Huawei AI Development Framework — MindSpore

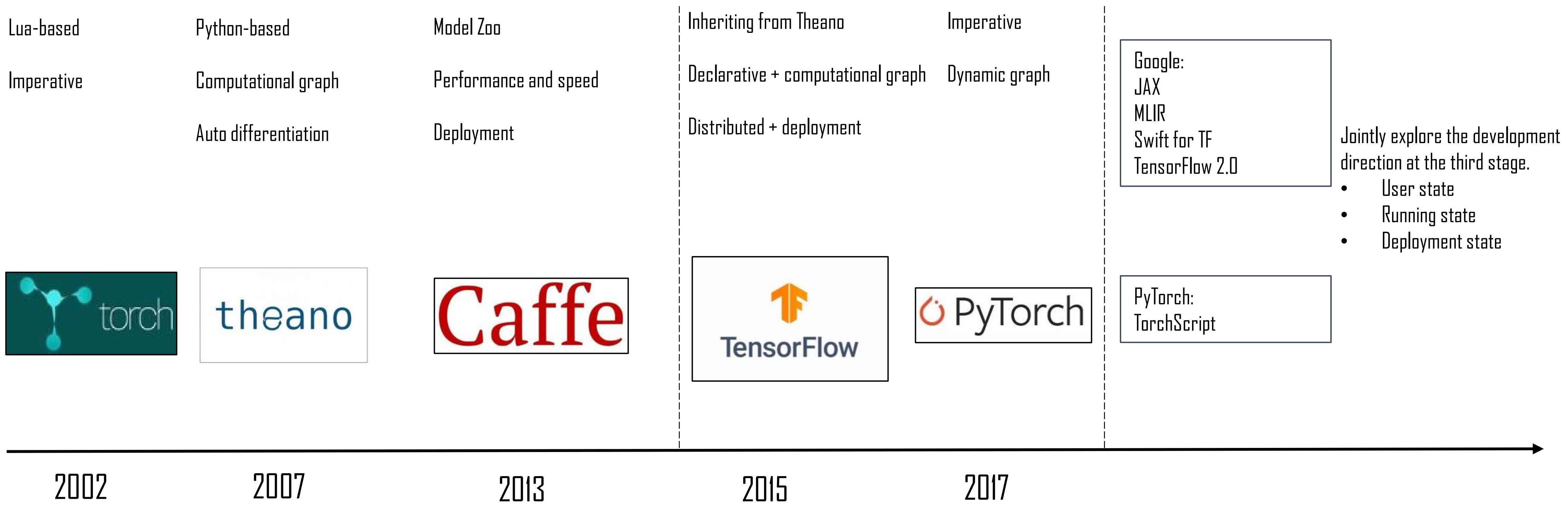
Objectives

On completion of this course, you will be able to:

- **Describe MindSpore.**
- **Understand the MindSpore framework.**
- **Understand MindSpore design ideas.**
- **Understand MindSpore features.**
- **Understand MindSpore environment setup process and development cases.**

Contents

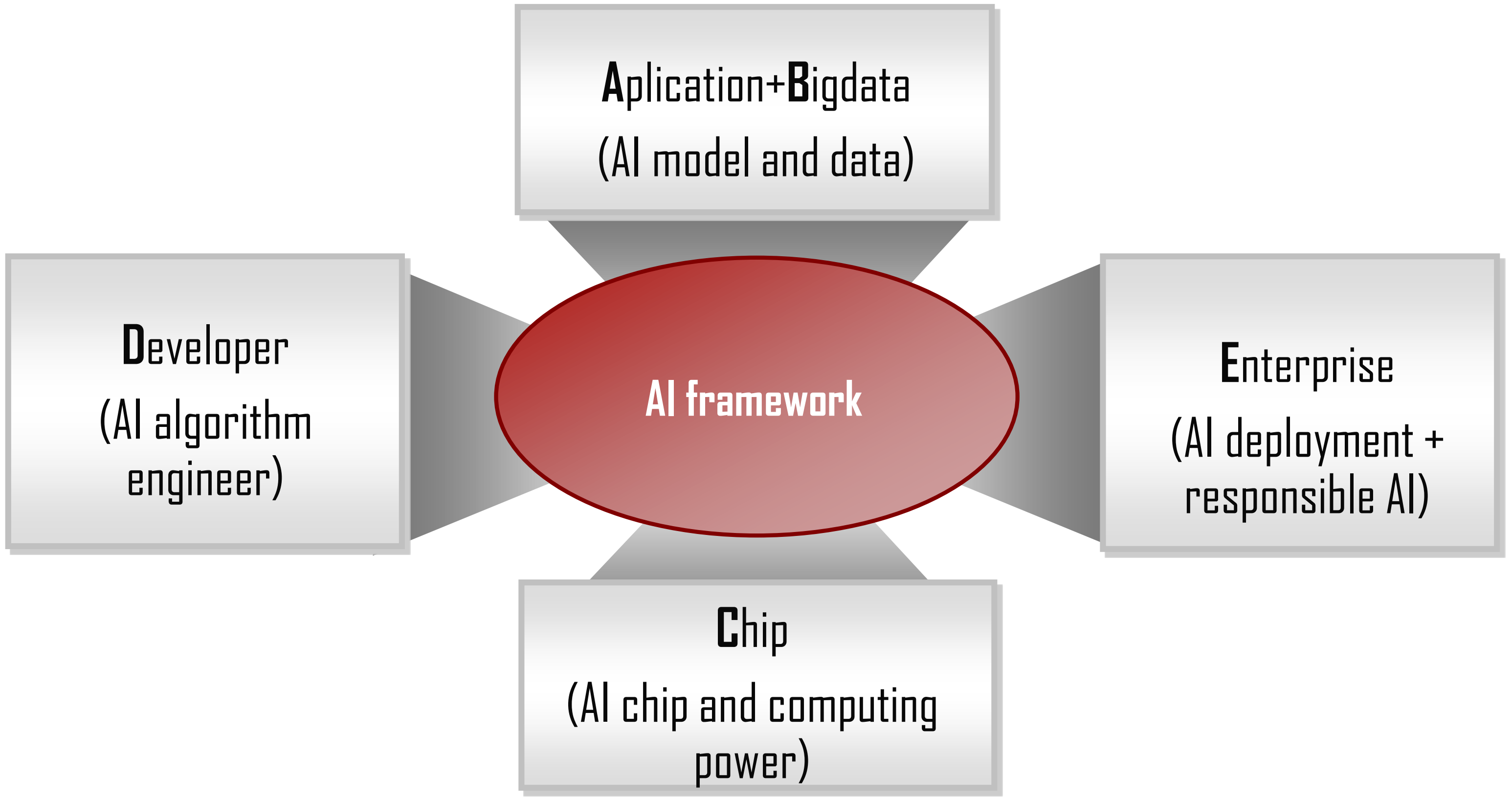
AI Framework Development History



The AI framework technology has not been converged. Google has invested in four different directions to explore technologies. It is estimated that the integration workload in the future is huge.

"ABCDE": Five Factors Driving the Evolution of the AI Framework

- Increasing model scale and complexity (GPT-3 parameter quantity reaches 175 billion.)
- Evolution from a single NN to general-purpose AI and scientific computing



- New AI programming language
- Unification of dynamic and static graphs

- Model deployment
- Security and privacy protection

- Continuous improvement of chip/cluster performance (Atlas 900 cluster supports a maximum of exabyte-level computing power.)
- Diversified heterogeneous computing power for CPUs, GPUs, and NPUs

Contents

Challenge 1: Increasing Model Scale and Complexity

Date	Model	Parameters	Institution
2018.4	ELMO	94m	Ai2
2018.7	GPT	110m	OpenAI
2018.10	BERT-Large	340m	Google
2019.1	Transformer ELMO	465m	Ai2
2019.1	GPT-2	1.5b	OpenAI
2019.7	MegatronLM	8.3b	NVDIA
2020.2	T-NLG	17.5b	Microsoft
2020.5	**GPT-3**	**175b**	**OpenAI**

- **GPT-3:**

1. Parameters: 175 billion (600 GB+)
2. Datasets (before processing): 45 TB
3. Training cost: tens of millions of dollars; 1024 V100 GPUs; 127 days

- **Technical challenges and trends:**

1. Performance (memory, communication, and computing usage)
   - Challenges: The single-device memory is insufficient (32 GB). The traffic volume varies greatly due to different parallel partitioning. The computing usage of different parallel partitioning is different. The data preprocessing bottleneck occurs.
   - Trend: memory overcommitment, hybrid parallelism (data parallelism, model parallelism, and pipeline parallelism), and data acceleration.
2. Efficiency
   - Challenges: Manual partitioning is demanding. Parallel logic and algorithm logic are coupled.
   - Trend: automatic parallelism.
3. Accuracy
   - Challenge: Optimizer for large batch sizes
   - Trend: second-order optimization

Challenge 2: Evolution from Single NN to General-Purpose Tensor Differentiable Computing

Deep probabilistic learning:
Combine NN and probability models.

Graph neural networks:
Combine NN and graph structure data.

Challenges:

- Integrate NN models and probability models for modeling, reducing the learning difficulty.
- Store, partition, and sample trillions of distributed graph data.
- Support dynamic network structure and elastically distributed training.

AI modeling
Build AI-based computable models.

AI solution
Design new solutions with the help of neural networks.

Framework resolution
Accelerate equation solving with the help of new frameworks.

Challenges:

- Equations as code. Users can quickly construct expressions, and the serial coding and parallel coding are consistent.
- Support large-scale heterogeneous parallelism and mixed precision computing.
- Support high-performance higher-order differentiation (the volume of computing higher-order differentiation increases exponentially with the order).

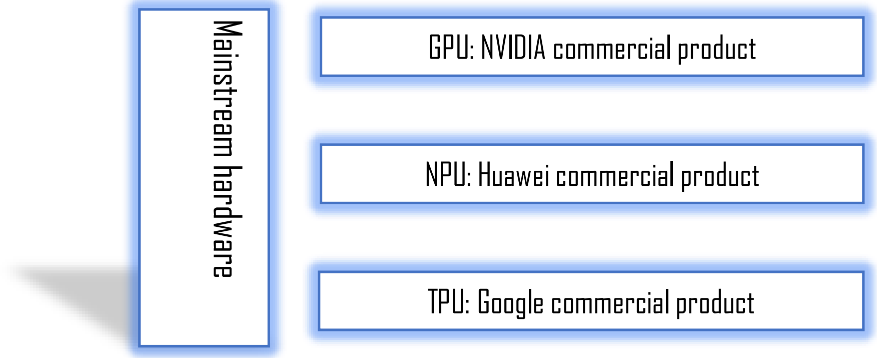
- **Computing graphs (Taichi)**

Differentiable physical engines

Challenges:

- Sparse expression
- Separation of data and computing
- Differentiable programming

Challenge 3: Continuously Increasing Computing Power and Complexity


Mainstream hardware
GPU: NVIDIA commercial product
NPU: Huawei commercial product
TPU: Google commercial product

Hardware development trends

- ➢ Increase the computing density of a single core, improve the bandwidth and the process, increase the number of cores, and package more silicon chips.
- ➢ Widely use SIMD and increase the tensor core processing scale (4 x 4 → 16 x 16).
- ➢ New data types (such as TF32 and BF16), high-speed interconnection between chips, and support for virtualization.

Key challenges to AI framework software during AI chip development:

➢ **Improve the coupling of optimization and hardware, and integrate graph build and operator build.**

- Fusion optimization at graph layer: Converge hardware-independent optimization, fully utilize the hardware computing power, and break the boundary between the subgraph level and operator level for overall optimization.
- Optimization at operator layer: Consider hardware capabilities when using operators to implement algorithms.

