编程范式

ZOMI

关于本内容

1. **内容背景**

   - AI框架的基础介绍

2. **具体内容**

   ◦ AI框架作用：深度学习基础 – AI框架的作用 - AI框架的目的

   ◦ AI框架之争：第一代框架 – 第二代框架 – 第三代框架

   ◦ 编程范式：声明式编程 - 命令式编程

BUILDING A BETTER CONNECTED WORLD

Programming Language and Programming Model

Numpy

```
import numpy as np

N, D = 3, 4
x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)
```
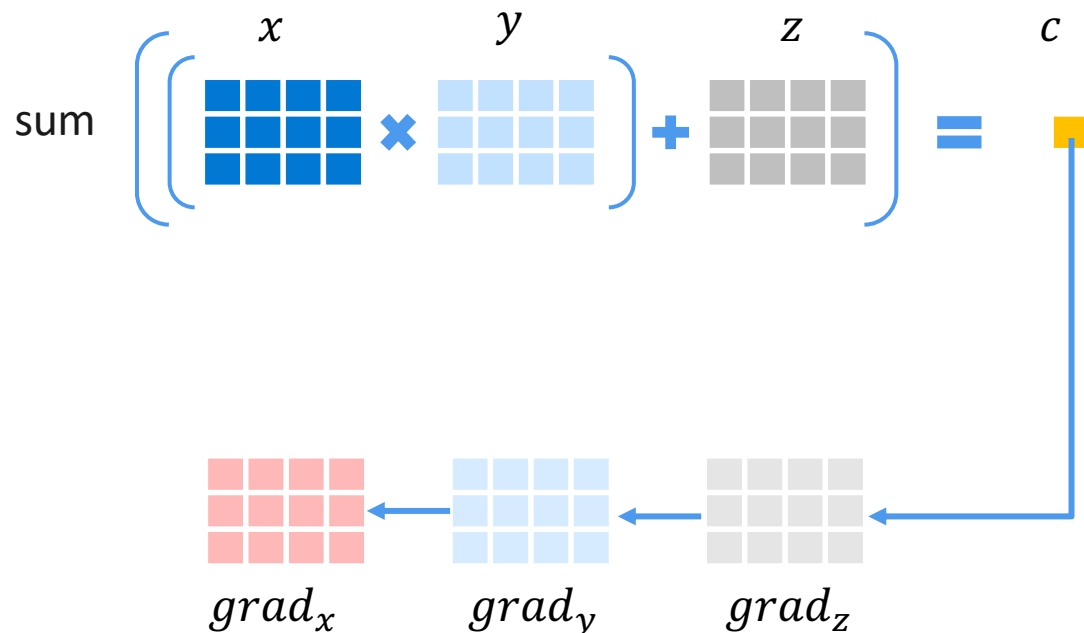
```
grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
grad_y = grad_a * x
```

