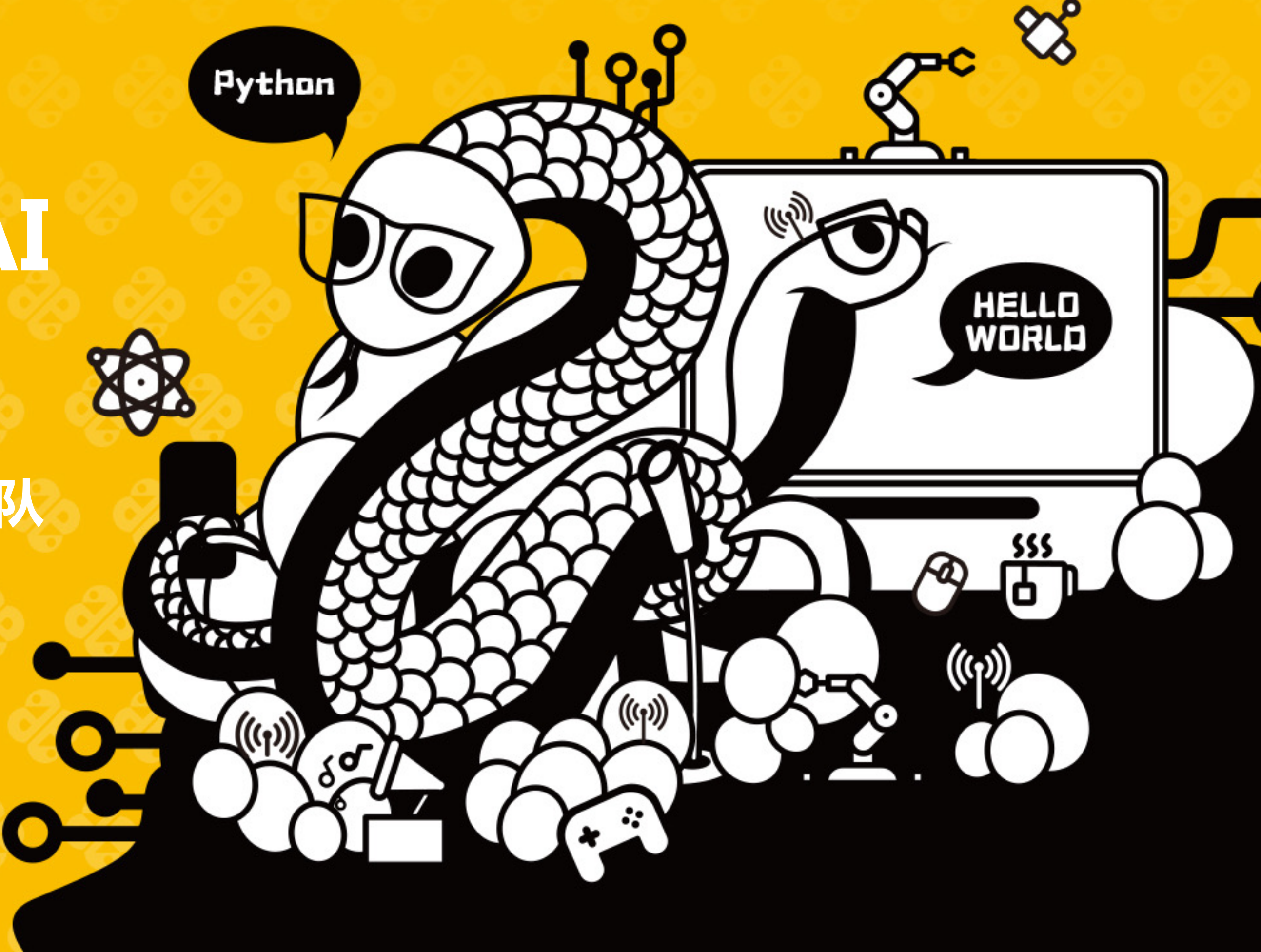
python™ × PYCHINA Python 中国社区

Python for Good

PyCon China 2022

ARM 芯片的 Python + AI 算力优化

主讲人： 朱宏林 – 阿里云程序语言与编译器团队

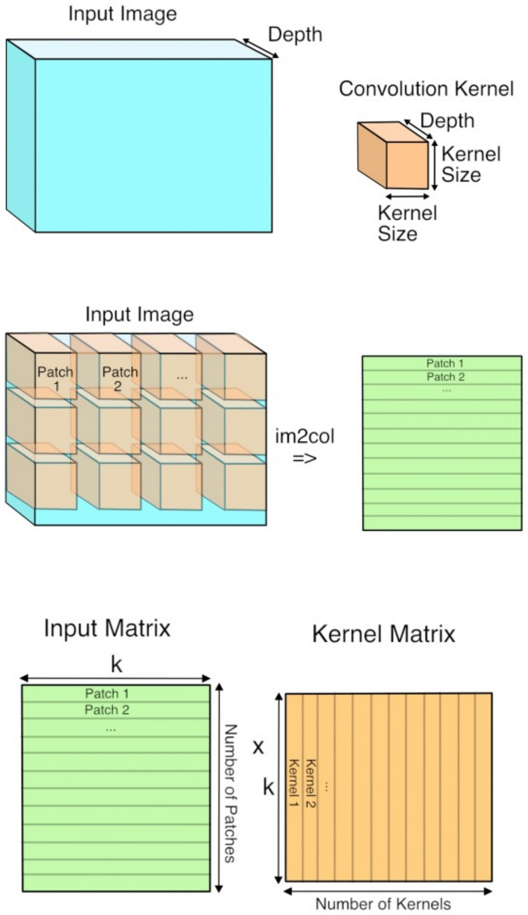


简介

- 当今开发者们大量使用 Python 语言编写的 AI 程序。过去这些程序总跑在 GPU 或者 x86 架构的 CPU 上。然而综合考虑到功耗、成本、性能等因素，云厂商们开始建设 ARM 架构的服务平台，如何整合 Python + AI 的相关软件并使其在该平台上发挥最高的性能成为了工程师们关注的焦点。
- 矩阵乘法是深度学习计算的重要组成部分，我们利用 ARM 架构新提供的矩阵扩展对 bf16 类型的矩阵乘法计算进行优化，该优化将纯矩阵乘法的运算速度提升 3 倍以上，对深度学习推理任务性能提升明显。目前，该成果已经被集成进 OpenBLAS 和 PyTorch 中。
- 本次演讲，将向大家介绍我们在倚天 710 ARM 芯片上开展的 Python + AI 优化工作，以及在 ARM 云平台上部署 Python + AI 任务的最佳实践。