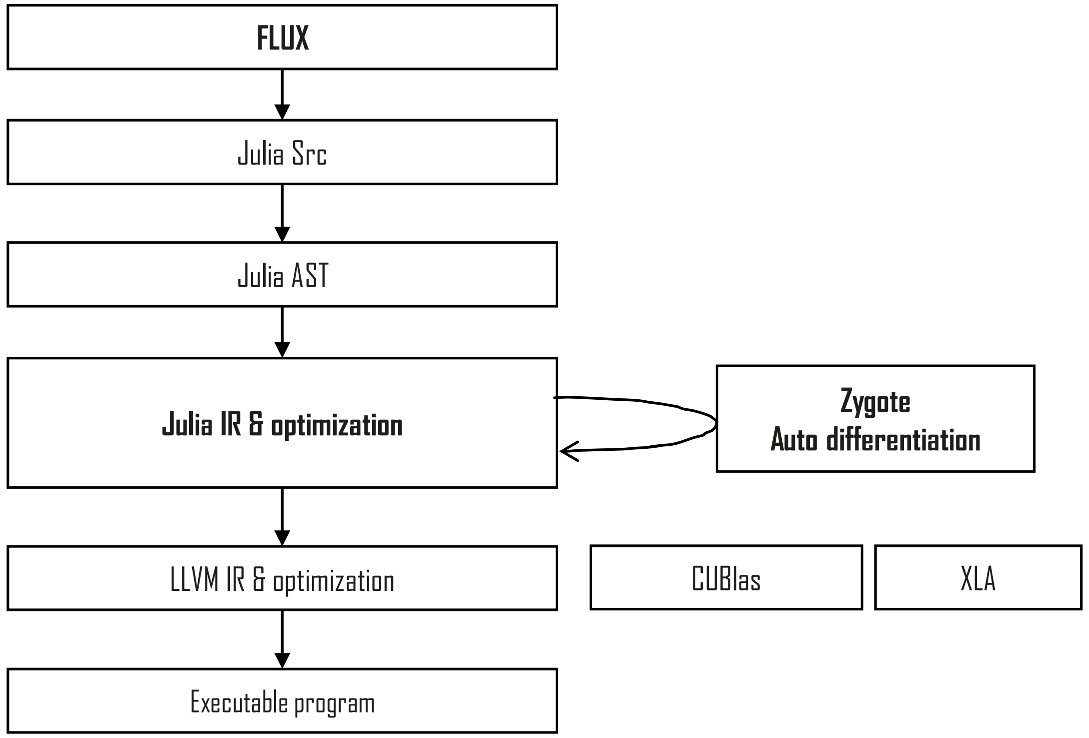
➢ **Apply model execution modes to scenarios and hardware.**

- Mix the graph sink mode and single-operator execution. Use different optimal mode according to the hardware.
- Use the data flow execution mode to better exert the computing power.
- Use the SoC-level distributed parallel strategy for packaging more silicon chips.
- Use virtualization execution mode in SoC.

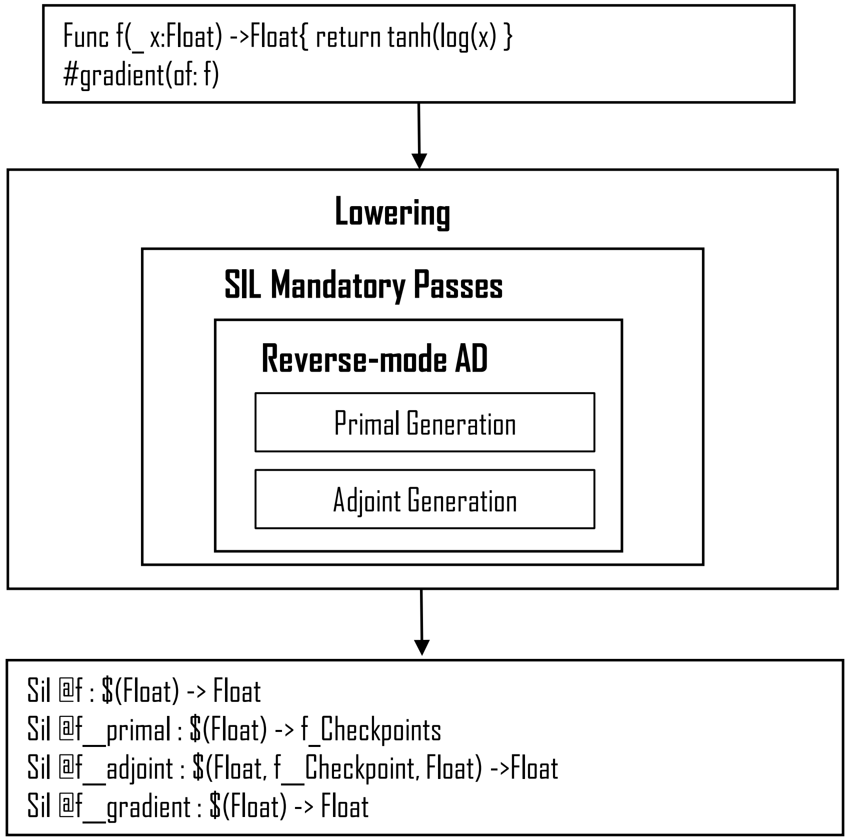
➢ **Huge programmability challenges.**

- The effective computing power is close to the theoretical computing power and has high requirements on the compiler;
- Sparse acceleration, image preprocessing acceleration module, and complex SIMD acceleration instructions;
- SoC-level heterogeneous programming: CUBE core, Vector core, and ARM.
- Multi-chip, single-chip cross-generation, and cross-model compatibility requirements.

Challenge 4: New Programming Languages Making Breakthroughs in Python



- Julia enters the AI field based on the tensor native expression, IR openness, and high performance as well as the accumulation in the scientific computing and HPC fields.

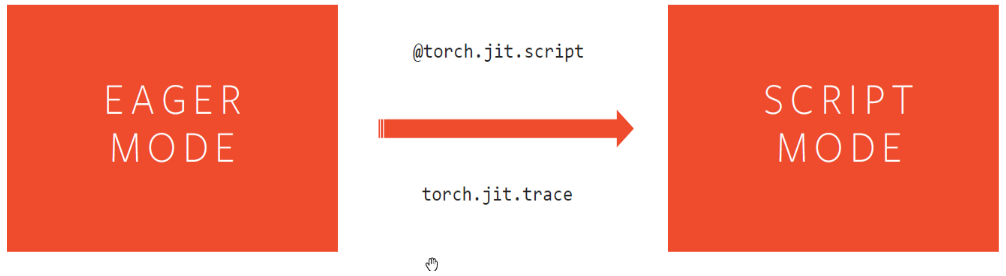


```
Func f(_ x:Float) ->Float{ return tanh(log(x) }
#gradient(of: f)
```

```
Sil @f : $(Float) -> Float
Sil @f__primal : $(Float) -> f_Checkpoints
Sil @f__adjoint : $(Float, f__Checkpoint, Float) ->Float
Sil @f__gradient : $(Float) -> Float
```

- Swift for TensorFlow tries to find differentiated competitiveness based on enterprise-class features such as static type, easy deployment, and high performance.

Challenge 5: Unification of Dynamic and Static Graphs



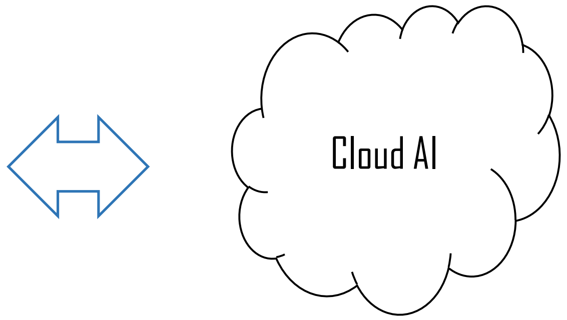
- Research phase: dynamic graph; Python affinity, flexibility, and usability.
- Production phase: static graph; performance, and deployment capability.

- Pain point: The representation of the dynamic graph is not completely the same as that of the static graph.
- Trend: Optimize JIT to achieve the consistency of the two representations.
- Challenge: It is difficult to fully use JIT to support Python flexibility and dynamics.

- Industry frameworks use compilers such as accelerated linear algebra (XLA) to work with chips for in-depth optimization.
- Gradually improve the IR from the perspective of optimization to form open AI infrastructure, such as Relay/TVM and MLIR.

Challenge 6: AI Deployment in All Scenarios

According to the 2019 CIO Agenda survey conducted by Gartner, **the proportion of enterprises that have deployed AI increased from 4% to 14%** from 2018 to 2019. The data is in sharp contrast to the industry's increasing awareness of the value of AI.



Privacy and security

Low latency

High reliability

Low bandwidth required

High computing power

Large model

Big data

High network bandwidth

Tapping mode identification:
Deep learning (90%) vs. Traditional method (60%)

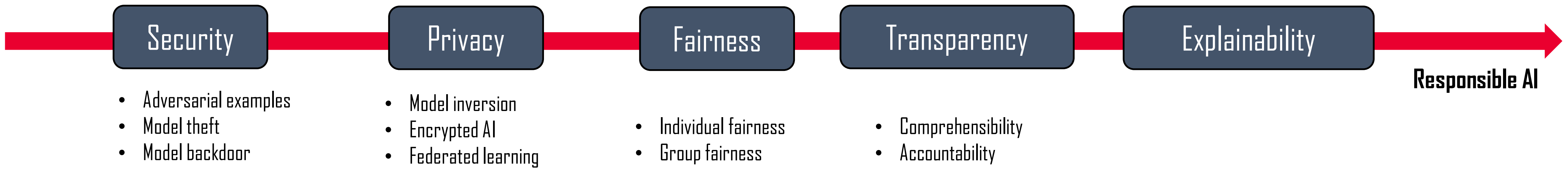
Trend 1: To reduce latency and improve user experience, on-device language model deployment becomes a trend. The challenge is how to reduce the model size and minimize the precision loss.

Trend 2: device-cloud synergy

1. Mobile AI = On-Device AI + Smart services, better considering personalization, security, and privacy.
2. Single agent → multiple agent collaboration, implementing real-time perception and decision-making.

Trend 3: Ubiquitous AI is deployed in scenarios where IoT and smart devices have extremely limited resources.

Challenge 7: Security, Privacy, and Protection



Trend insights:

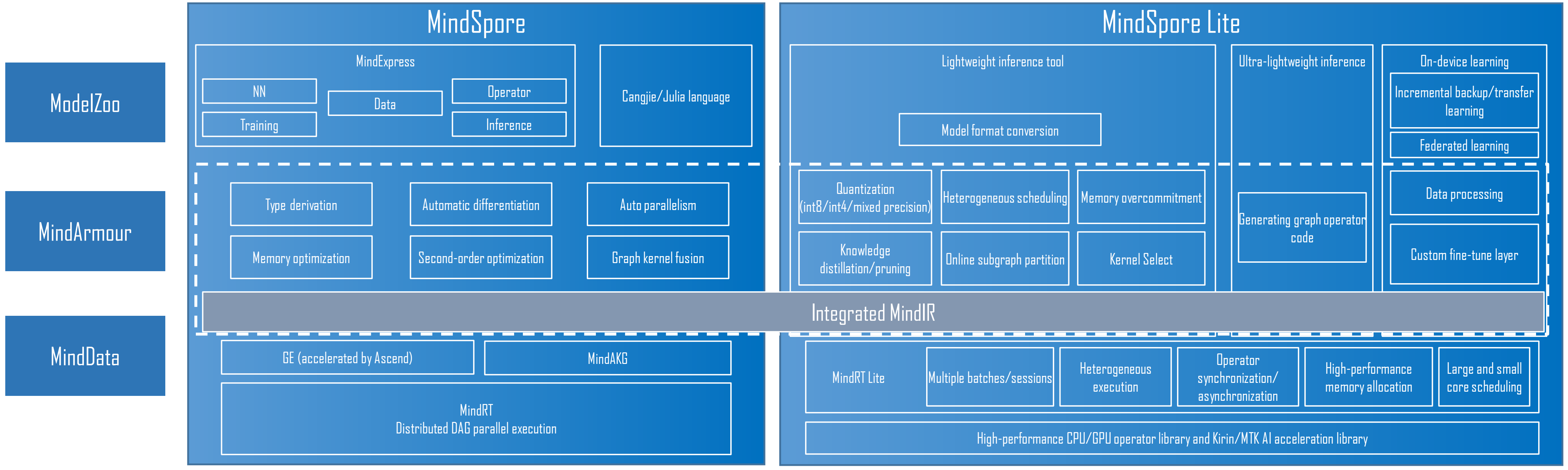
- ➢ In the future, in addition to accuracy and performance, meeting responsible AI will be a key requirement for AI service success.
- ➢ The AI framework bears AI services and must have the capability of enabling responsible AI.