Programming Language and Programming Model

PyTroch

```
import torch

N, D = 3, 4
x = torch.randn(N, D)
y = torch.randn(N, D)
z = torch.randn(N, D)
```
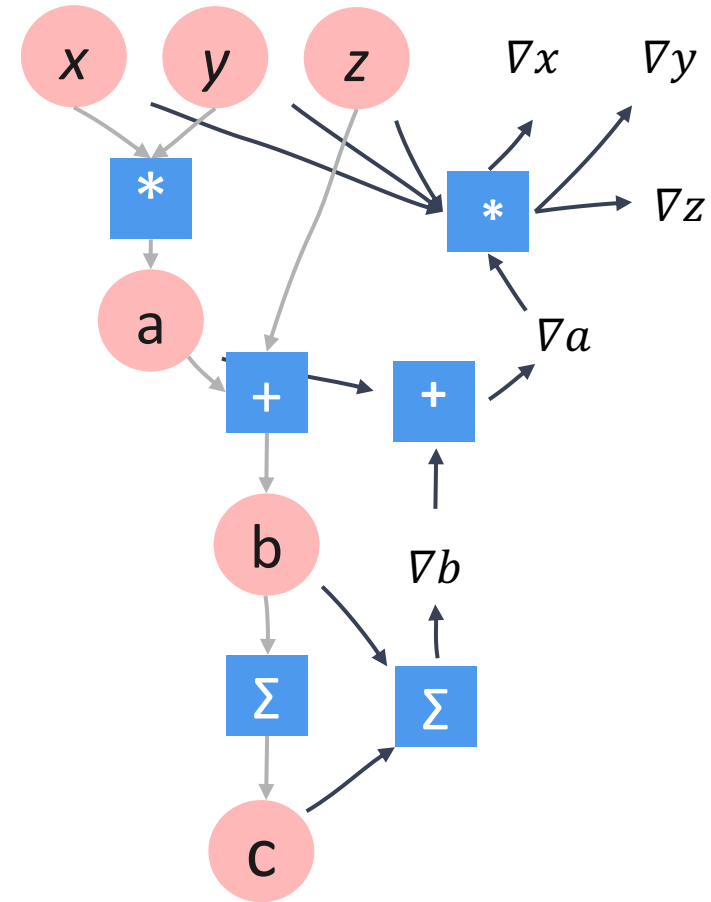
```
a = x * y
b = a + z
c = torch.sum(b)
```
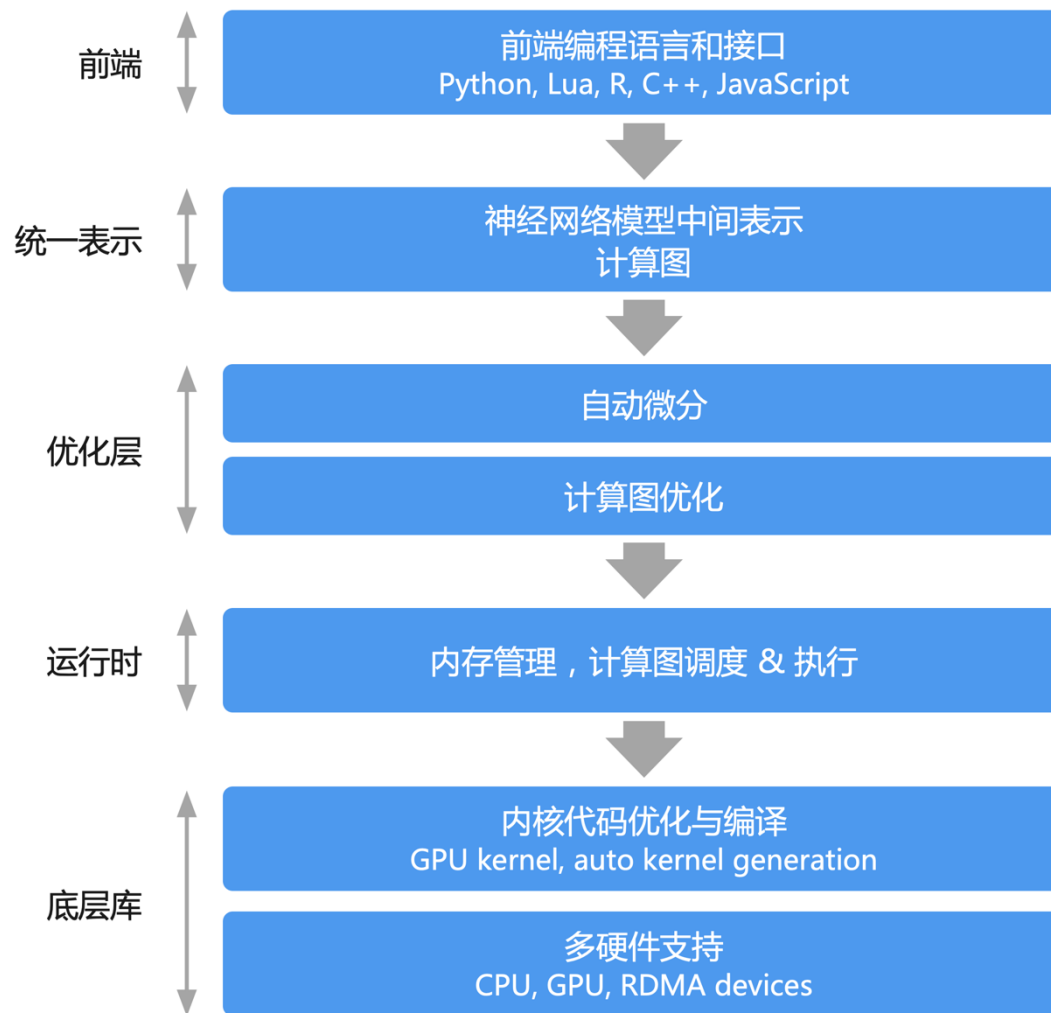
```
c.backward(retain_graph=True)
```