深度学习

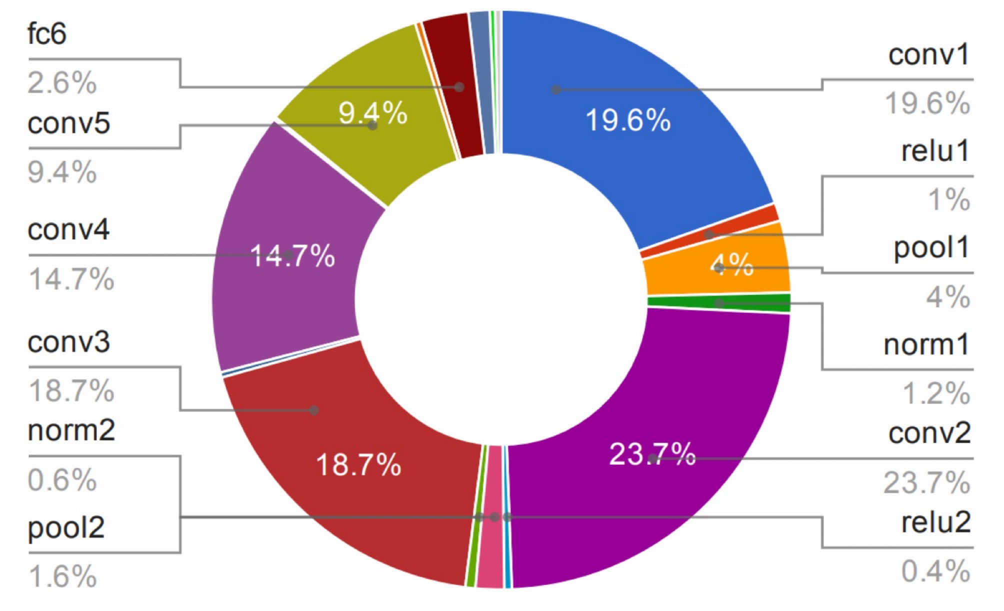
- 广泛使用的深度学习框架
  - TensorFlow、PyTorch
- 结合硬件（ARM 服务端芯片）
  - 倚天 710
  - AWS graviton
- 矩阵乘法
  - 为什么矩阵乘法是深度学习的核心
  - Conv、Linear、Transformers



来源: *Why GEMM is at the heart of deep learning, Pete Warden*

Convolution

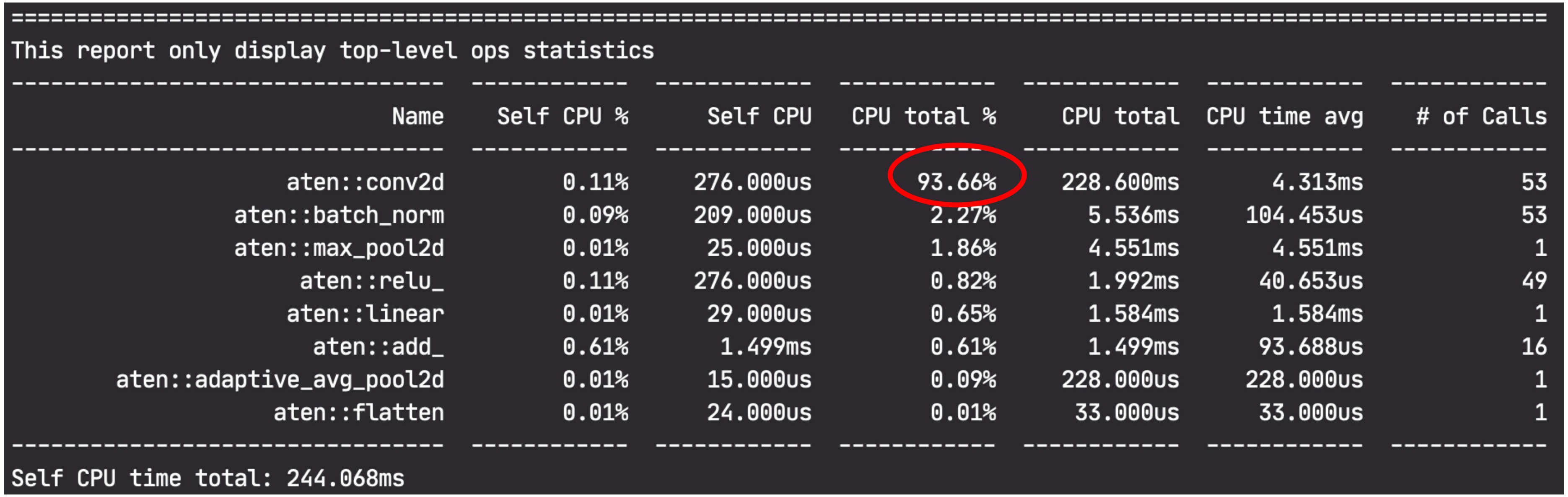
- AlexNet 模型推理各个层计算比例
  - 86.1%
  - 2.6%



来源: *Learning Semantic Image Representations at a Large Scale, Yangqing Jia*

Convolution

- ResNet-50
- PyTorch Profiler



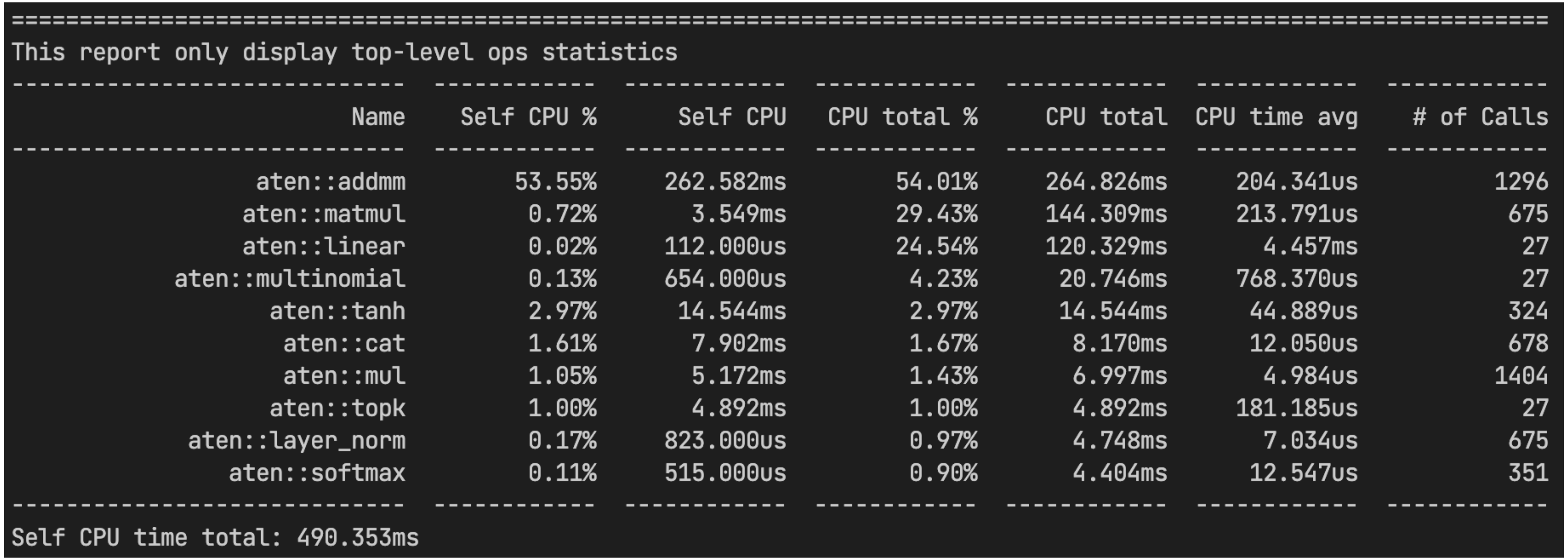
```
===============================================================================================================================
This report only display top-level ops statistics
----------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                              Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
----------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                      aten::conv2d         0.11%     276.000us        93.66%     228.600ms       4.313ms            53
                  aten::batch_norm         0.09%     209.000us         2.27%       5.536ms     104.453us            53
                  aten::max_pool2d         0.01%      25.000us         1.86%       4.551ms       4.551ms             1
                       aten::relu_         0.11%     276.000us         0.82%       1.992ms      40.653us            49
                      aten::linear         0.01%      29.000us         0.65%       1.584ms       1.584ms             1
                        aten::add_         0.61%       1.499ms         0.61%       1.499ms      93.688us            16
         aten::adaptive_avg_pool2d         0.01%      15.000us         0.09%     228.000us     228.000us             1
                     aten::flatten         0.01%      24.000us         0.01%      33.000us      33.000us             1
----------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 244.068ms
```