Key challenges:

- ➢ There is no general analysis method and measurement system for all aspects of responsible AI, and there is no automatic measurement method for scenario awareness.
- ➢ AI model robustness, privacy protection technologies, and encrypted AI have great impact on model performance in actual scenarios.
- ➢ Responsible AI is deeply combined with AI explainability and verifiability.

Contents

MindSpore Open-source Deep Learning Framework



ModelZoo
MindArmour
MindData
MindSpore
MindExpress
NN
Training
Data
Operator
Inference
Cangjie/Julia language
Type derivation
Automatic differentiation
Auto parallelism
Memory optimization
Second-order optimization
Graph kernel fusion
GE (accelerated by Ascend)
MindAKG
MindRT
Distributed DAG parallel execution
MindSpore Lite
Lightweight inference tool
Model format conversion
Ultra-lightweight inference
On-device learning
Incremental backup/transfer learning
Federated learning
Quantization (int8/int4/mixed precision)
Heterogeneous scheduling
Memory overcommitment
Knowledge distillation/pruning
Online subgraph partition
Kernel Select
Generating graph operator code
Data processing
Custom fine-tune layer
Integrated MindIR
MindRT Lite
Multiple batches/sessions
Heterogeneous execution
Operator synchronization/asynchronization
High-performance memory allocation
Large and small core scheduling
High-performance CPU/GPU operator library and Kirin/MTK AI acceleration library

Superior performance

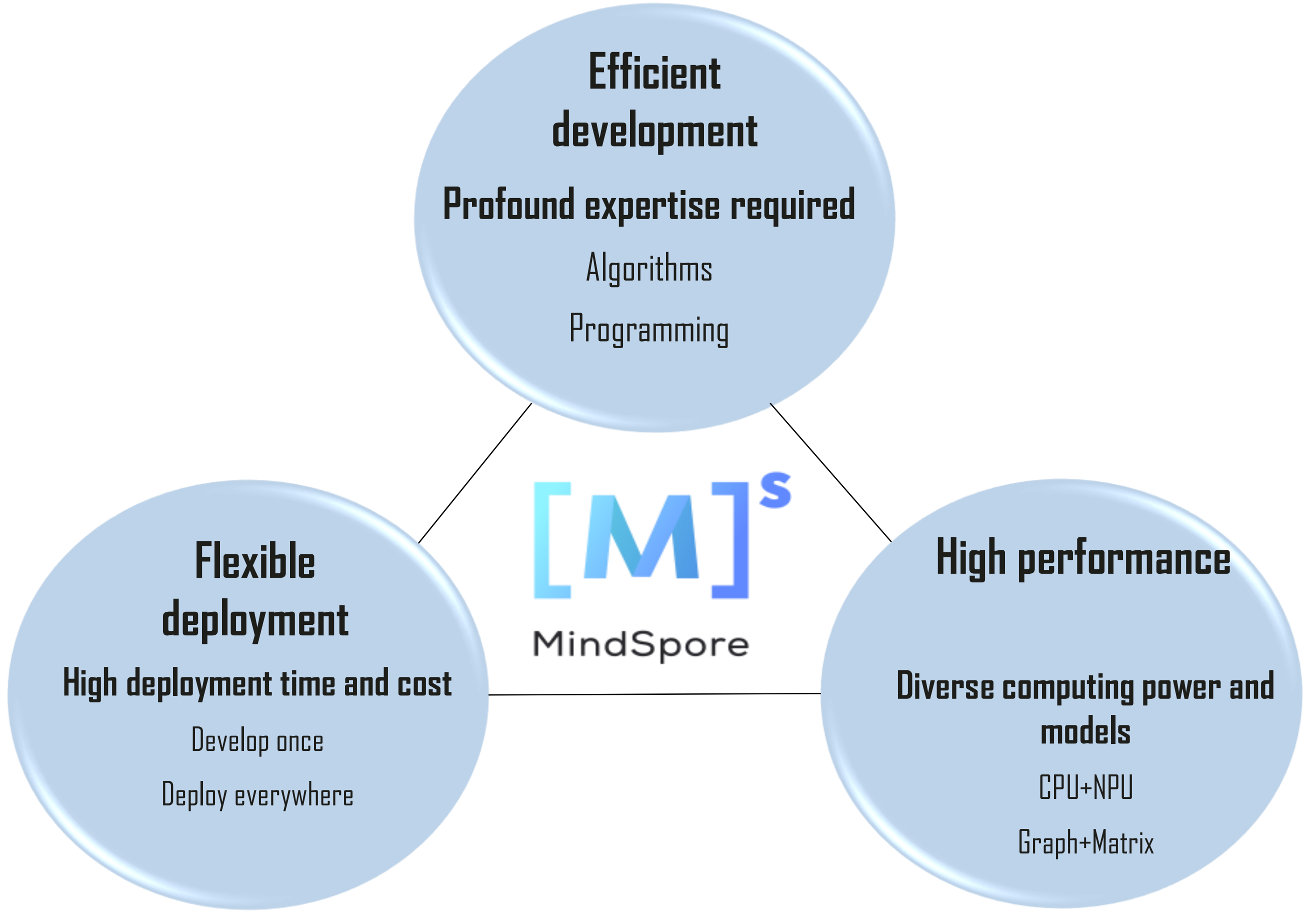
All-scenario support

Lightweight

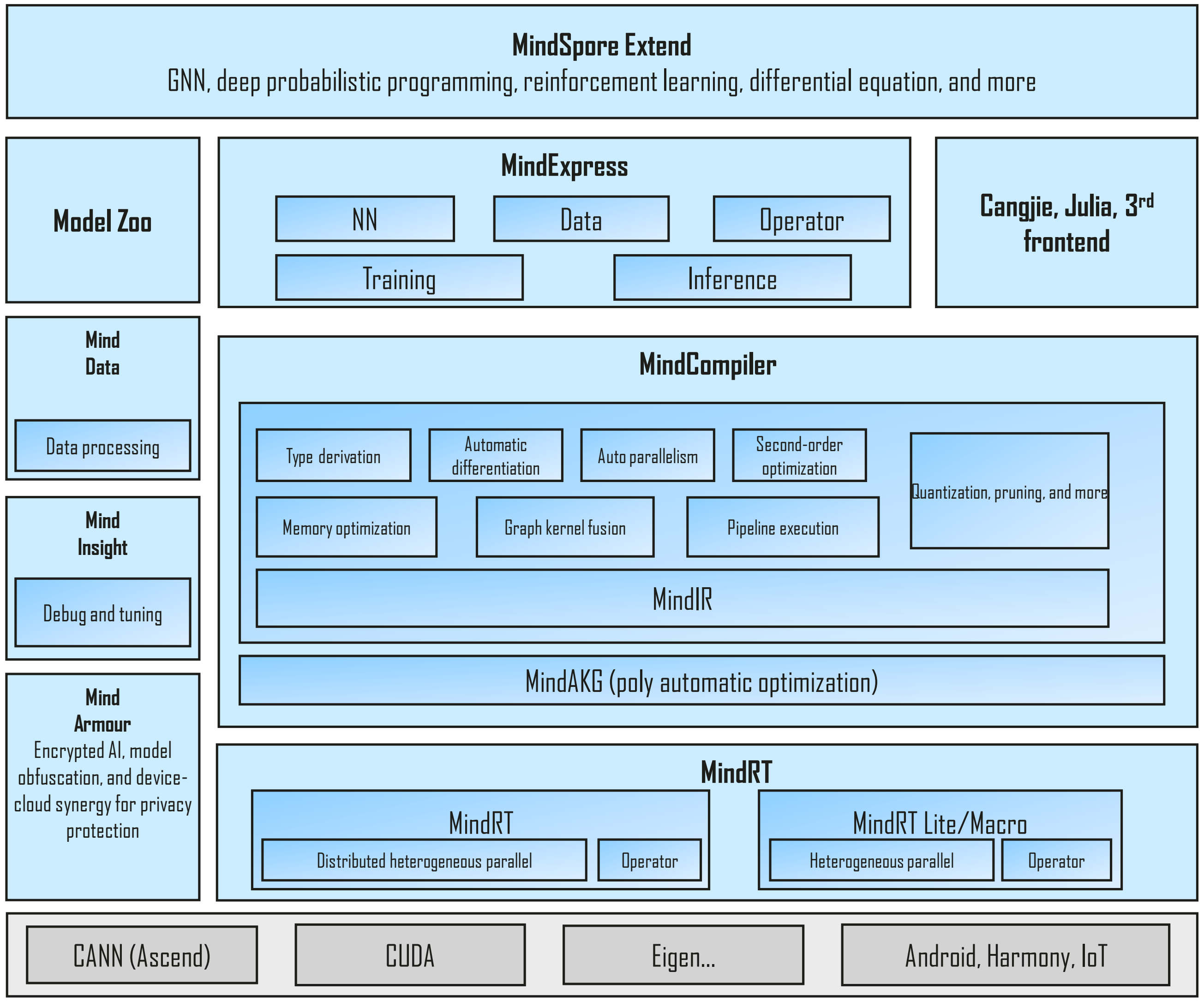
Efficient deployment

MindSpore Vision and Value

- Lower the barrier for AI development, maximize Ascend computing power, and empower inclusive AI.



MindSpore Logical Architecture



MindSpore Extend
GNN, deep probabilistic programming, reinforcement learning, differential equation, and more
Model Zoo
MindExpress
NN
Data
Operator
Training
Inference
Cangjie, Julia, 3rd frontend
Mind Data
Data processing
Mind Insight
Debug and tuning
Mind Armour
Encrypted AI, model obfuscation, and device-cloud synergy for privacy protection
MindCompiler
Type derivation
Automatic differentiation
Auto parallelism
Second-order optimization
Quantization, pruning, and more
Memory optimization
Graph kernel fusion
Pipeline execution
MindIR
MindAKG (poly automatic optimization)
MindRT
MindRT
Distributed heterogeneous parallel
Operator
MindRT Lite/Macro
Heterogeneous parallel
Operator
CANN (Ascend)
CUDA
Eigen...
Android, Harmony, IoT

