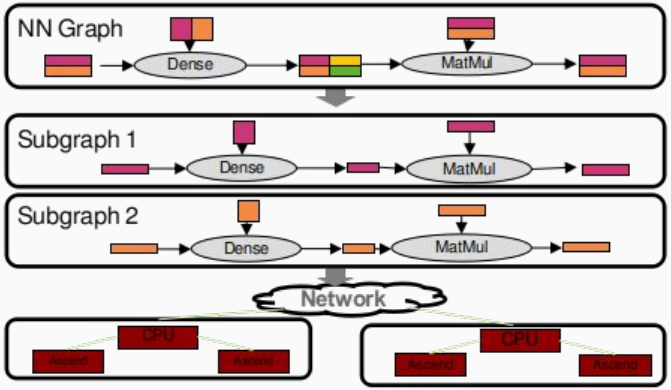
Challenges

Ultra-large models realize efficient distributed training:
As NLP-domain models swell, the memory overhead for training ultra-large models such as Bert (340M)/GPT-2(1542M) has exceeded the capacity of a single card. Therefore, the models need to be split into multiple cards before execution.
Manual model parallelism is **used currently**. Model segmentation needs to be designed and the cluster topology needs to be understood. The development is extremely challenging. The performance is lackluster and can be hardly optimized.



Key Technologies

Automatic graph segmentation: It can segment the entire graph based on the input and output data dimensions of the operator, and integrate the data and model parallelism. Cluster topology awareness scheduling: It can perceive the cluster topology, schedule subgraphs automatically, and minimize the communication overhead.



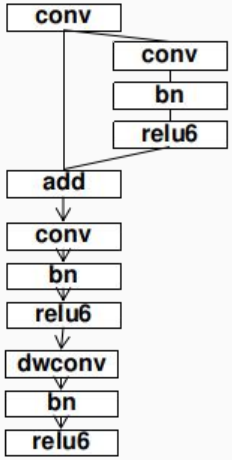
Effect: Realize model parallelism based on the existing single-node code logic, improving the development efficiency tenfold compared with manual parallelism.

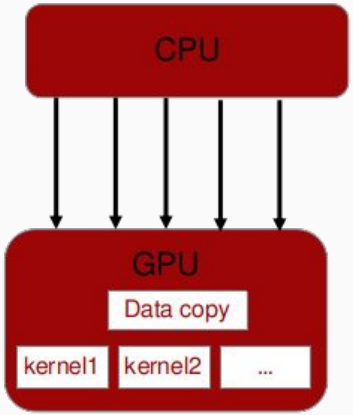
On-Device Execution (1/2)

Challenges

Challenges for model execution with supreme chip computing power:

Memory wall, high interaction overhead, and data supply difficulty. Partial operations are performed on the host, while the others are performed on the device. The interaction overhead is much greater than the execution overhead, resulting in the low accelerator usage.

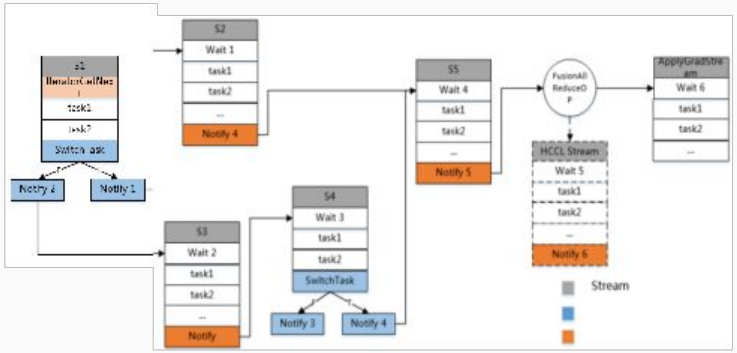




Large data interaction overhead and difficult data supply

Key Technologies

Chip-oriented deep graph optimization reduces the synchronization waiting time and maximizes the parallelism of data, computing, and communication. Data pre-processing and computation are integrated into the Ascend chip:

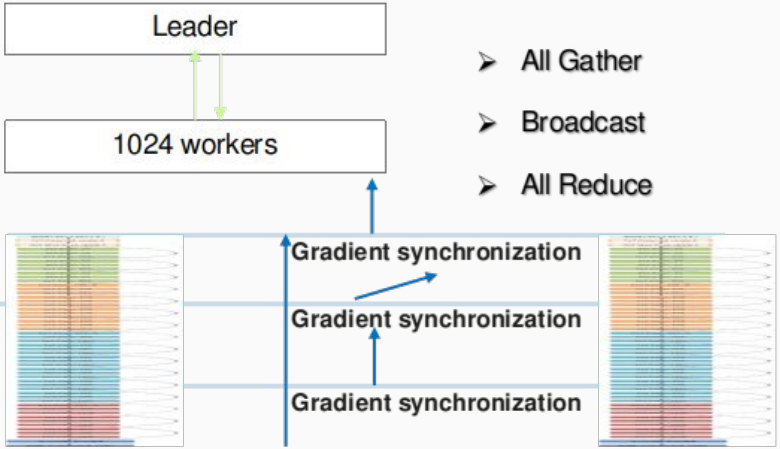


Effect: Elevate the training performance tenfold compared with the on-host graph scheduling.

On-Device Execution (2/2)

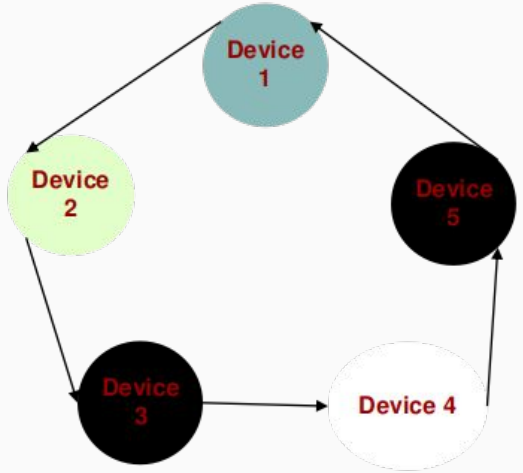
Challenges

Challenges for distributed gradient aggregation with supreme chip computing power:
the synchronization overhead of central control and the communication overhead of frequent synchronization of ResNet50 under the single iteration of 20 ms; the traditional method can only complete All Reduce after three times of synchronization, while the data-driven method can autonomously perform All Reduce without causing control overhead.



Key Technologies

The optimization of the **adaptive graph segmentation driven by gradient data** can realize decentralized All Reduce and synchronize gradient aggregation, boosting computing and communication efficiency.



Effect: a smearing overhead of less than 2 ms

Distributed Device-Edge-Cloud Synergy

Challenges

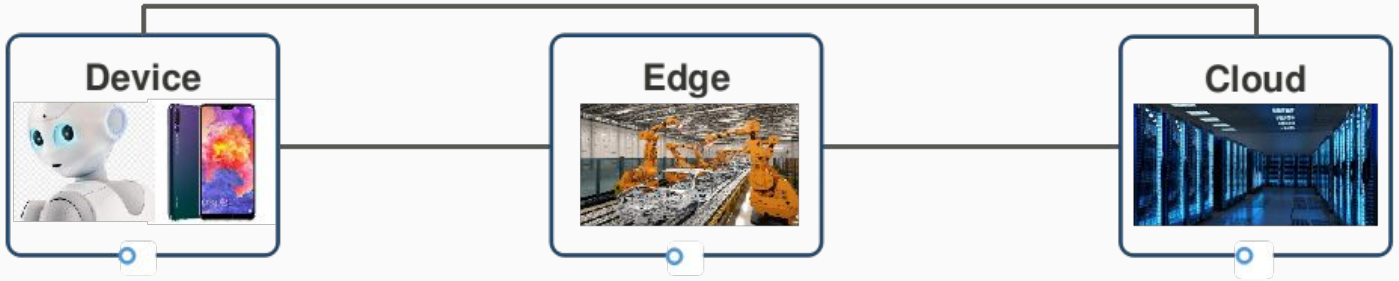
The diversity of hardware architectures leads to full-scenario deployment differences and performance uncertainties. The separation of training and inference leads to isolation of models.

