Ascend | 开发者创享日

MindIR: the intermediate representation of MindSpore

Zichun Ye

2012 labs, Huawei

COMPUTATIONAL ABSTRACTIONS FOR PROBABILISTIC AND DIFFERENTIABLE PROGRAMMING WORKSHOP

2022.07.09

MindSpore: Open Source All-Scenario AI Framework

Mar 2020 — MindSpore Went open source

Sep 2020 — MindSpore 1.0 Commercial use

[M]s 昇思 MindSpore — **MindSpore 1.8**

2022 — MindSpore 2.0

Develop once Deploy on device, edge, and cloud

All-scenario applications

5,000+ applications online

All-scenario deployment

Free from cross-platform model conversion

Large-scale model training native support, accelerating innovation

200-billion-parameter model

Auto parallel

80% ↓

Parallel code

Auto tuning

60% ↓

Tuning time

MindSpore for science, accelerating AI for Science

MindSporeElec

Electromagnetic simulation kit

10x ↑

Mobile phone EM simulation performance

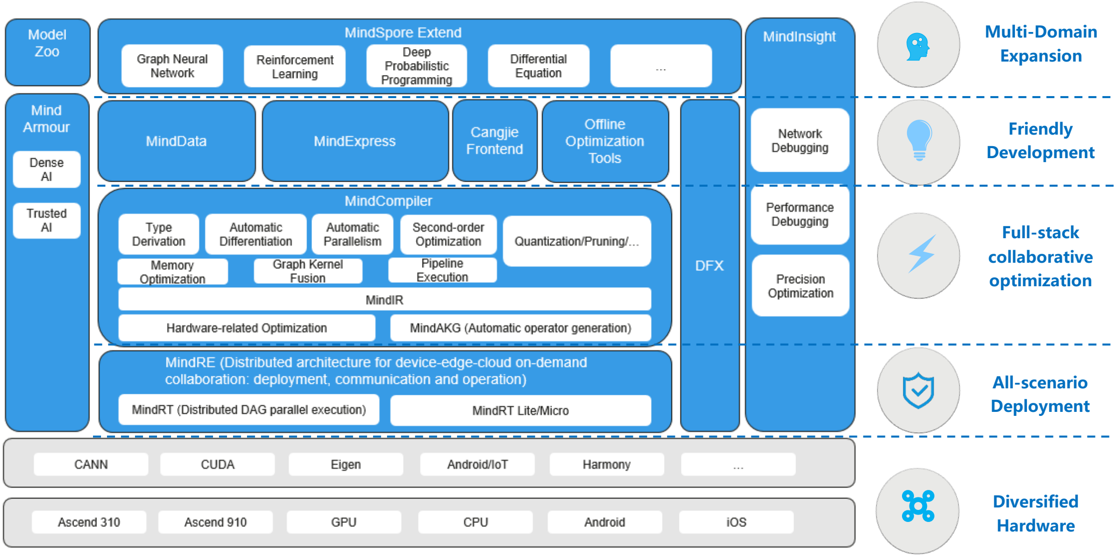
MindSpore SPONGE

Molecular simulation kit

50% ↑

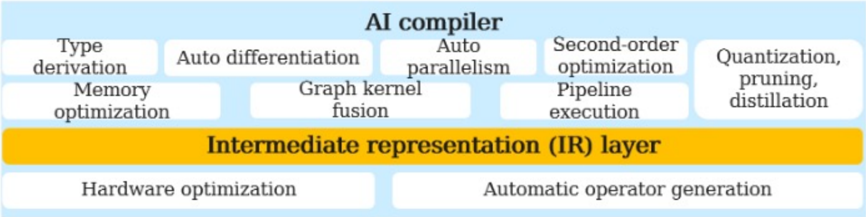
Biopharmaceutical compound computing efficiency

MindSpore: Open Source All-Scenario AI Framework



MindCompiler: the graph compiler of MindSpore

- In the AI software stack, the graph compiler optimizes the processing of a forward, or backward pass over the computation graphs that describe deep learning models.
- MindCompiler is the graph compiler of MindSpore to achieve three major kinds of optimizations, including
  - hardware-independent optimization (type inference, automatic differentiation, expression simplification, etc.);
  - hardware-related optimization (automatic parallelism, memory optimization, graph kernel fusion, pipeline execution, etc.);
  - deployment and inference-related optimizations (quantization, pruning, etc.).
- MindCompiler based on the unified device-cloud MindIR.