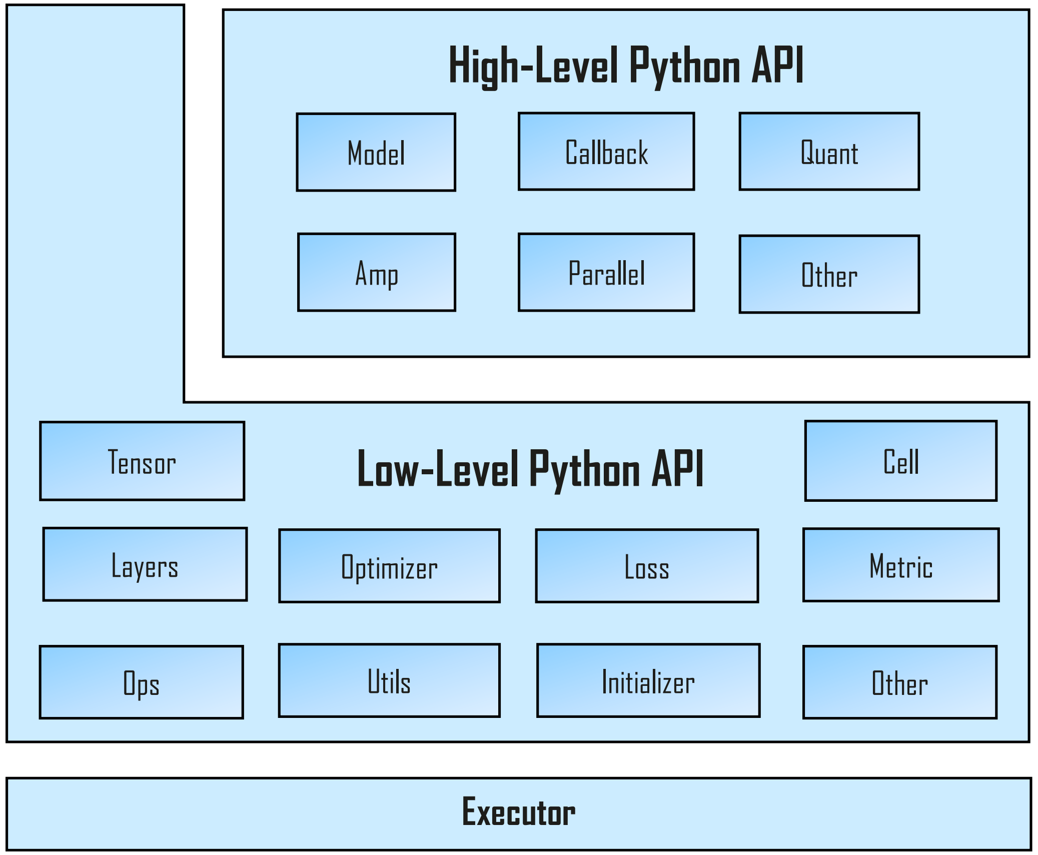
Design Objectives

- **Beyond AI**: NN applications general AI + numerical computation
  - Key feature: general-purpose tensor derivable computing
- **Distributed parallel native**: supporting AI models to go beyond trillions of parameters
  - Key features: automatic parallelism, memory-constrained programming, and second-order optimization
- **In-depth graph kernel fusion**: capitalizing on the computing power of AI chips
  - Key features: joint graph and kernel optimization as well as automatic optimization based on Poly
- **Enterprise-level capabilities in all scenarios**: flexible deployment and collaboration, secure, reliable, and explainable
  - Key features: ultra-lightweight runtime, private training, adaptive model generation, quantitative training, and explainable AI

Design philosophy: AI "JDK"

- **Representation/optimization/operation decoupling:** multi-frontend, cross-chip, and cross-platform
- **Openness:** opening the general graph compilation and running capabilities to third-party frameworks
- **Centralized architecture for all scenarios:** integrated APIs and IRs, enabling smooth AI applications

Subsystem: MindExpress



Design objectives:

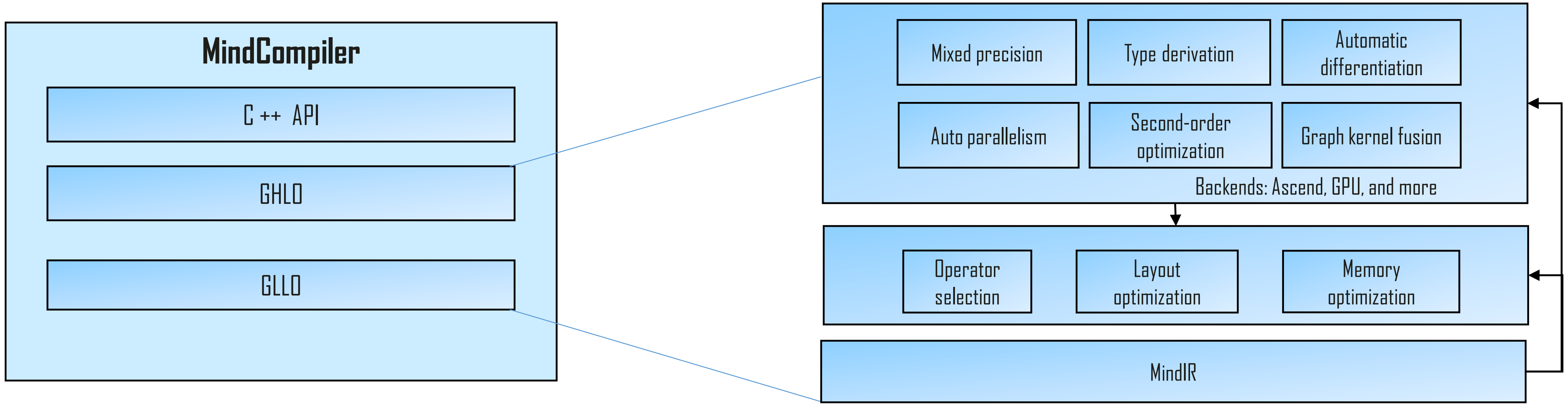
- Design both high-level and low-level APIs for users, supporting network building, entire graph execution, subgraph execution, and single-operator execution.
- Provide integrated APIs for model training, inference, and export, suitable for various scenarios, such as the device, edge, and cloud.
- Provide unified encoding for dynamic and static graphs.
- Provide unified encoding for standalone and distributed training.

Functional modules:

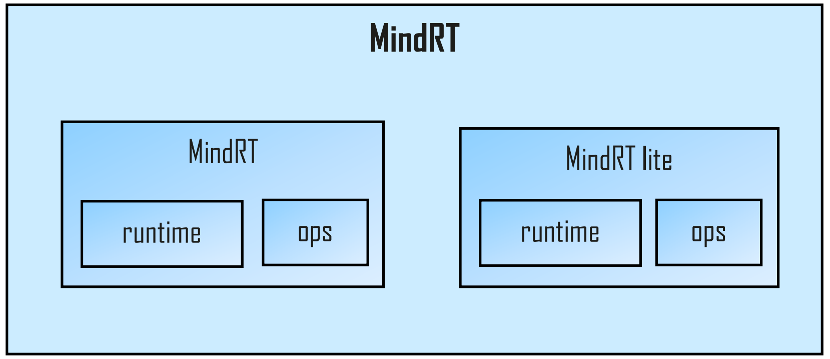
- High-level APIs provide management, callback, quantization, mixed precision, and parallel control APIs for training and inference, facilitating process control on the entire network.
- Low-level APIs provide basic tensors, cells, NN-layers, optimizers, and initialization, helping users flexibly build networks and control execution processes.
- The executor controls computing execution and interacts with the MindSpore backend.

Subsystem: MindCompiler

- **MindCompiler** provides the **just-in-time compilation** capability for **MindIR**.
  - **Graph high level optimization (GHLO)** is application-oriented and provides frontend optimization and functions, such as Type derivation, automatic differentiation, second-order optimization, and automatic parallelism.
  - **Graph low level optimization (GLLO)** is hardware-oriented and performs bottom-layer optimization, such as operator fusion, layout optimization, redundancy elimination, and memory optimization.



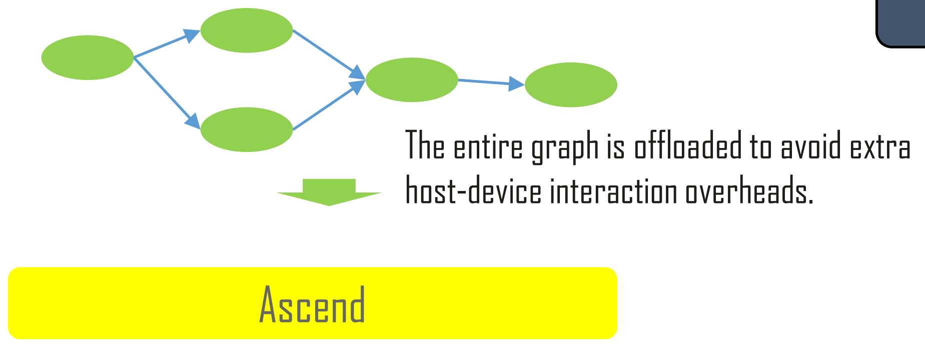
Subsystem: MindRT

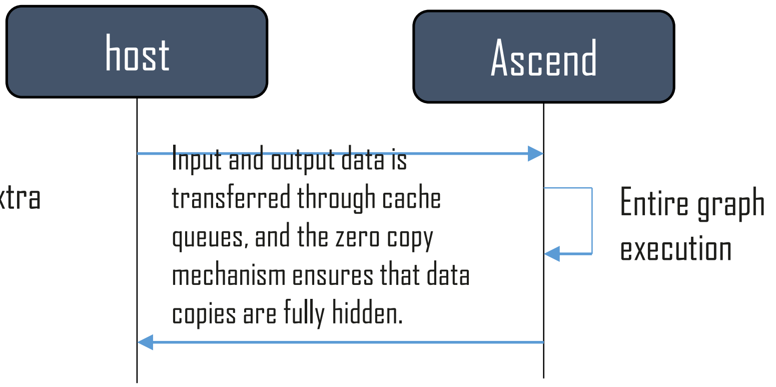


The centralized runtime system supports:

- Multiple device types on the device and cloud
- Scheduling management of multiple hardware platforms, such as Ascend, GPU, and CPU
- Memory pooling management and efficient memory overcommitment
- Asynchronous operators, heterogeneous execution, and multi-flow concurrency

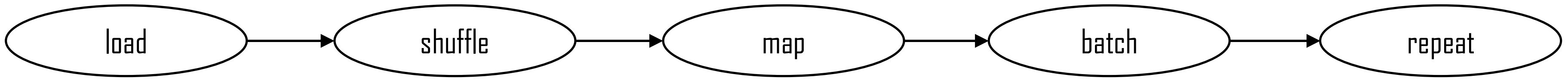
Technology features


The entire graph is offloaded to avoid extra host-device interaction overheads.
Ascend

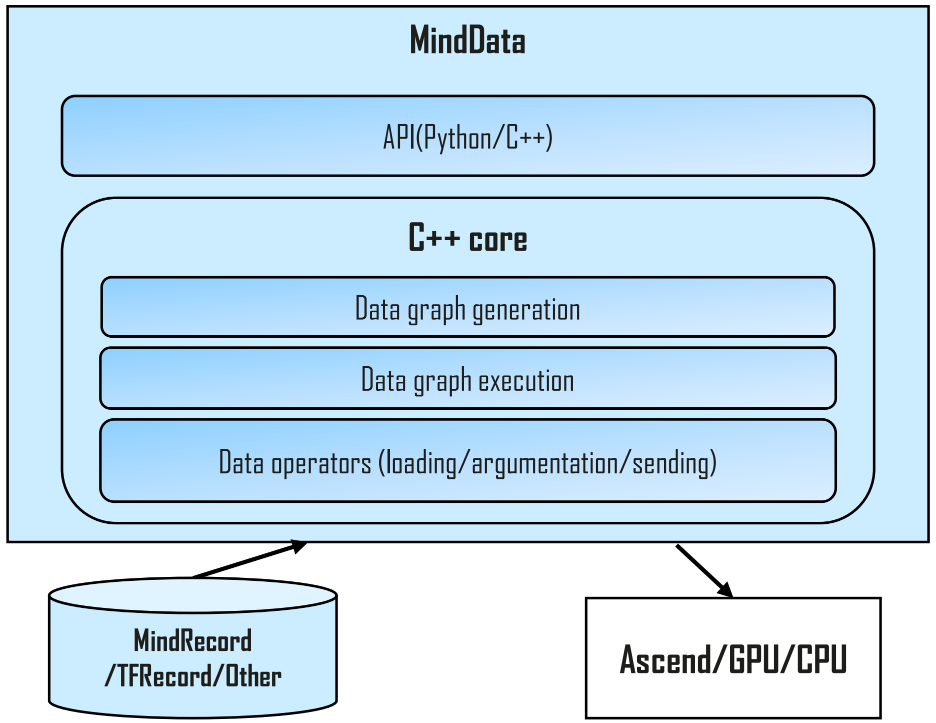

host
Ascend
Input and output data is transferred through cache queues, and the zero copy mechanism ensures that data copies are fully hidden.
Entire graph execution

Subsystem: MindData

MindData is responsible for efficiently executing the training data processing pipeline, forming a pipeline with computing, and promptly importing data for training.