Huawei Confidential. Ascend & MindSpore

Programing Method

前端：

- 以高层次语言如Python作为前端
- 提供脚本式的编程体验

后端：

- 提供更低层次的编程模型和编程语言开发
- 后端高性能可复用模块与前端深度绑定
- 通过前端驱动后端方式执行

前端 ⇅ **前端编程语言和接口**
Python, Lua, R, C++, JavaScript

统一表示 ⇅ **神经网络模型中间表示**
计算图

优化层 ⇅ **自动微分**

⇅ **计算图优化**

运行时 ⇅ **内存管理，计算图调度 & 执行**

底层库 ⇅ **内核代码优化与编译**
GPU kernel, auto kernel generation

⇅ **多硬件支持**
CPU, GPU, RDMA devices

Imperative programming

命令式编程

- define-by-run or 动态图

执行方式

- 前端语言直接驱动后端算子执行
- 用户表达式会立即被求值

优点
- 方便调试
- 灵活性高

缺点
- 缺少对算法的统一描述
- 缺乏编译期优化

代表 PyTorch

前端 ⇅ 前端编程语言和接口
Python, Lua, R, C++, JavaScript

⬇

统一表示 ⇅ 神经网络模型中间表示
计算图

⬇

优化层 ⇅ 自动微分

计算图优化

⬇

运行时 ⇅ 调度与执行

⬇

底层库 ⇅ 多硬件支持
CPU, GPU, RDMA devices
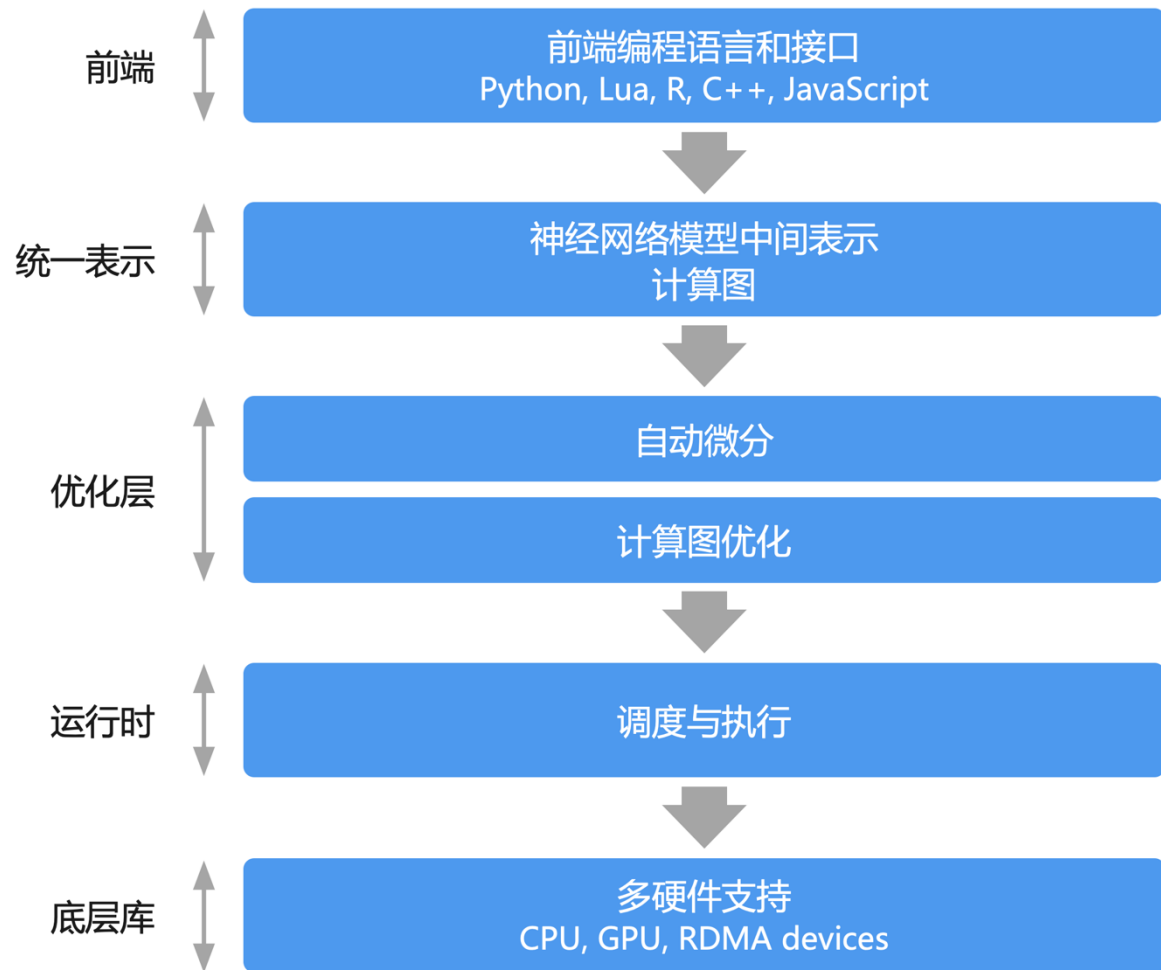
Declarative programming