Transformers

Python for Good

PyCon China 2022

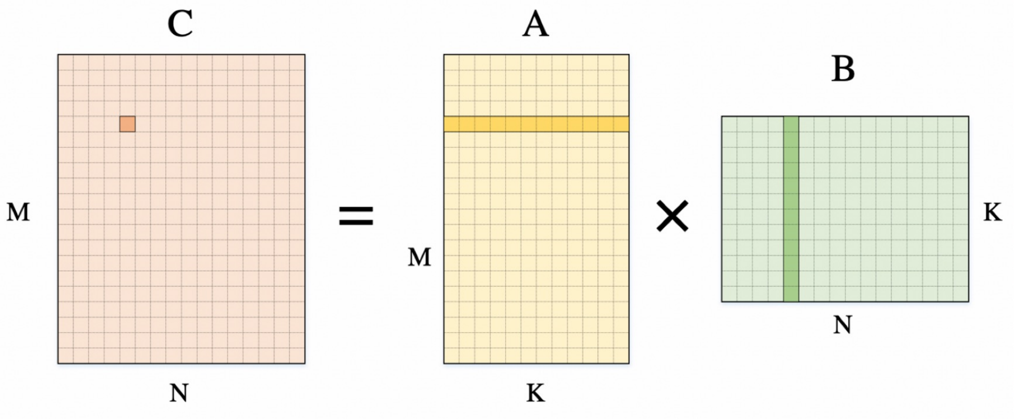
- GPT-2
- PyTorch Profiler

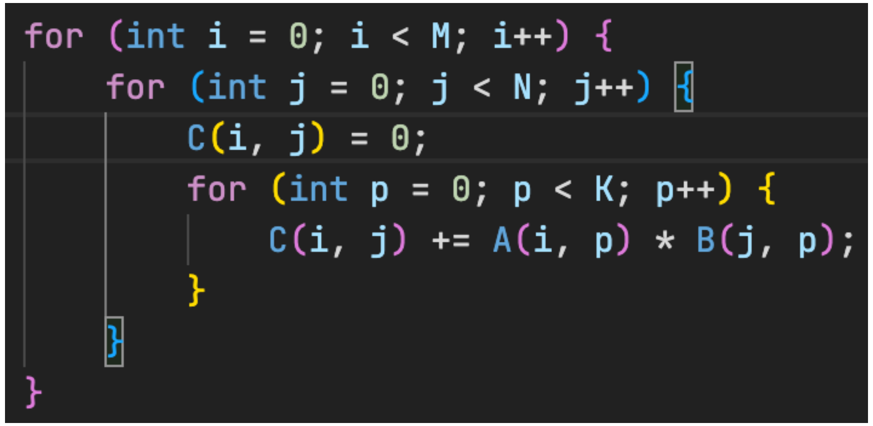


```
=================================================================================================================
This report only display top-level ops statistics
-----------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                         Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
-----------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                  aten::addmm        53.55%     262.582ms        54.01%     264.826ms     204.341us          1296
                 aten::matmul         0.72%       3.549ms        29.43%     144.309ms     213.791us           675
                 aten::linear         0.02%     112.000us        24.54%     120.329ms       4.457ms            27
            aten::multinomial         0.13%     654.000us         4.23%      20.746ms     768.370us            27
                   aten::tanh         2.97%      14.544ms         2.97%      14.544ms      44.889us           324
                    aten::cat         1.61%       7.902ms         1.67%       8.170ms      12.050us           678
                    aten::mul         1.05%       5.172ms         1.43%       6.997ms       4.984us          1404
                   aten::topk         1.00%       4.892ms         1.00%       4.892ms     181.185us            27
             aten::layer_norm         0.17%     823.000us         0.97%       4.748ms       7.034us           675
                aten::softmax         0.11%     515.000us         0.90%       4.404ms      12.547us           351
-----------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 490.353ms
```

GEMM

- General Matrix Multiplication: C = αAB + βC
  - MNK 级别的时间复杂度
  - 深度学习卷积运算量很大(例如尺寸256 x 1152 x 192 ≈ 5700万)
  - GEMM 通过优化内存局部性和向量指令，比朴素实现快 10 倍以上