Typical training data processing pipeline



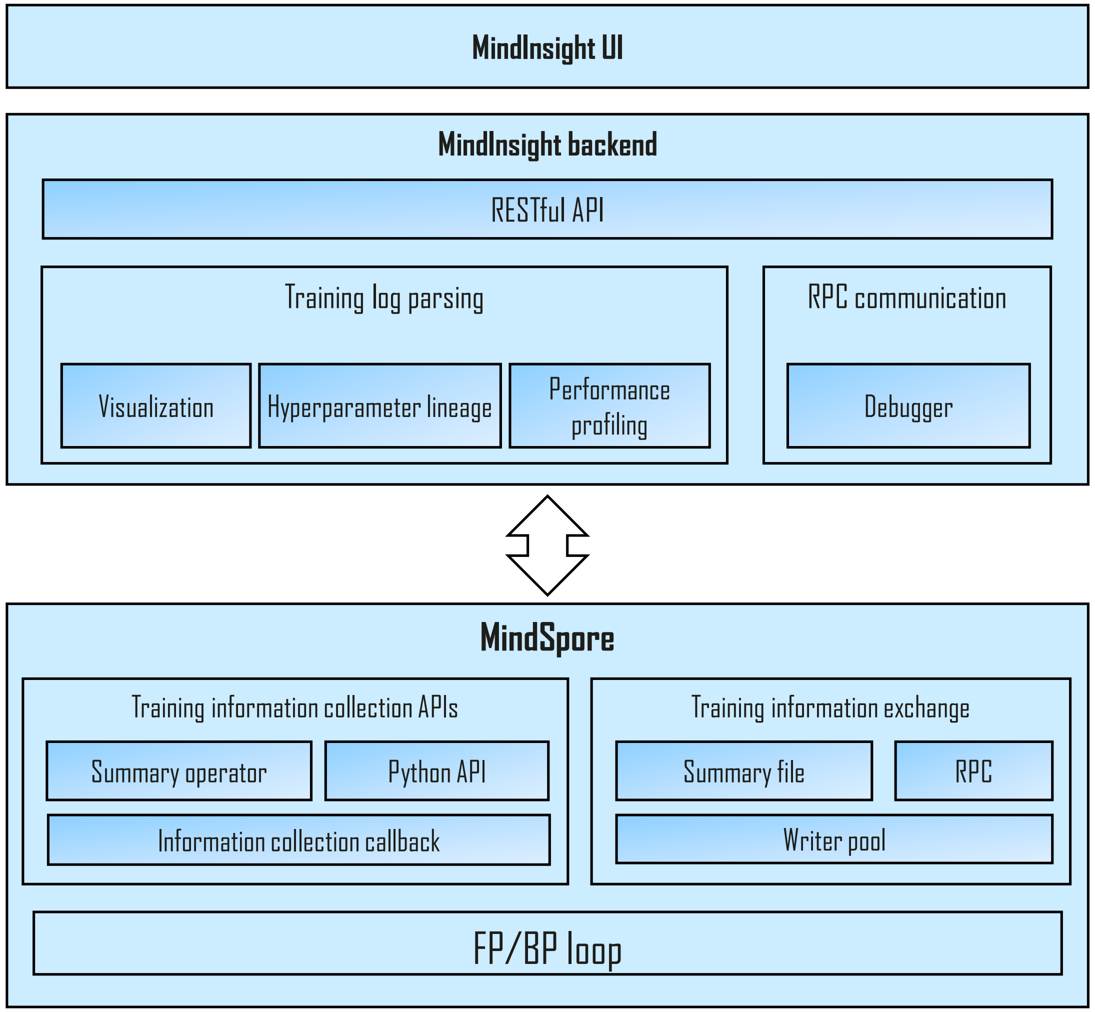
Key functions:

- Pipeline + parallel execution, improving data processing throughput
- Various data operators
- User-defined Python operators and pipelines (data loading, sampling, and argumentation)
- Heterogeneous hardware acceleration (Ascend/GPU/CPU)
- MindRecord: built-in metadata and aggregated storage

Running process:

1. Data graph generation: Data graphs are generated based on Python APIs called by users.
2. Data graph execution: The pipeline executes data operators in a data graph; this happens in parallel to complete dataset loading, shuffle, data argumentation, and batch processing.
3. Importing data to device: The processed data is imported to the device for training.

Subsystem: MindInsight



MindInsight is the **debugging and optimization subsystem** of MindSpore. It provides the training process visualization, model lineage, debugger, and performance profiling functions.

Key functions:

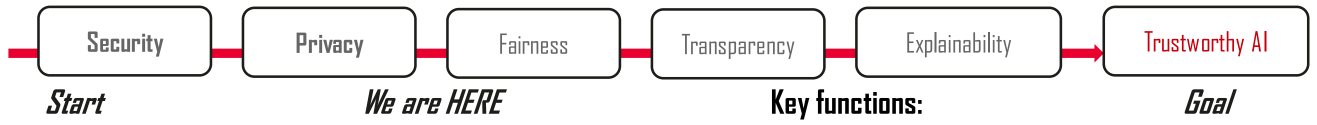
- APIs are easy to use, enabling users to easily collect training process metrics, including computational graphs, scalar data (such as loss and accuracy), histogram data (such as gradient and weight), and performance data, and display them on the web UI.
- Collect training hyperparameters, datasets, and data augmentation information to implement model lineage and compare training results.

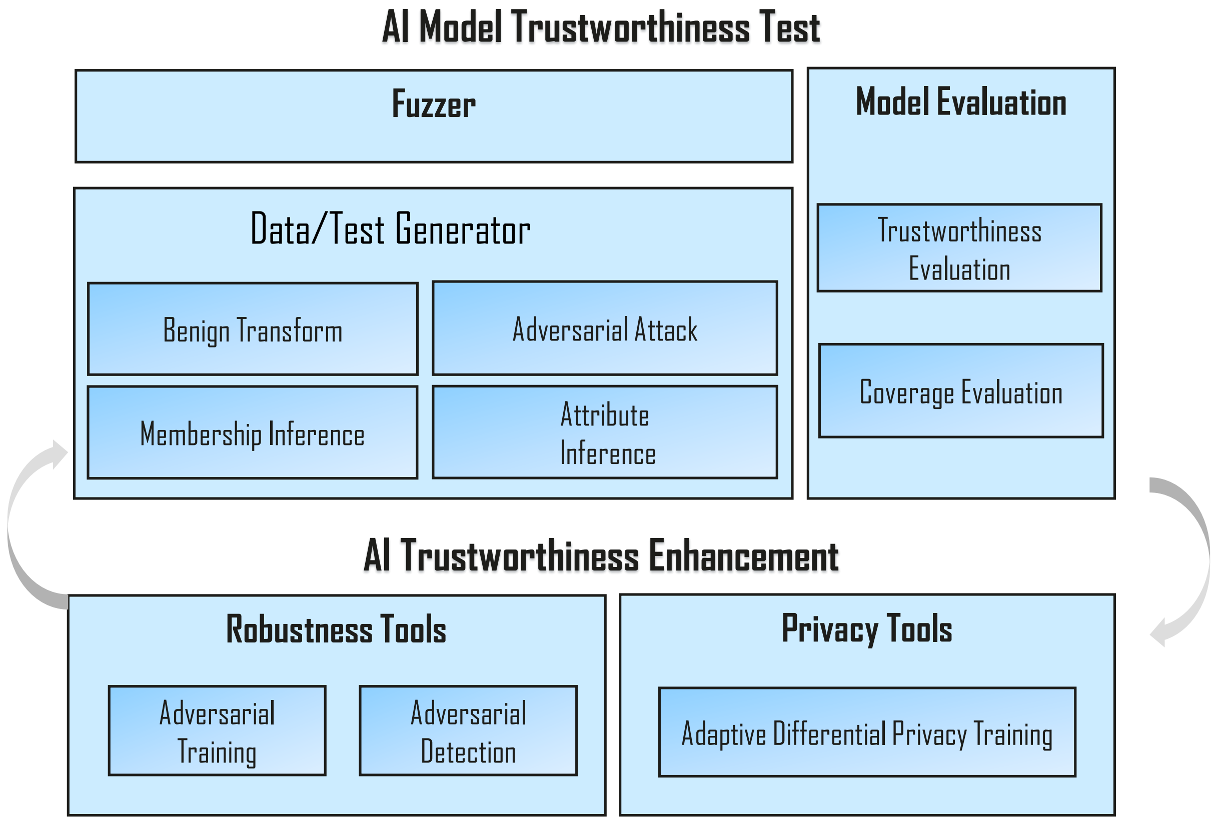
Running process:

1. Collecting training information: Users collect common training indicators using the callback API, and can decide which information to collect based on their requirements. For example, use the summary operator to collect information about the computational graph and the Python API for information about the Python layer.
2. Generating training logs: Training logs are generated based on the process information collected during training.
3. Displaying training information: MindInsight opens and parses training logs to display the training process information in a graph.

Subsystem: MindArmour

MindArmour provides comprehensive, effective, and easy-to-use evaluation tools and enhancement methods for AI trustworthiness in each domain.





Key functions:

- Test data generation methods in all scenarios, such as black-and-white box adversarial attacks, member and attribute inference attacks, and data drifts.
- Coverage-based fuzzing test process as well as flexible and customizable test strategies and indicators.
- Common methods of adversarial example detection and model robustness enhancement, including adversarial training and input rebuilding.
- Efficient, adaptive, and differential privacy training and budget statistics algorithms, as well as mathematically proven model for privacy leakage constraints.

Running process:

1. Configuring strategies: Define test strategies based on threat vectors and trustworthiness requirements, and select the appropriate test data generation methods.
2. Executing fuzzing test: Generate trustworthiness test data heuristically based on the model coverage rate and configured strategies.
3. Generating evaluation reports: Generate said reports based on built-in or user-defined trustworthiness indicators.
4. Enhancing trustworthiness: Use preset methods to enhance the trustworthiness of AI models.

Contents

MindSpore Feature: Automatic Parallelism

- Efficient hybrid parallel of ultra-large models, and smooth expansion of computing power

Challenges

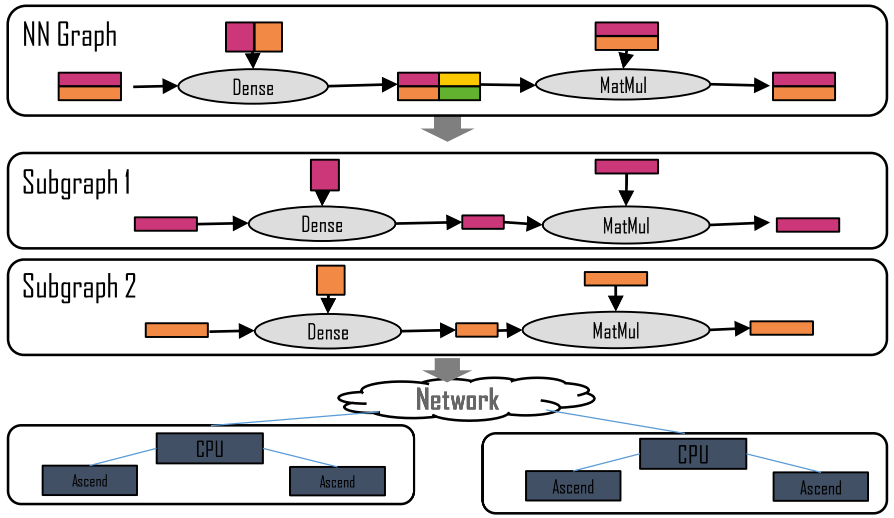
Challenges to efficient distributed training of ultra-large models:

NLP models become larger and larger. The memory overhead for training ultra-large models such as BERT (340 million)/GPT-2 (1542 million) exceeds the capacity of a single device. Therefore, the model needs to be partitioned into multiple devices for execution.