MindIR: A bit of motivation

- Compared other “general purpose” IRs, the IR of the graph compiler of AI framework has some special requirements and challenges:
    - Tensor representation;
    - Automatic differentiation;
    - JIT;
    - Implicit parallelism.
- Some hints from previous studies:
    - Sea of nodes: to construct graphs via use-def chains. The graph form is a single tiered structure instead of a two-tiered Control-Flow Graph (CFG) containing basic blocks (tier 1) of instructions (tier 2). control and data dependencies have the same form and implementation.
    - Thorin: a functional graph-based IR that abandons explicit scope nesting in favor of a dependency graph. The relationship between free variables and subgraphs is obtained via data dependency on the graph.
    - A-Normal form: partition expressions into two forms, atomic expressions and complex expression. All complex expression must be let-bound, or else appear in tail position.

```
<aexp> ::= NUMBER | STRING | VAR | BOOLEAN | PRIMOP
         |  (lambda (VAR …) <exp>)
<cexp> ::= (<aexp> <aexp> …)
         |  (if <aexp> <exp> <exp>)
<exp> ::= (let ([VAR <cexp>]) <exp>) | <cexp> | <aexp>
```

MindIR: a function-style graph-based IR

- MindSpore IR (MindIR) is a function-style IR based on graph representation.
  - Function-style IR makes automatic differentiation and implicit parallelism analysis more convenient;
  - Graph-based IR with a single tiered structure is suitable for JIT optimizations;
  - A-Normal form gets rid of scope and is easy to read and check.
- BNF: An variant A-Normal form used in MindIR with ‘if’ being treated as a prim operation
  - ➢ <Anode> ::= Scalar | Named | Tensor | Var | Prim | MetaFunc | Func | Type | Shape| Param
  - ➢ <CNode> ::= (<Anode> …)
  - ➢ <AnfNode> ::= <CNode> | <ANode>
- ANode in a MindIR corresponds to the atomic expression of ANF.
  - ValueNode refers to a constant node, which can carry a constant value (such as a scalar, symbol, tensor, type, and dimension), a primitive function (Primitive), or a common function (FuncGraph).
  - ParameterNode refers to a parameter node, which indicates the formal parameter of a function.
- CNode in a MindIR corresponds to the compound expression of ANF, indicating a function call.

[M]^S 昇思 MindSpore

MindIR: An example

Python Code

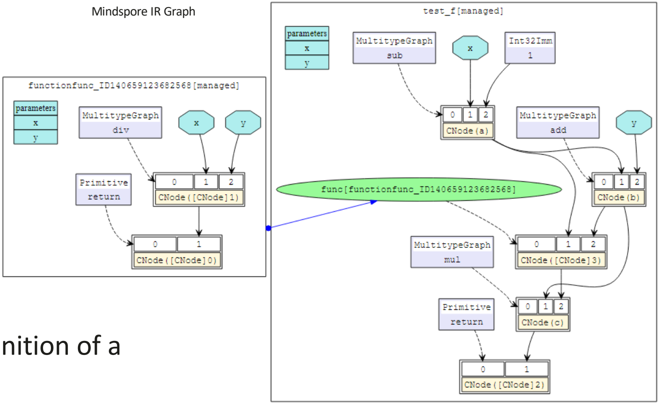
```python
def func(x, y):
    return x / y

def test_f(x, y):
    a = x - 1
    b = a + y
    c = b * func(a, b)
    return c
```