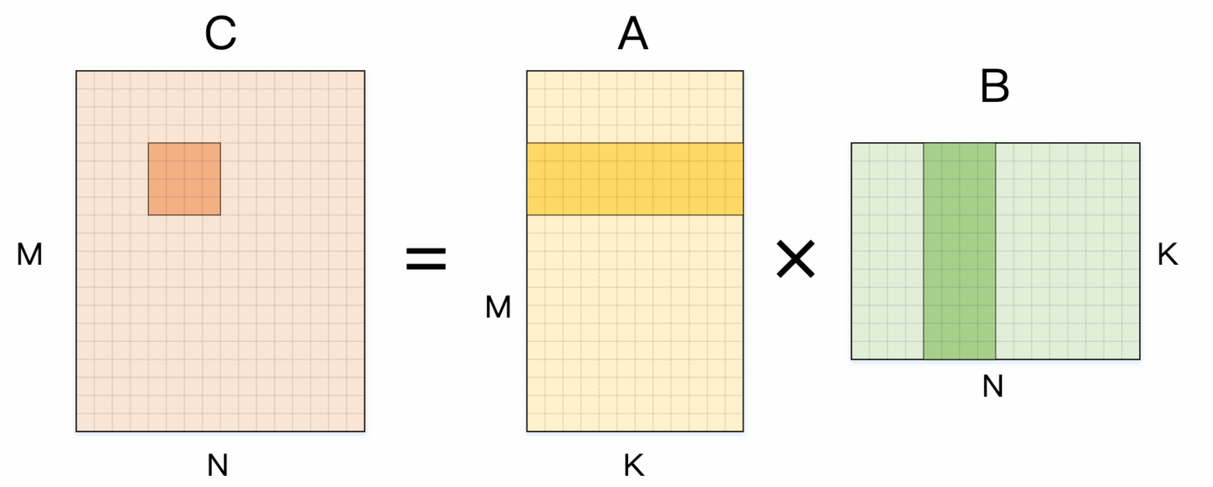


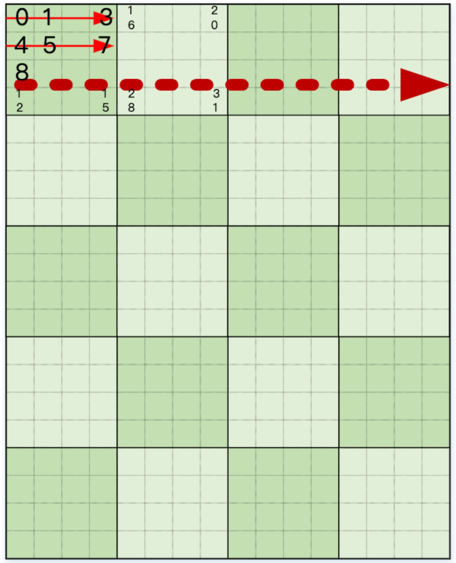


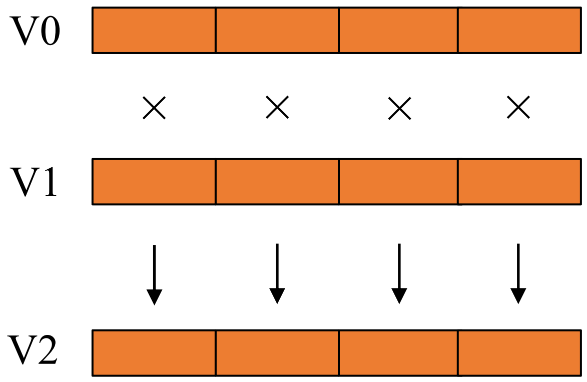
```
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        C(i, j) = 0;
        for (int p = 0; p < K; p++) {
            C(i, j) += A(i, p) * B(j, p);
        }
    }
}
```

GEMM

- 优化 GEMM
  - 内存布局：矩阵分块；重排
  - 向量化指令：AVX、NEON

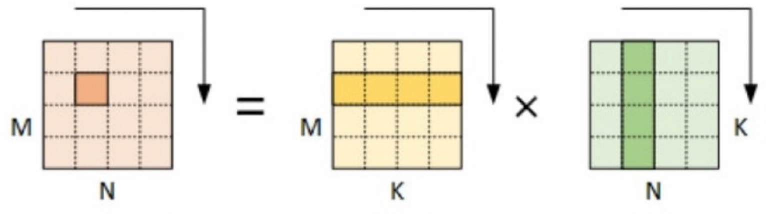


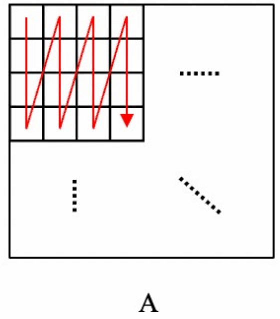


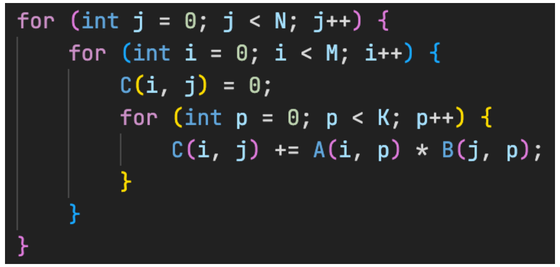


GEMM 例子

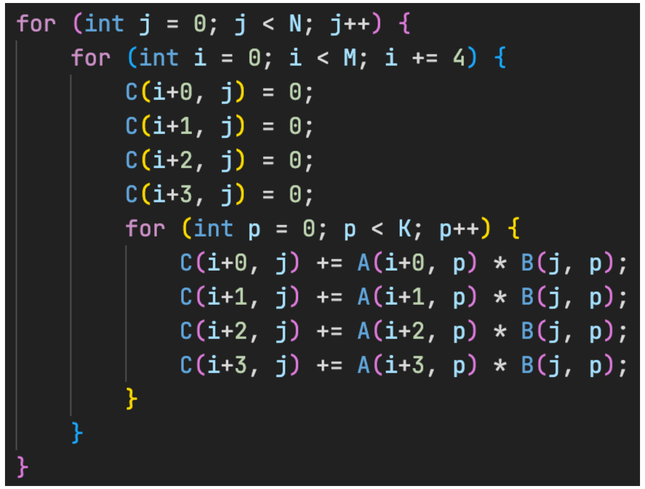
- 优化 GEMM
  - 内存布局：矩阵分块；重排
  - 向量化指令：AVX、NEON



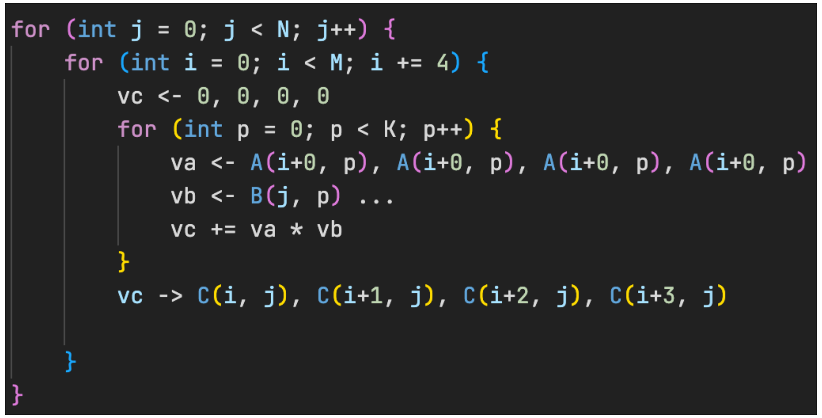




```
for (int j = 0; j < N; j++) {
    for (int i = 0; i < M; i++) {
        C(i, j) = 0;
        for (int p = 0; p < K; p++) {
            C(i, j) += A(i, p) * B(j, p);
        }
    }
}
```



```
for (int j = 0; j < N; j++) {
    for (int i = 0; i < M; i += 4) {
        C(i+0, j) = 0;
        C(i+1, j) = 0;
        C(i+2, j) = 0;
        C(i+3, j) = 0;
        for (int p = 0; p < K; p++) {
            C(i+0, j) += A(i+0, p) * B(j, p);
            C(i+1, j) += A(i+1, p) * B(j, p);
            C(i+2, j) += A(i+2, p) * B(j, p);
            C(i+3, j) += A(i+3, p) * B(j, p);
        }
    }
}
```



```
for (int j = 0; j < N; j++) {
    for (int i = 0; i < M; i += 4) {
        vc <- 0, 0, 0, 0
        for (int p = 0; p < K; p++) {
            va <- A(i+0, p), A(i+0, p), A(i+0, p), A(i+0, p)
            vb <- B(j, p) ...
            vc += va * vb
        }
        vc -> C(i, j), C(i+1, j), C(i+2, j), C(i+3, j)

    }
}
```