Currently, manual model parallelism requires model partitioning design and cluster topology awareness, which is difficult to develop, and it is hard to ensure high performance and perform tuning.

MindSpore Key Features

Automatically partition an entire graph based on the input and output data of the operator, and integrate data parallelism and model parallelism. **Cluster topology aware scheduling:** The cluster topology is aware, and subgraphs are automatically scheduled to minimize communication costs.



Effect: The standalone code logic is kept to implement model parallelism, improving development efficiency by 10 times compared with manual parallelism!

MindSpore Feature: Second-order Optimization

- CNN training acceleration is supported.
- The training convergence speed is accelerated by 20.6% based on ResNet-1.5@ImageNet2012.

Challenges

Deep learning model training requires a large amount of computing power, and training convergence takes a long time.
The second-order optimization method accelerates model convergence and reduces the number of training steps. However, it introduces a large number of complex computation, limiting its application in deep model training.
The second-order optimizer parameters are updated as follows:

$$\theta^{(t+1)} = \theta^{(t)} - \epsilon I M_{\theta^{(t)}}^{-1} \nabla g(\theta^{(t)})$$

Parameter

Learning rate

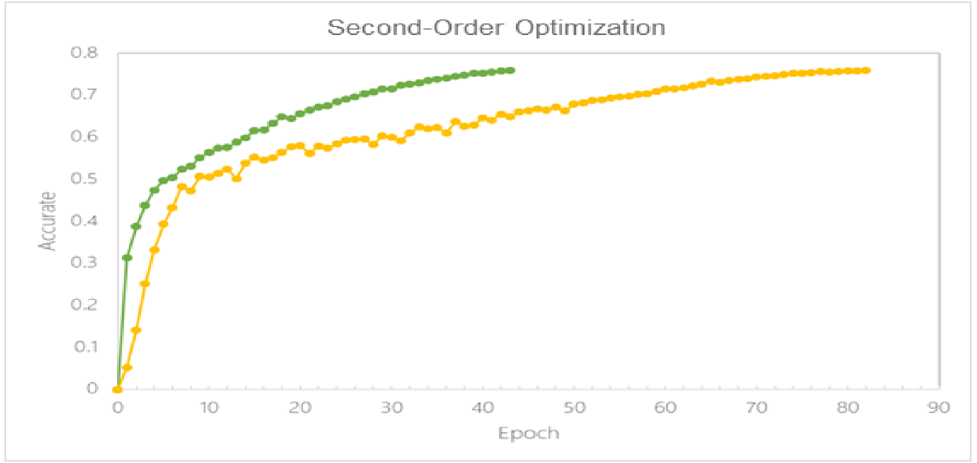
Second-order information matrix

First-order gradient

Core problem: The second-order optimizer needs to compute the inverse matrix of the second-order information matrix. The computation workload is heavy, and it can take hours to solve the second-order matrix directly, creating a technical difficulty.

MindSpore Key Features

The second-order matrix is approximated to reduce the computational complexity, and then the frequency and dimension of the matrix are reduced to accelerate the computation.



Optimizer	Epoch	Convergence Time	Test Scale
SGD+MOMENTUM	About 60	90 minutes	8-device Ascend 910
MindSpore second-order optimization	42	71.5 minutes	8-device Ascend 910

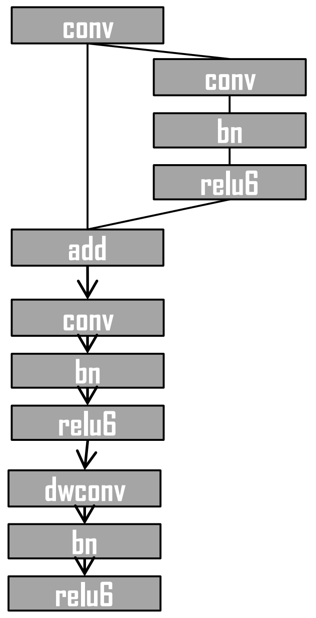
MindSpore Feature: On-Device Execution

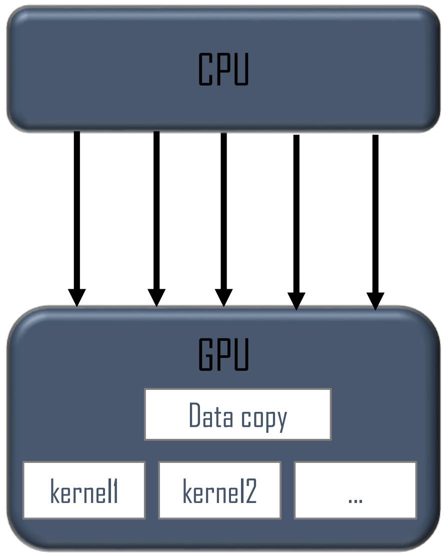
- The entire graph is offloaded to devices, maximizing the computing power of Ascend.

Challenges

Challenges to model execution with powerful chip computing power:

Memory wall problems, high interaction overhead, and difficult data supply. Some operations are performed on the host, while others are performed on the device. The interaction overhead is much greater than the execution overhead. As a result, the accelerator usage is low.

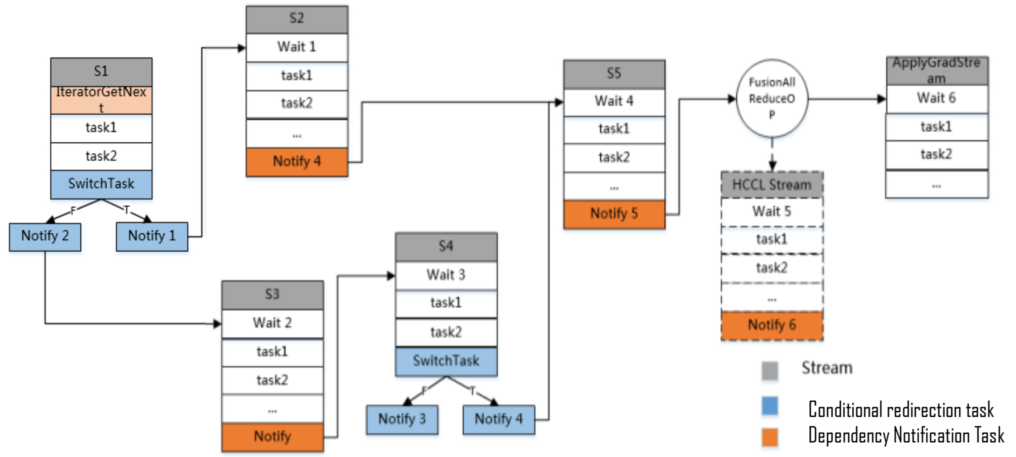




High data interaction overhead and difficult data supply.

MindSpore Key Features

The chip-oriented deep graph optimization is used to reduce synchronization waiting time and maximize the parallelism degree of "data-computing-communication". Data + Entire computational graph to the Ascend chips.



Effect: Compared with the host-side graph scheduling mode, the training performance is improved by 10 times!

MindSpore Feature: Deployment and Collaboration in All Scenarios

The IR of the unified model copes with the upper-layer differences in different language scenarios.

Challenges

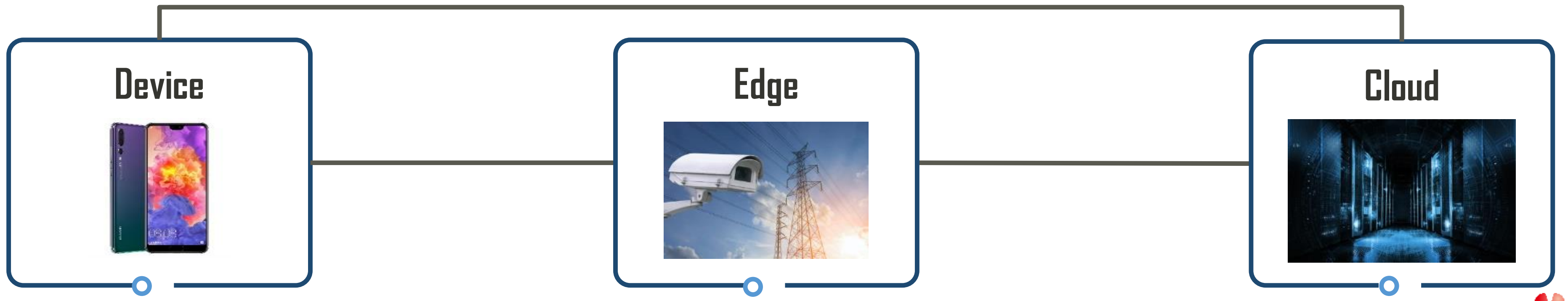
The diversity of hardware architectures leads to deployment differences and performance uncertainties in all scenarios, and the separation of training and inference results in model isolation.

MindSpore Key Features

- **Unified model IR** brings consistent deployment experience.
- The graph optimization technology based on software and hardware collaboration shields scenario differences.
- Federal meta learning based on device-cloud synergy breaks the boundaries of devices and the cloud. The multi-device collaboration model is updated in real time.

Effect: In the unified architecture, the deployment performance of models in all scenarios is consistent, and the accuracy of personalized models is improved!

On-demand collaboration in all scenarios and consistent development experience

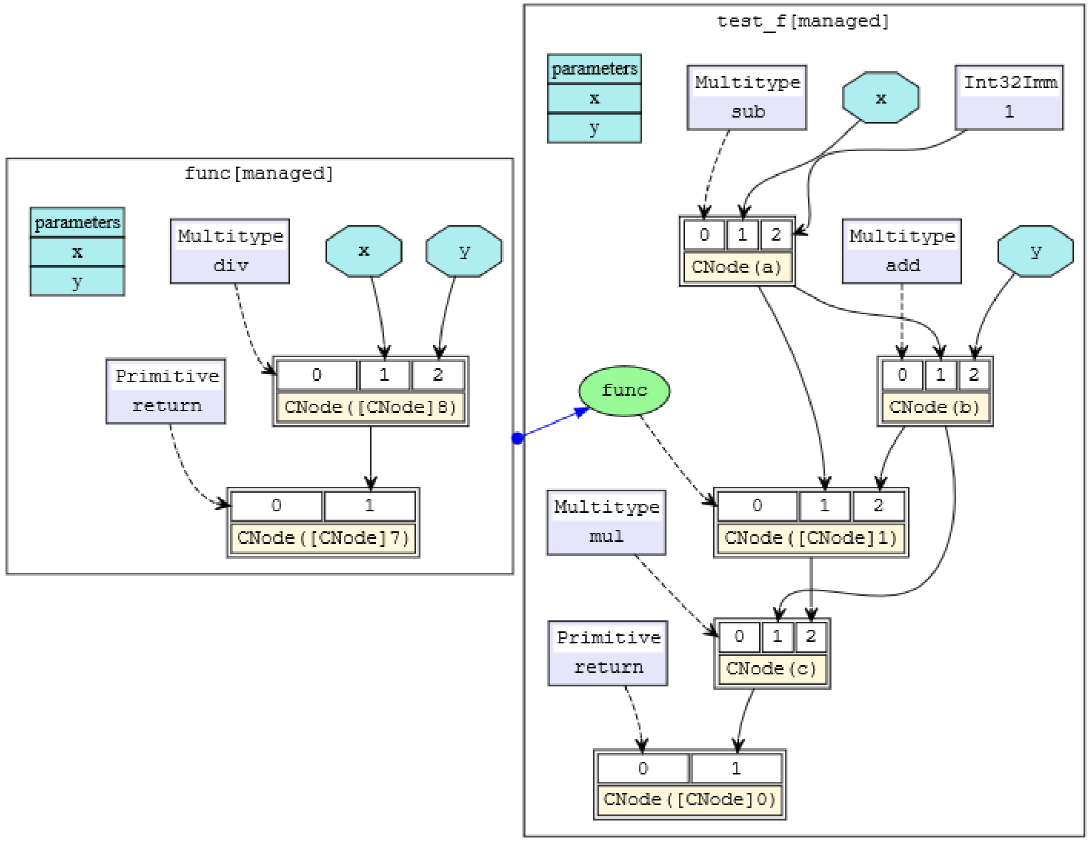


MindSporeIR

- **MindSporeIR** is **a simple**, **efficient**, and **flexible** graph-based functional IR that can represent functional semantics such as free variables, higher-order functions, and recursion.