ANF

```
lambda (x, y)
    let a = x - 1 in
    let b = a + y in
    let func = lambda (x, y)
        let ret = x / y in
        ret end in
    let %1 = func(a, b) in
    let c = b * %1 in
    c end
```

Mindspore IR Graph



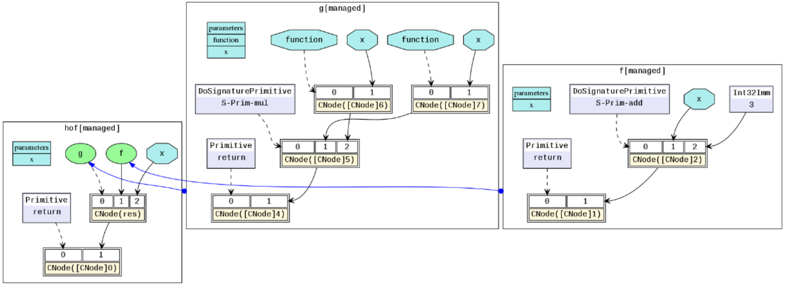
- In a MindIR, a function graph indicates the definition of a common function.
- Each expression is bound as a node, and the dependency is represented by using the directed edges between nodes.

MindIR: Higher-Order Functions

Python Code

```python
@ms_function
def hof(x):
    def f(x):
        return x + 3
    def g(function, x):
        return function(x) * function(x)
    res = g(f, x)
    return res
```

Mindspore IR Graph



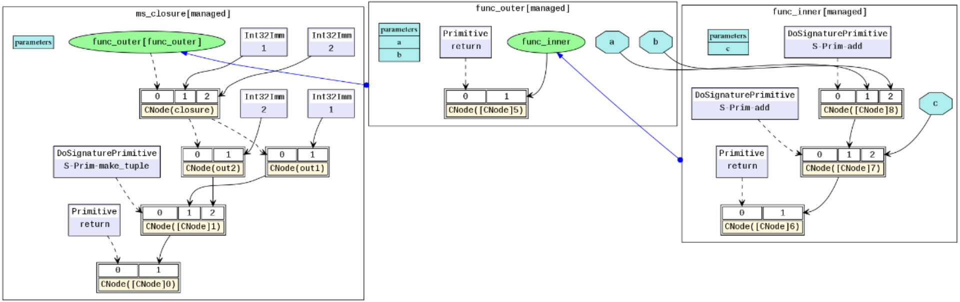
- In functional programming, the function definition itself is a value.
- In a MindIR, a function, defined by a subgraph, can be transferred as the input or output of other higher-order functions.
- Higher-order semantics greatly improve the flexibility and simplicity of MindSpore representations.

MindIR: Free Variables and Closures

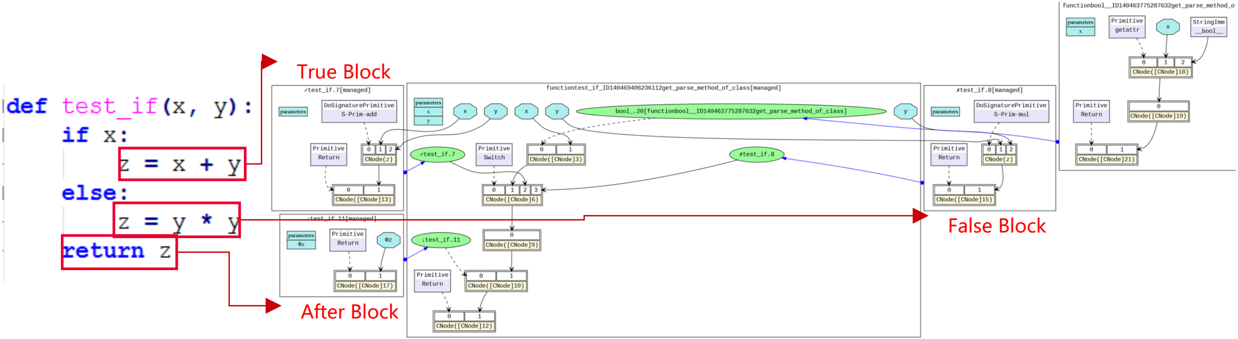
- In a MindIR, a code block is represented as a function graph.
- The scope environment can be considered as the context where the function is called.
- The capture method of free variables is value copy instead of reference.

```python
@ms_function
def func_outer(a, b):
    def func_inner(c):
        return a + b + c
    return func_inner

@ms_function
def ms_closure():
    closure = func_outer(1, 2)
    out1 = closure(1)
    out2 = closure(2)
    return out1, out2
```