Key Technologies

- **Unified model IR** delivers a consistent deployment experience.
- **The graph optimization technology featuring software and hardware collaboration** bridges different scenarios.
- Device-cloud Synergy Federal Meta Learning breaks the device-cloud boundary and updates the multi-device collaboration model in real time.

Effect: consistent model deployment performance across all scenarios thanks to the unified architecture, and improved precision of personalized models

On-demand collaboration in all scenarios and consistent development experience



MindSpore Key Features

AI Computing Framework: Challenges

Industry challenges include a huge gap between industry research and all-scenario AI applications

- High entry barriers
- High execution cost
- Long deployment duration

MindSpore provides innovative technologies to facilitate high performance, inclusive AI across applications

- New programming mode
- New execution mode
- New collaboration mode

MindSpore: New Programming Paradigm

Algorithm scientist + experienced system developer write specific code for debugging and parallelism settings

Less code can be written by algorithm scientist in MindSpore

- One-line Automatic Parallelism
- Efficient Automatic Differentiation
- **One-line debug-mode switch**

For example, a Transformer NLP Model demands ~20% less code in MindSpore

New Execution Mode (1/2)

Execution Challenges

Complex AI computing and diverse computing units

1. CPU cores, cubes, and vectors
2. Scalar, vector, and tensor computing
3. Mixed precision computing
4. Dense matrix and sparse matrix computing

Multi-device execution: High cost of parallel control

Performance cannot linearly increase as the node quantity increases.

On-device execution

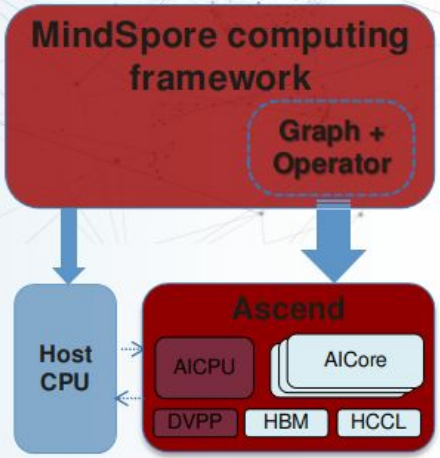
Offloads graphs to devices, maximizing the computing power of Ascend

Framework optimization

- Pipeline parallelism
- Cross-layer memory overcommitment

Soft/hardware co-optimization

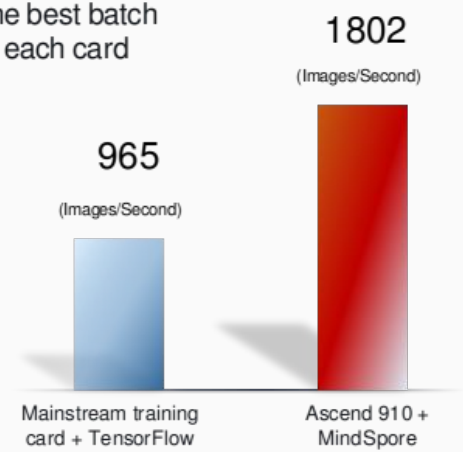
- On-device execution
- Deep graph optimization



HUAWEI

New Execution Mode (2/2)

- ResNet 50 V1.5
- ImageNet 2012
- With the best batch size of each card



Performance of ResNet-50 is doubled.
Single iteration:
58 ms (other frameworks+V100) v.s. about **22 ms** (MindSpore) (ResNet50+ImageNet, single-server, eight-device, batch size=32)