- **It is the program carrier in the process of auto differentiation and compilation optimization.**

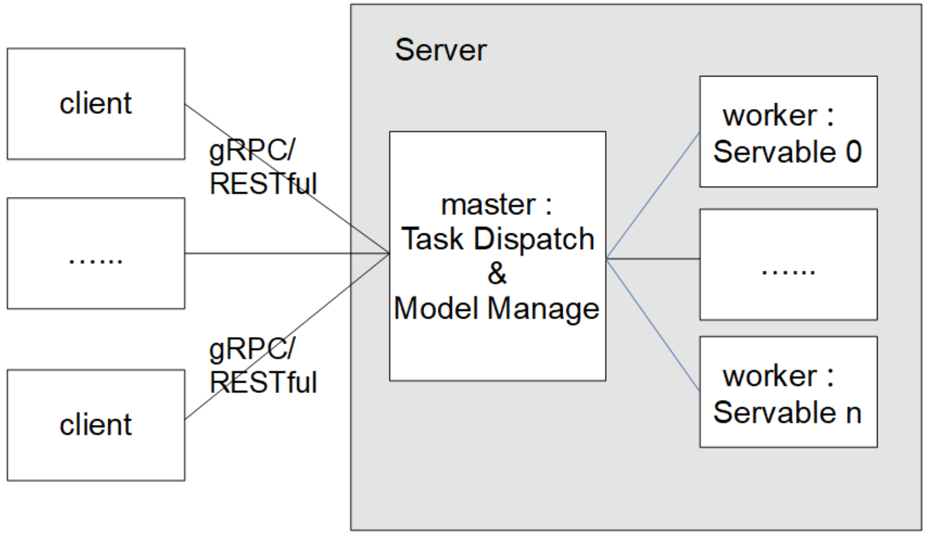
- Each **graph** represents a **function** definition graph, which consists of ParameterNode, ValueNode, and ComplexNode(CNode).
- The figure shows the def-use relationship.



MindSpore Serving: Efficient Deployment of Online Inference Services

MindSpore Serving is a lightweight and high-performance service module that helps MindSpore developers efficiently deploy online inference services in the production environment.

- **Easy-to-use**
- **One-click release and deployment**
- **Batching**
- **High performance and scalability**



MindSpore Serving structure

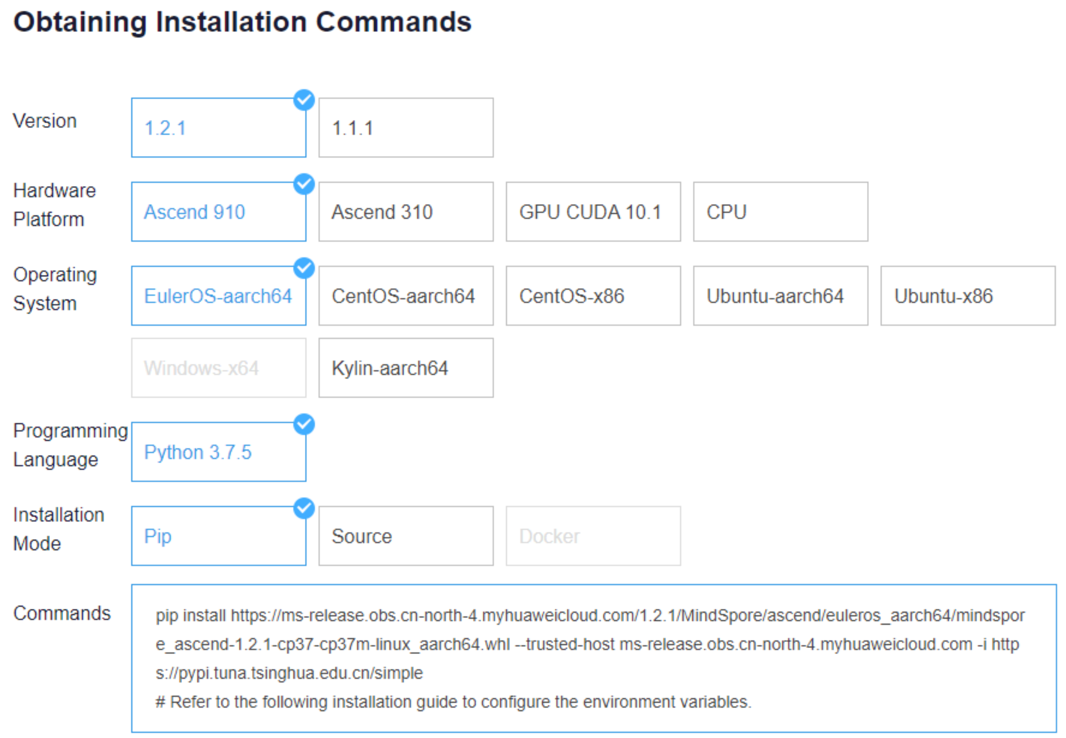
https://gitee.com/mindspore/serving/blob/r1.1/README_CN.md#%E9%85%8D%E7%BD%AE%E7%8E%AF%E5%A2%83%E5%8F%98%E9%87%8F

Contents

Installing MindSpore

For details about how to install MindSpore, visit https://mindspore.cn/install/en.

MindSpore supports platforms such as Windows, Ubuntu, and CentOS, and hardware such as Ascend 910, CPU, and GPU.



Obtaining Installation Commands

Version	[✓ 1.2.1] 1.1.1
Hardware Platform	[✓ Ascend 910] Ascend 310 GPU CUDA 10.1 CPU
Operating System	[✓ EulerOS-aarch64] CentOS-aarch64 CentOS-x86 Ubuntu-aarch64 Ubuntu-x86 Windows-x64 Kylin-aarch64
Programming Language	[✓ Python 3.7.5]
Installation Mode	[✓ Pip] Source Docker

Commands

```
pip install https://ms-release.obs.cn-north-4.myhuaweicloud.com/1.2.1/MindSpore/ascend/euleros_aarch64/mindspor
e_ascend-1.2.1-cp37-cp37m-linux_aarch64.whl --trusted-host ms-release.obs.cn-north-4.myhuaweicloud.com -i http
s://pypi.tuna.tsinghua.edu.cn/simple
# Refer to the following installation guide to configure the environment variables.
```

MindSpore Experience

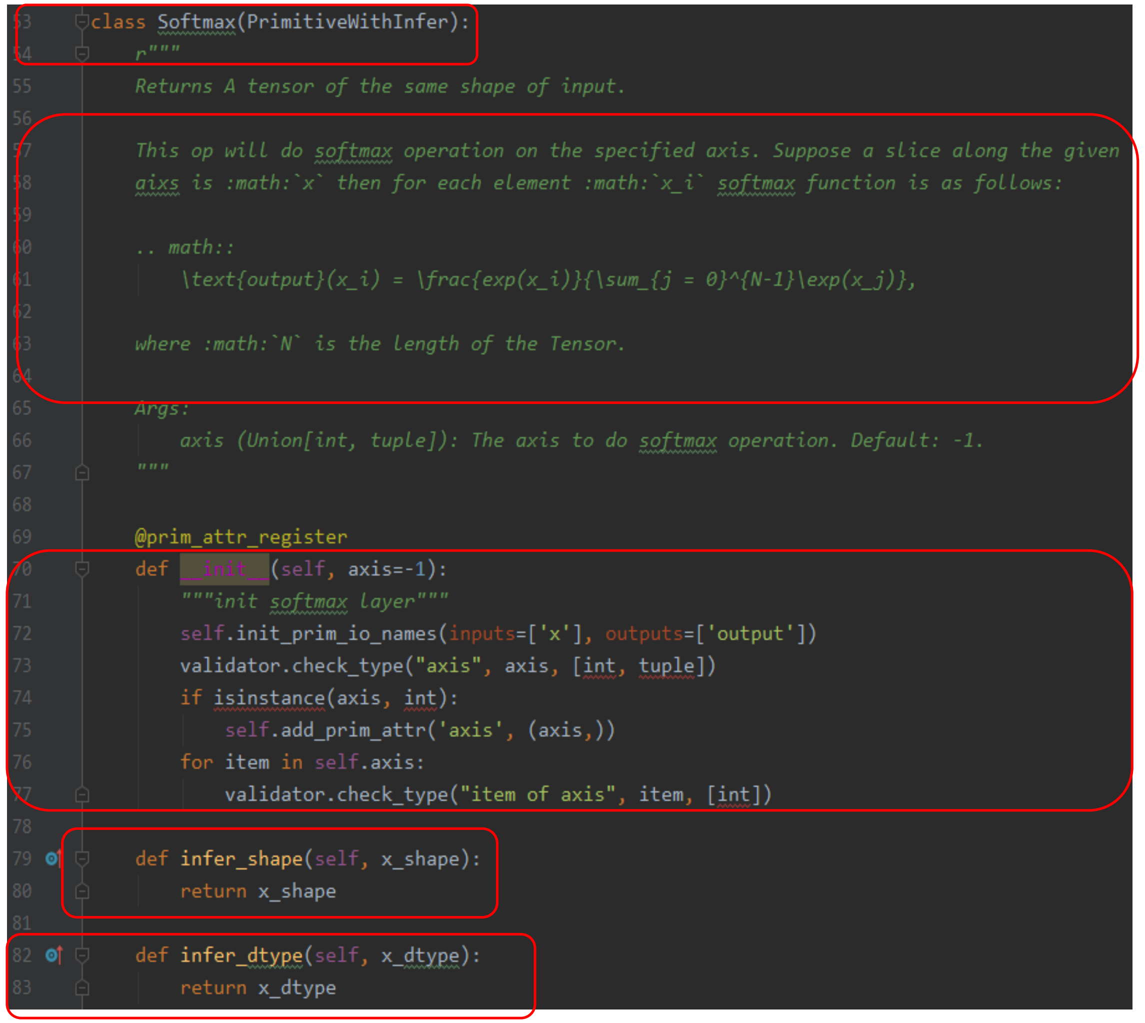
- In MindSpore, the data storage component is a tensor. Common tensor operations are as follows:
  - asnumpy()
  - size()
  - dim()
  - dtype()
  - set_dtype()
  - tensor_add(other: Tensor)
  - tensor_mul(ohter: Tensor)
  - shape()
  - __str__# Convert into a character string.

Component	Description
model_zoo	Definition of common network models
communication	Data loading module, which provides the dataloader, dataset definition, and data processing functions such as image and text processing
dataset	Dataset processing module, such as data reading and preprocessing
common	Definitions of tensor, parameter, dtype, and initializer
context	Context class definition, which is used to set parameters for model running, for example, switching to graph or pynative mode
akg	Automatic differentiation and custom operator libraries
nn	Definitions of MindSpore cell, loss function, and optimizer
ops	Basic operator definition and backward operator registration
train	Training model-related and summary function modules
utils	Utilities mainly for parameter validation (for internal framework use)

ME Module Components

MindSpore Programming Concept: Operator

Softmax operator



```
class Softmax(PrimitiveWithInfer):
    r"""
    Returns A tensor of the same shape of input.

    This op will do softmax operation on the specified axis. Suppose a slice along the given
    aixs is :math:`x` then for each element :math:`x_i` softmax function is as follows:

    .. math::
        \text{output}(x_i) = \frac{exp(x_i)}{\sum_{j = 0}^{N-1}\exp(x_j)},

    where :math:`N` is the length of the Tensor.

    Args:
        axis (Union[int, tuple]): The axis to do softmax operation. Default: -1.
    """

    @prim_attr_register
    def __init__(self, axis=-1):
        """init softmax layer"""
        self.init_prim_io_names(inputs=['x'], outputs=['output'])
        validator.check_type("axis", axis, [int, tuple])
        if isinstance(axis, int):
            self.add_prim_attr('axis', (axis,))
        for item in self.axis:
            validator.check_type("item of axis", item, [int])

    def infer_shape(self, x_shape):
        return x_shape

    def infer_dtype(self, x_dtype):
        return x_dtype
```

1. Operator name and base class

2. Operator comment

3. Operator initialization. The operator attribute values are initialized.

4. Shape derivation

5. data_type derivation

Common MindSpore operators are as follows:

- array: Array-related operators

- ExpandDims - Squeeze
- Concat - OnesLike
- Select -StridedSlice
-ScatterNd, etc.