MindIR: Control Flows



- In a MindIR, control flows are expressed in the form of high-order function selection and calling.
- This form transforms a control flow into a data flow of higher-order functions, making automatic differentiation of control flows possible.

MindIR: Autodiff

- MindIR implements automatic differentiation based source transformation.
- Each function call is transformed to return an additional value, which is a closure called the 'backpropagator'.
  - The backpropagator computes the derivative with respect to the inputs given the derivatives with respect to the outputs;
  - The backpropagators of primitives are known;
  - The backpropagators of user-defined functions is constructed by calling the backpropagators of the function calls in the body in reverse order by the chain rule.
- The design of MindIR help the automatic differential algorithm works in the case of control flows such as conditional jumps, loops, and recursion.

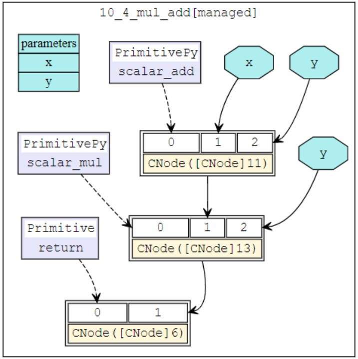
```
def f(x, y):
    a = pow(x, 3)
    b = pow(y, 4)
    c = mul(a, b)
    return c
```

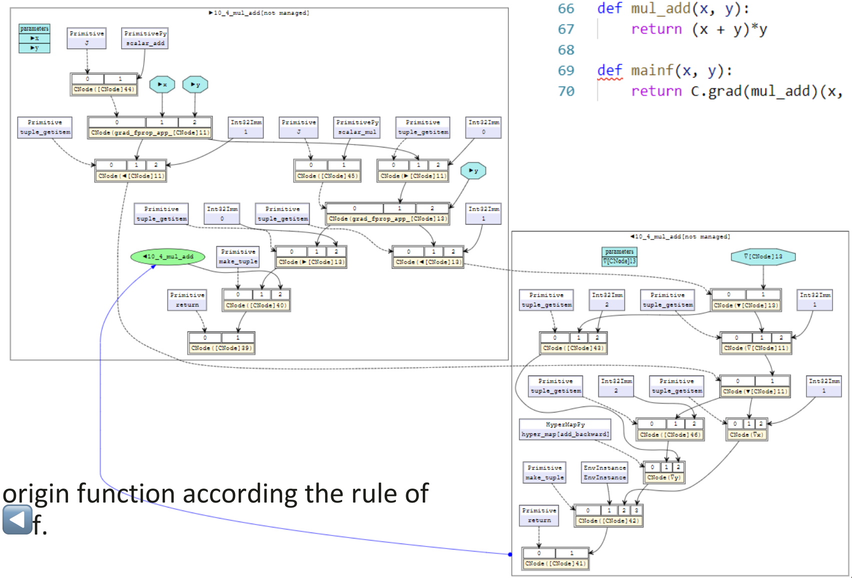
```
def ▶f(▶x, ▶y):
    ▶a, ◀a = ▶pow(▶x, ▶3)
    ▶b, ◀b = ▶pow(▶y, ▶4)
    ▶c, ◀c = ▶mul(▶a, ▶b)
    def ◀f(∇c):
        ∇x, ∇y, ∇a, ∇b, ∇c = 0...
        ∇mul_fv, ∇a, ∇b += ◀c(∇c)
        ∇pow_fv, ∇y, ∇exp_4 += ◀b(∇b)
        ∇pow_fv, ∇x, ∇exp_3 += ◀a(∇a)
        return {}, ∇x, ∇y
    return ▶c, ◀f
```

```
dfdx = ◀f(1.0)[1] = ▶f(x, y)[1](1.0)[1]
```