Detecting objects in 60 ms

Tracking objects in 5 ms

Multi-object real-time recognition
MindSpore-based mobile deployment, a smooth experience of multi-object detection

HUAWEI

New Collaboration Mode

Deployment Challenge

V.
S.

- Varied requirements, objectives, and constraints for device, edge, and cloud application scenarios

V.
S.

- Different hardware precision and speed

Unified development; flexible deployment; on-demand collaboration, and high security and reliability

Development

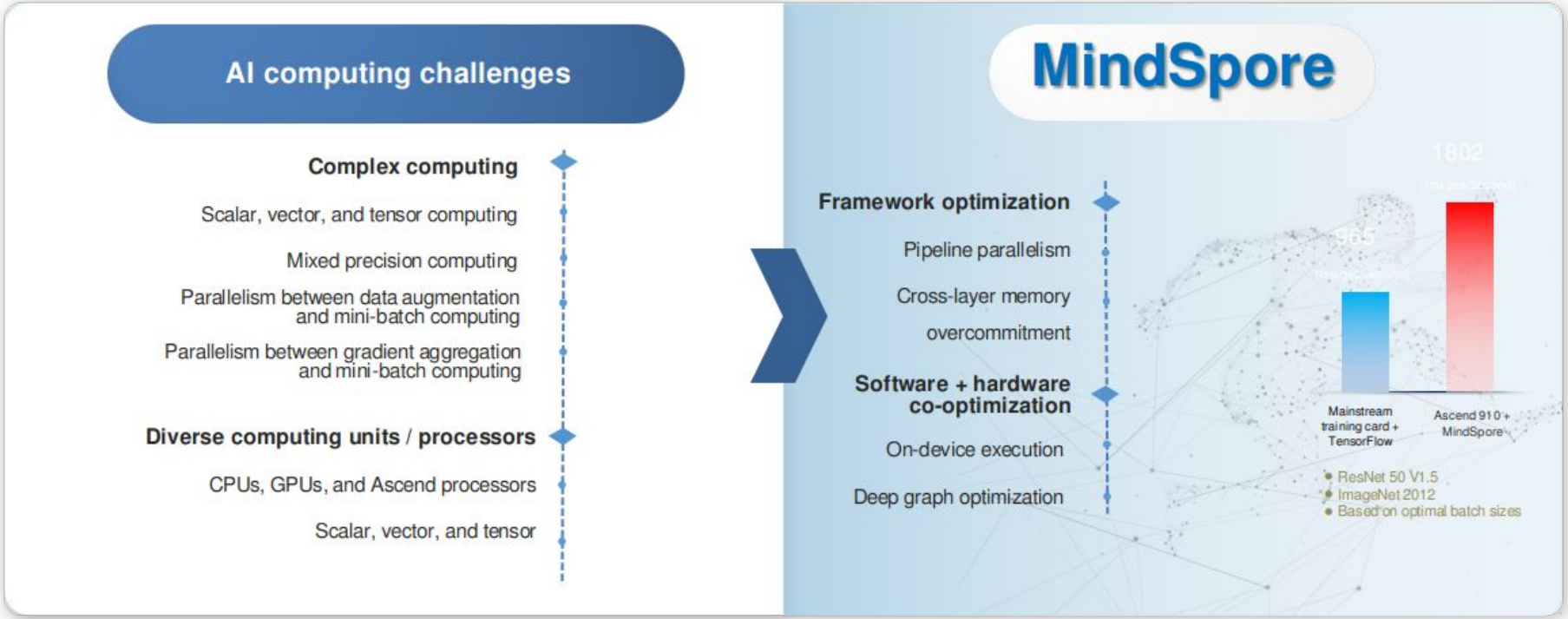
Deployment

Execution

Saving model

Unified development and flexible deployment

High Performance



Thank You!

Next: 7.2 MindSpore Application Development