- math: Mathematical operators

- AddN - Cos
- Sub - Sin
- Mul - LogicalAnd
- MatMul - LogicalNot
- RealDiv - Less
- ReduceMean - Greater, etc.

- nn: Network operators

-Conv2D - MaxPool
-Flatten - AvgPool
- Softmax - TopK
- ReLU - SoftmaxCrossEntropy
- Sigmoid - SmoothL1Loss
- Pooling - SGD
- BatchNorm - SigmoidCrossEntropy,
- etc.

- control: Control operators

- ControlDepend

-random: Random number-related operators

MindSpore Programming Concept: Cell

- A **cell** provides basic modules that define computing execution. Cell objects can be directly executed.
  - **__init__**: initializes and verifies components such as parameters, cells, and primitives.
  - **construct**: defines the execution process; in graph mode, a graph is compiled for execution, which is subject to specific syntax restrictions.
  - **bprop** (optional): backward propagation of user-defined modules; if this method is undefined, the framework automatically generates a backward graph to compute the backward propagation of the construct part.
- The following cells are predefined in MindSpore: loss functions (SoftmaxCrossEntropyWithLogits and MSELoss), optimizers (Momentum, SGD, and Adam), network packaging functions (TrainOneStepCell for network gradient calculation and update, and WithGradCell for gradient calculation).

Contents

Application caes

Computer Vision

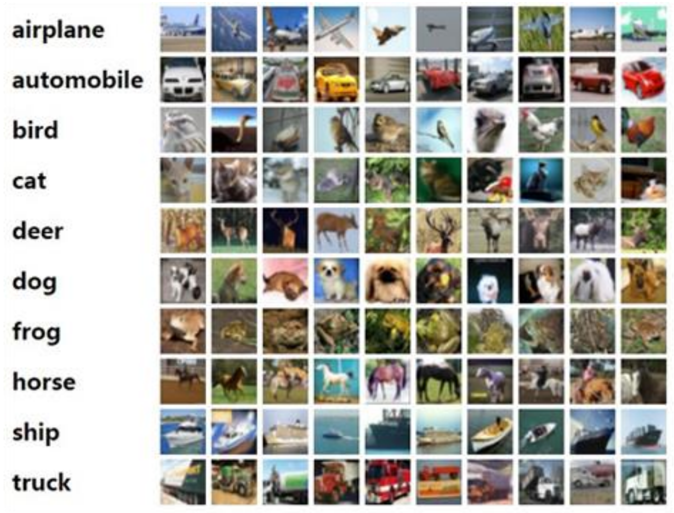


Image classification
arXiv:1512.03385

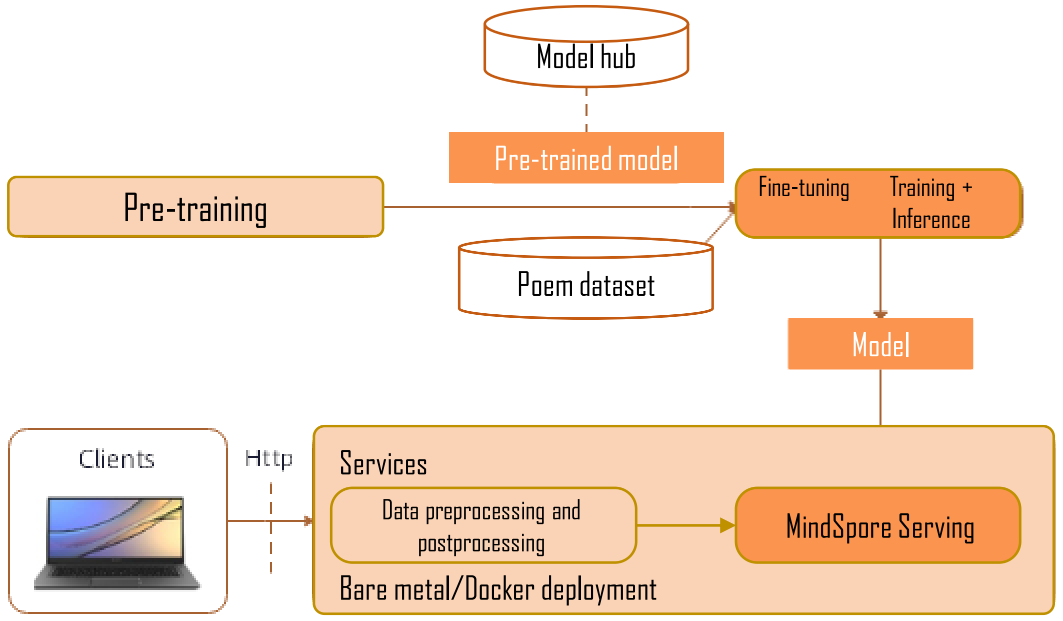
Object segmentation
arXiv:1703.06870

Keypoint detection
arXiv:1611.08050

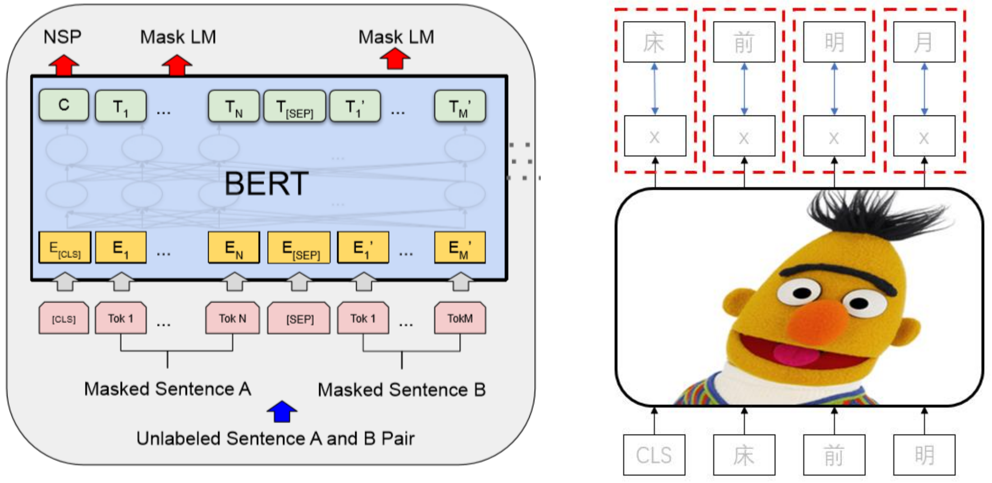
https://www.mindspore.cn/tutorial/training/zh-CN/r1.1/advanced_use/cv.html#
https://arxiv.org/pdf/1611.08050.pdf
https://arxiv.org/pdf/1706.05587.pdf

Natural Language Processing (NLP)

Use MindSpore to train intelligent poetry writing models and deploy prediction services



Application flowchart



Bert model and training

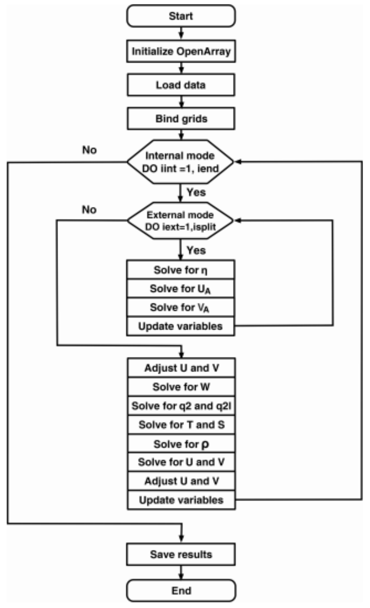
https://www.mindspore.cn/tutorial/training/en/r1.1/advanced_use/nlp_bert_poetry.html

High Performance Computing (HPC)

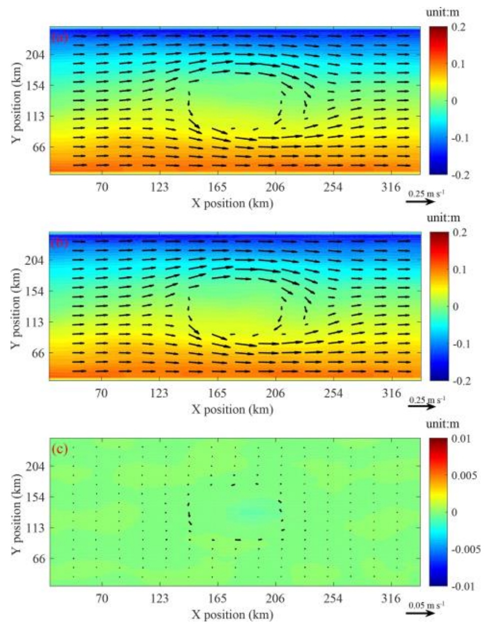
Computing faces unprecedented challenges due to massive amounts of data and access devices. Therefore, AI and HPC are slated to converge in the future, with AI transforming tradition HPC.

The 3D ocean model is key for the entire earth system model. By simulating ocean currents and whirlpools, it can predict typhoons and tsunamis in real time.

However, the code for conventional ocean models is complex and they run on CPUs. However, MindSpore accelerates the GOMO model framework, which runs on a GPU, significantly improving the model performance.



GOMO flowchart

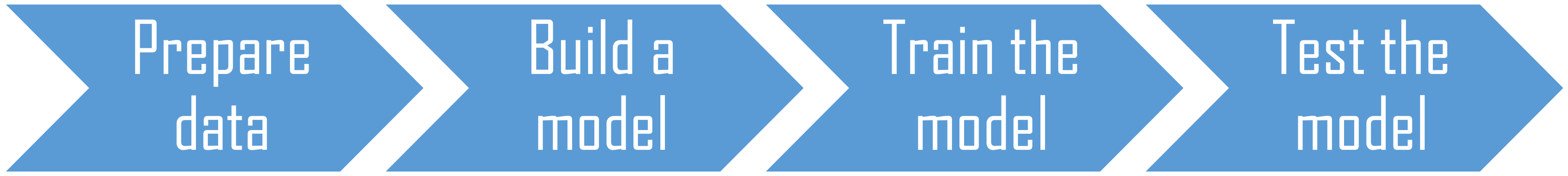


GOMO prediction

https://gmd.copernicus.org/articles/12/4729/2019/gmd-12-4729-2019.pdf

MindSpore Application Case

- We will use the MNIST handwritten digit recognition to demonstrate the MindSpore-based modeling process.



- Data preparation
- Data visualization
- Data preprocessing

- Model architecture
- Optimizer
- Loss function
- Evaluation indicator

- Batch size and epoch setting
- Callback

- Test sample processing

Quiz

1. Which of the following operators does **nn** belong to in MindSpore? ()

   A. Math-related operators

   B. Network operators

   C. Control operators

   D. Other operators

Summary

- This chapter introduces the MindSpore framework, design ideas, and features, and describes the MindSpore environment setup process and development procedure.

More Information

MindSpore official website: https://mindspore.cn/en

Huawei Talent Online website: https://e.huawei.com/en/talent/#/home

WeChat official accounts:

EMUI

Huawei Device Open Lab

Huawei Developer

Contact Huawei Talent Online

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home, and organization for a fully connected, intelligent world.

HUAWEI