Reverse-Mode AD in a Functional Framework: Lambda the Ultimate Propagator by B. A. Pearlmutter and J. M. Siskind 2008

MindIR: Autodiff





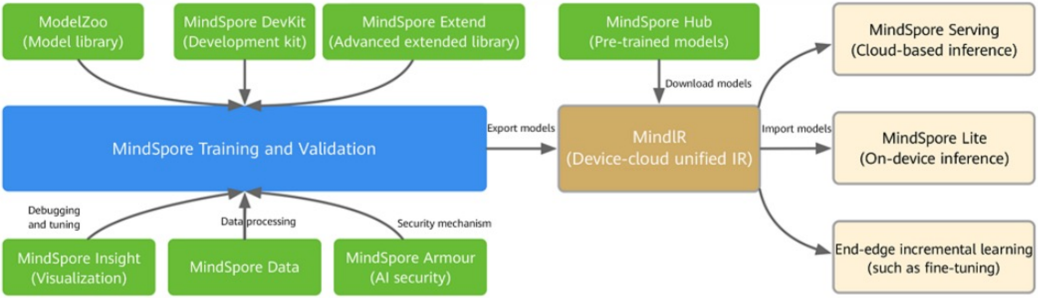
```python
def mul_add(x, y):
    return (x + y)*y

def mainf(x, y):
    return C.grad(mul_add)(x, y)
```

Perform auto diff transform on origin function according the rule of Generating ▶️f and Generating ◀️f.

MindIR: Execution Process

- As the unified model file of MindSpore, MindIR stores network structures and weight parameter values. In addition, it can be deployed on the on-cloud Serving and the on-device Lite platforms to execute inference tasks.
- A MindIR file supports the deployment of multiple hardware forms.
  - On-cloud deployment and inference on Serving: After MindSpore trains and generates a MindIR model file, the file can be directly sent to MindSpore Serving for loading and inference.
  - On-device inference and deployment on Lite: MindIR can be directly used for Lite deployment.




thank you
tenki
ขอบคุณ
takk
спасибо
kam sah hamnida
dank je
hvala
הדות
dhanyavadagalu
tack
дзякуй
dziękuję
bedankt
blagodaram
mèsi
xièxie
tanemirt
gracias
rahmet
enkosi
mochchakkeram
trugarez
djere dieuf
danke
kop khun krap
faafetai lava
arigatô
akun
shukriya
ありがとう
kia ora
dankon
děkuji
manana
diolch
grazzi
gràçie
kiitos
spas
ačiū
mamnun
barka
dhanyavad
gratias ago
शुक्रिया
sulpáy
tapadh leat
chnorakaloutioun
tau
dankie
gràcies
bayarlalaa
obrigada
teşekkür ederim
σας ευχαριστώ
sagolun
murakoze
taiku
mahalo
didi madloba
chokrane
rahmat
sukriya
obrigado
dakujem
misaotra
welalin
najis tuke
terima kasih
mercé
köszönöm
謝謝
ارك شك
asante
grazie
nandri
mersi
sobodi
mauruuru
merci
nanni
vinaka
matondo
cảm ơn bạn
go raibh maith agat
paldies
ngiyabonga