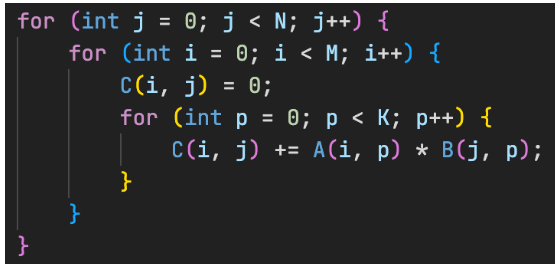
原始算法

展开4x1

向量化

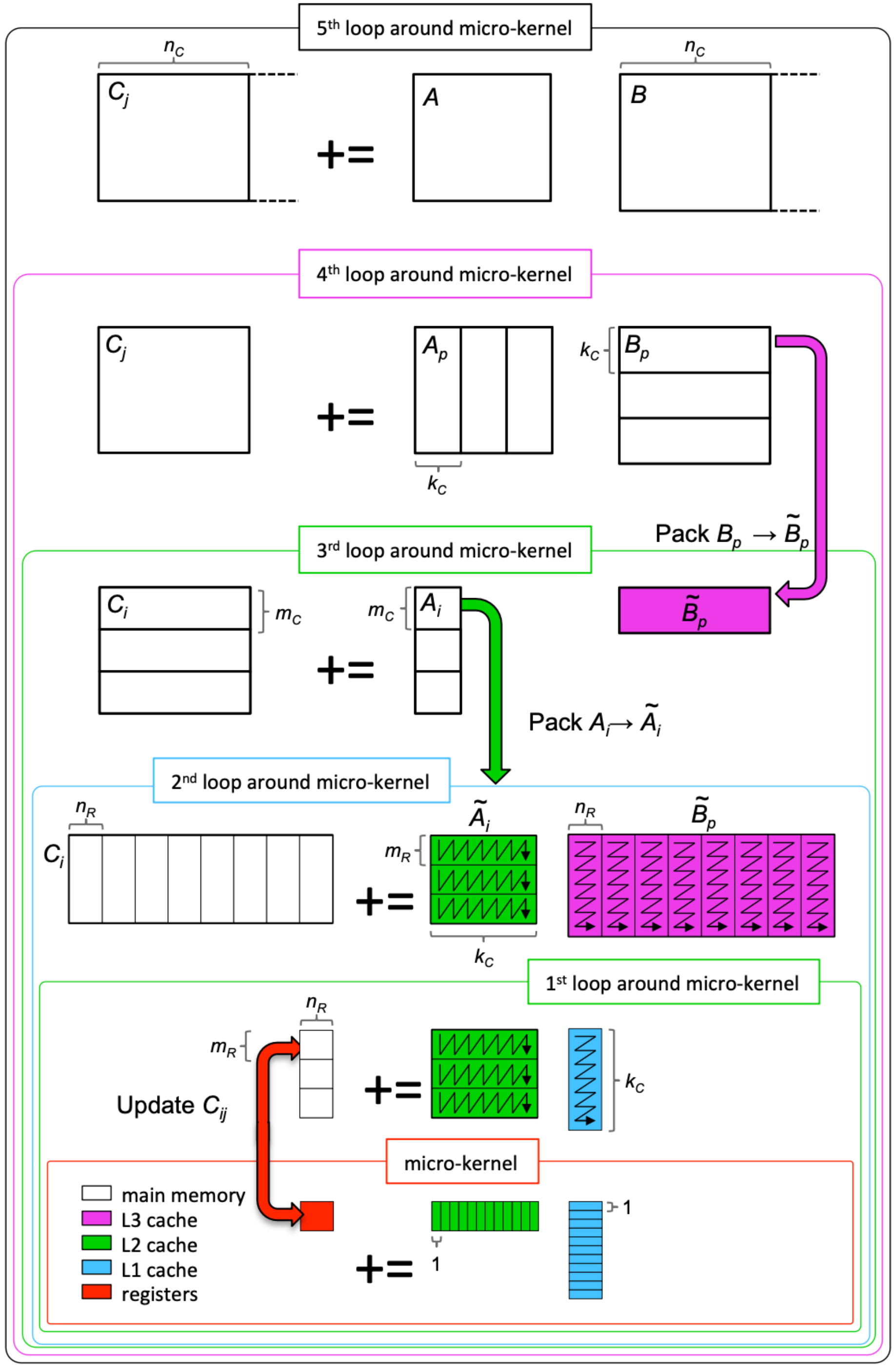
GEMM 例子

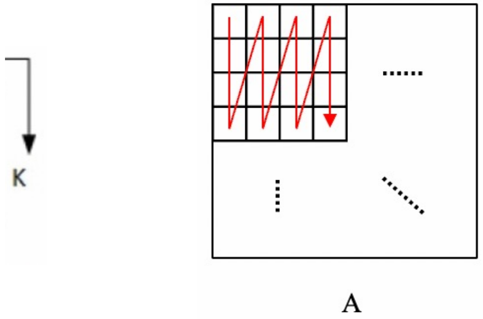
- 优化 GEMM
  - 内存布局：矩阵分块；重
  - 向量化指令：AVX、NEO

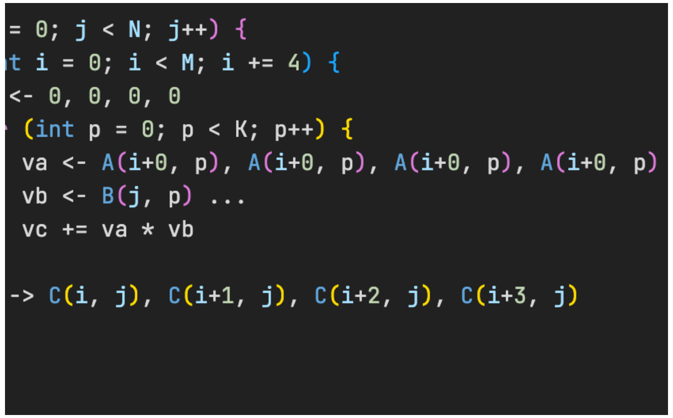


```
for (int j = 0; j < N; j++) {
    for (int i = 0; i < M; i++) {
        C(i, j) = 0;
        for (int p = 0; p < K; p++) {
            C(i, j) += A(i, p) * B(j, p);
        }
    }
}
```