声明式编程

- define-and-run or 静态图

执行方式

- 前端语言中的表达式不直接执行
- 构建一个完整前向计算过程表示
- 对数据流图经过优化然后再执行

优点
- 执行之前得到全程序描述
  - 运行前编译优化
  - 极致性能优化

缺点
- 数据和控制流限制强
- 不方便调试
- 灵活性低

代表 TensorFlow

前端 → 前端编程语言和接口
Python, Lua, R, C++, JavaScript

统一表示 → 神经网络模型中间表示
计算图

优化层 → 自动微分

计算图优化

运行时 → 内存管理，计算图调度 & 执行

底层库 → 内核代码优化与编译
GPU kernel, auto kernel generation

多硬件支持
CPU, GPU, RDMA devices

www.hiascend.com
www.mindspore.cn
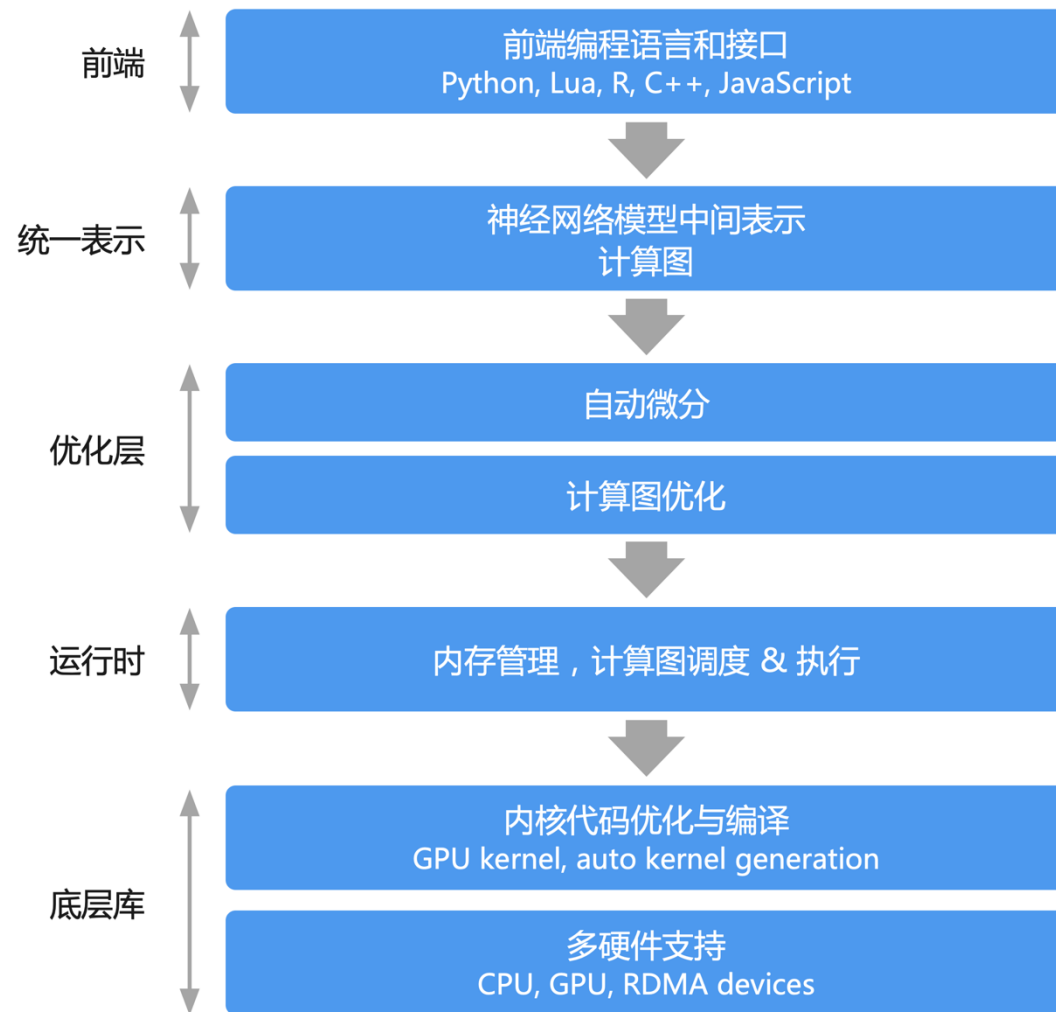
Fusion

- 分阶段编程（Multi-stage）
- 及时编译（Just-in-time, JIT)

代表　　[M]ˢ MindSpore　　⏻ PyTorch JIT　　TF Eager

Summary

1. 回顾了深度学习的编程方式

2. 了解了什么是声明式编程和命令式编程

3. 未来以命令式编程易用性为主，结合声明式编程的优化方式相融合

BUILDING A BETTER CONNECTED WORLD

BUILDING A BETTER CONNECTED WORLD

THANK YOU

www.hiascend.com

www.mindspore.cn