原始算法





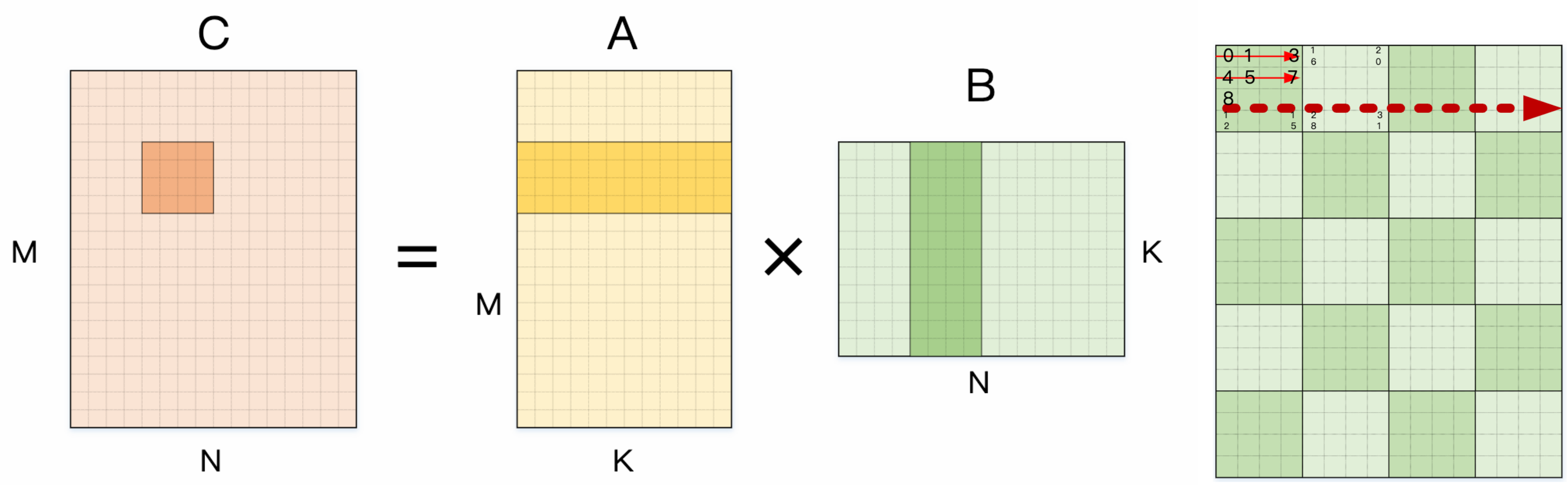


```
= 0; j < N; j++) {
t i = 0; i < M; i += 4) {
<- 0, 0, 0, 0
 (int p = 0; p < K; p++) {
 va <- A(i+0, p), A(i+0, p), A(i+0, p), A(i+0, p)
 vb <- B(j, p) ...
 vc += va * vb

-> C(i, j), C(i+1, j), C(i+2, j), C(i+3, j)
```

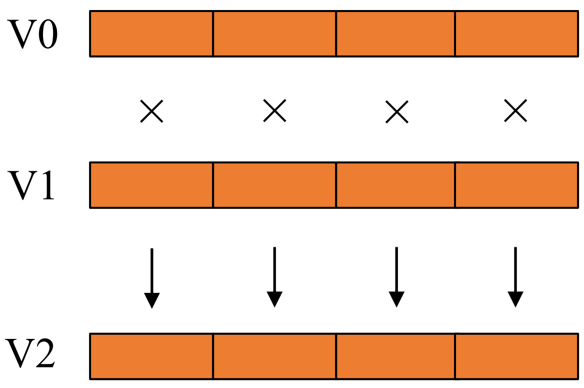
向量化

GEMM

- 优化 GEMM
  - 内存布局：矩阵分块；重排
  - 向量化指令：AVX、NEON

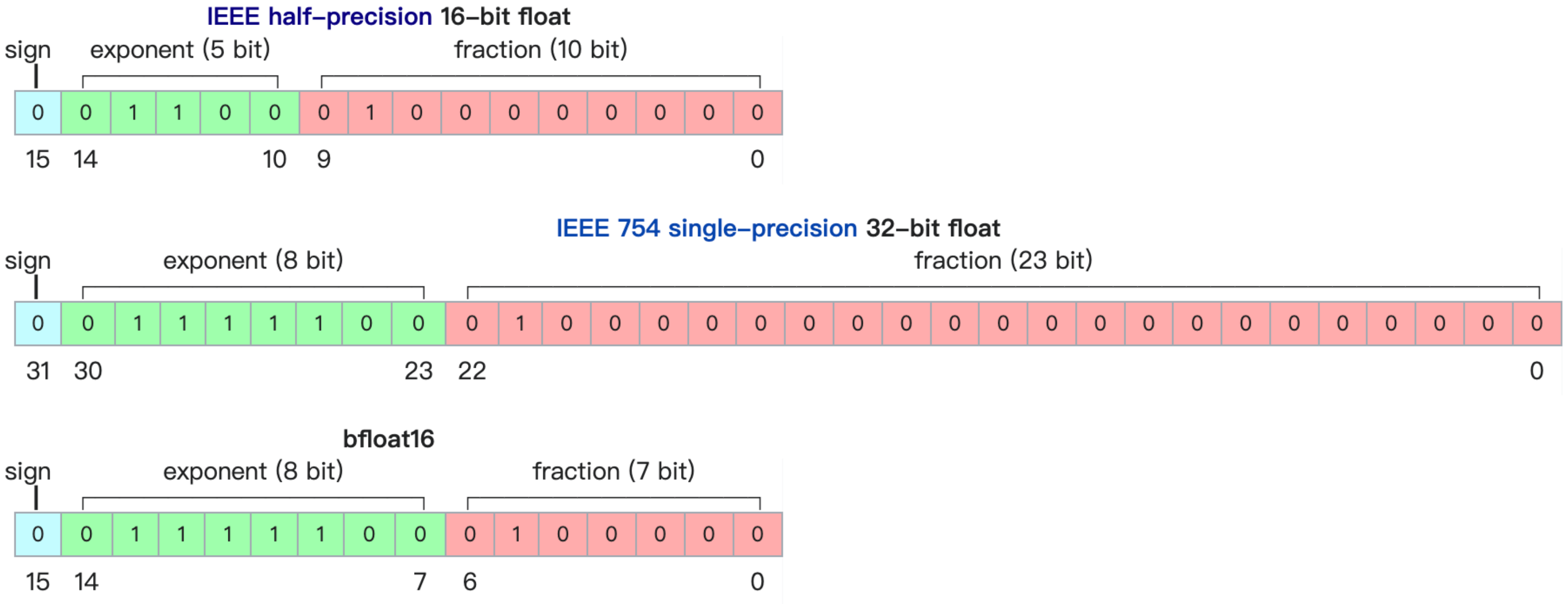


- 硬件加速
  - Nvidia Volta 架构引入 tensor core
  - Intel AMX, Advanced Matrix Extension
  - ARM SME, Scalable Matrix Extension
  - CPU 存在优势场景，但当前尚没有可大规模使用 AMX 和 SME 实例



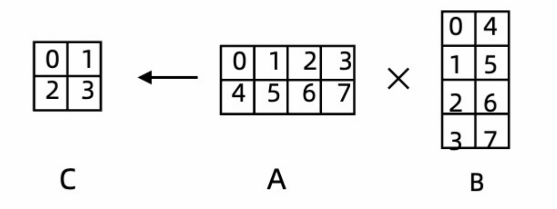
BF16 数据类型

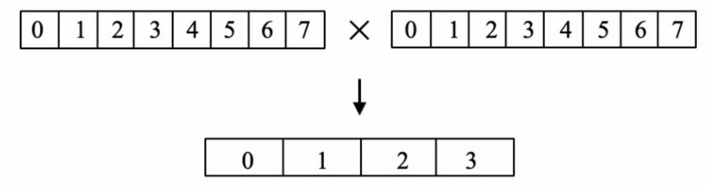
- BF16（Brain Floating Point，bfloat16）
  - Google Brain 团队
  - float32、float16、bfloat16（FP32、FP16、BF16）
- 特点
  - 表示范围和 FP32 一致
  - 转换便利
  - 节省存储空间
  - 硬件指令支持

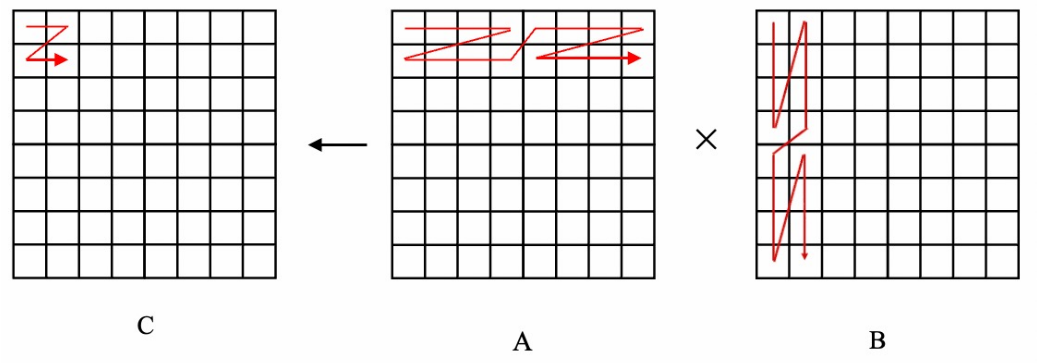


ARMv8.6 bf16 扩展

- bf16 扩展
  - ARMv8.6
  - 矩阵乘法指令 BFMMLA
  - 类型转换指令 BFCVT
- BFMMLA
  - 128 bit 向量寄存器
  - 单指令完成 (2x4) * (4x2)
  - 16 mul + 16 add

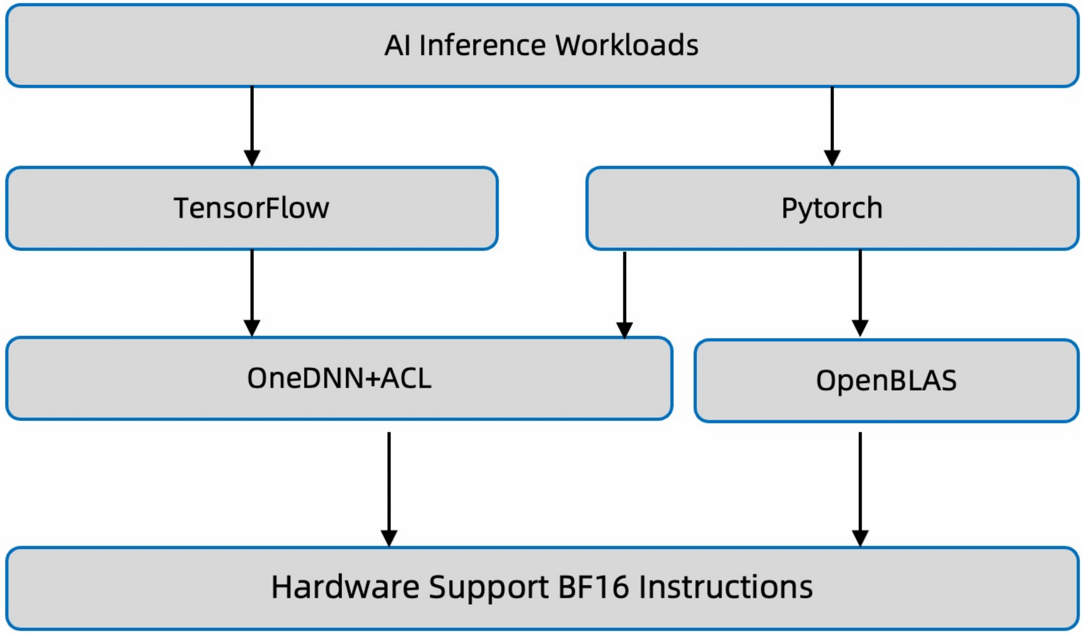






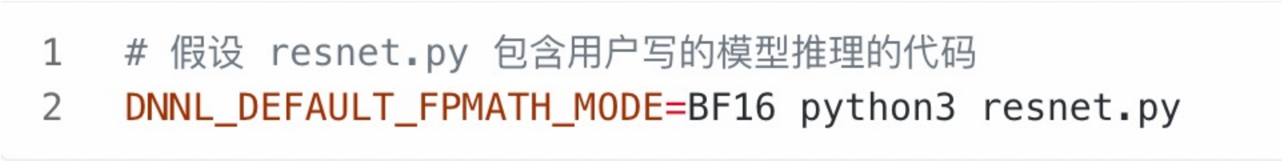
深度学习推理加速

- BF16 gemm 实现
  - ARM Compute Library
  - OpenBLAS
- TensorFlow
  - oneDNN + ACL
  - `DNNL_DEFAULT_FPMATH_MODE=BF16`
- PyTorch
  - OpenBLAS
  - oneDNN + ACL
  - `torch.set_float32_fast_math_mode("BF16")`

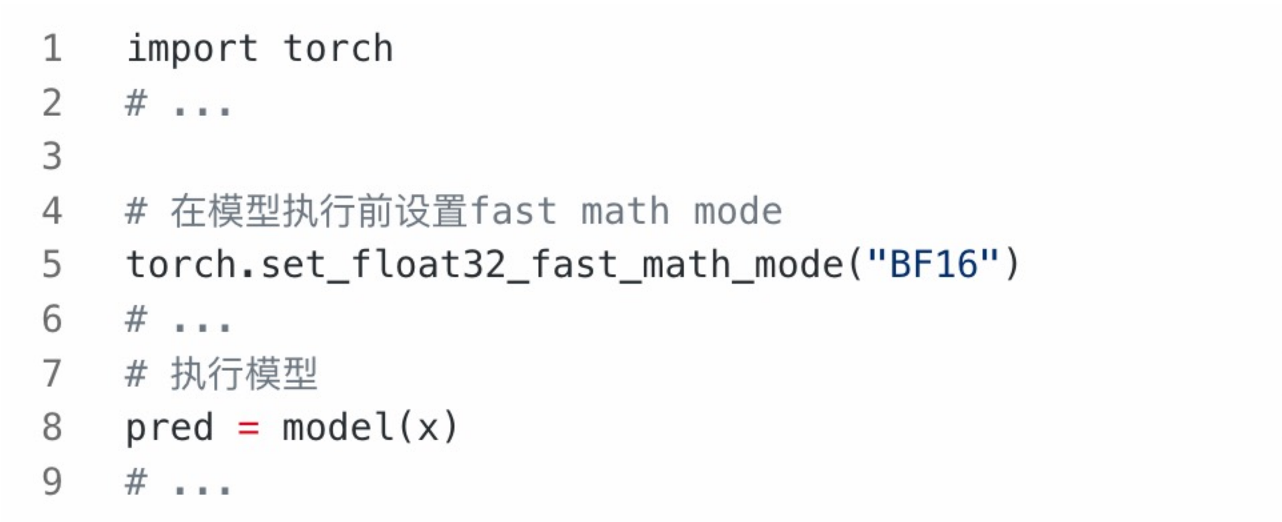


深度学习推理加速

- BF16 gemm 实现
  - ARM Compute Library
  - OpenBLAS
- TensorFlow
  - oneDNN + ACL
  - DNNL_DEFAULT_FPMATH_MODE=BF16
- PyTorch
  - OpenBLAS
  - oneDNN + ACL
  - torch.set_float32_fast_math_mode("BF16")



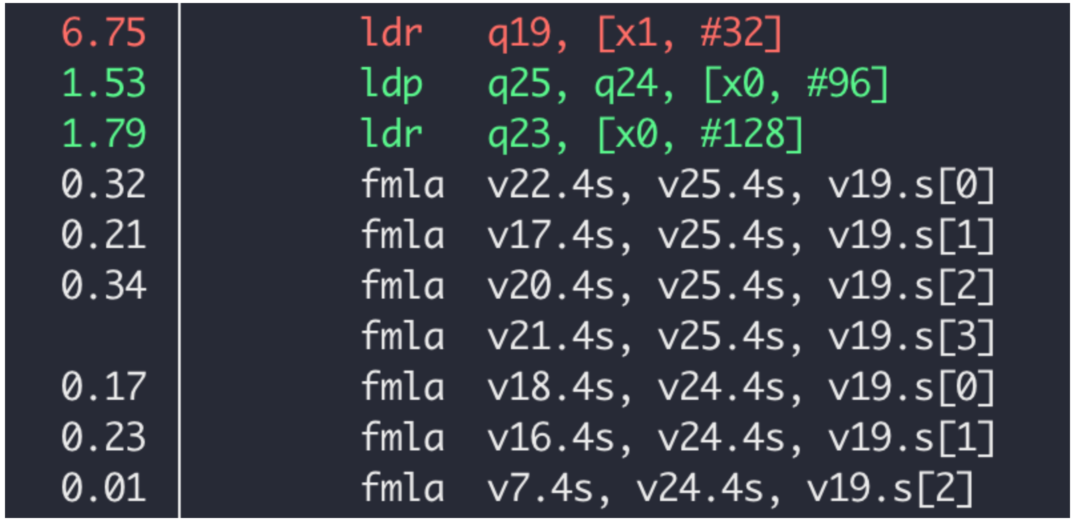
```
# 假设 resnet.py 包含用户写的模型推理的代码
DNNL_DEFAULT_FPMATH_MODE=BF16 python3 resnet.py
```



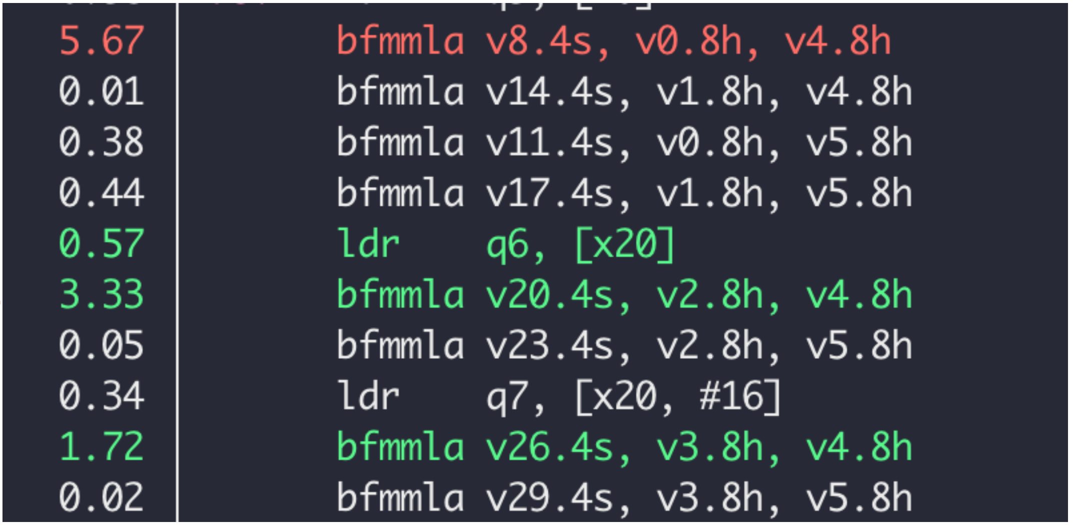
```
import torch
# ...

# 在模型执行前设置fast math mode
torch.set_float32_fast_math_mode("BF16")
# ...
# 执行模型
pred = model(x)
# ...
```

深度学习推理加速

- BF16 gemm 实现
  - ARM Compute Library
  - OpenBLAS
- TensorFlow
  - oneDNN + ACL
  - DNNL_DEFAULT_FPMATH_MODE=BF16
- PyTorch
  - OpenBLAS
  - oneDNN + ACL
  - torch.set_float32_fast_math_mode("BF1



```
6.75 │          ldr    q19, [x1, #32]
1.53 │          ldp    q25, q24, [x0, #96]
1.79 │          ldr    q23, [x0, #128]
0.32 │          fmla   v22.4s, v25.4s, v19.s[0]
0.21 │          fmla   v17.4s, v25.4s, v19.s[1]
0.34 │          fmla   v20.4s, v25.4s, v19.s[2]
     │          fmla   v21.4s, v25.4s, v19.s[3]
0.17 │          fmla   v18.4s, v24.4s, v19.s[0]
0.23 │          fmla   v16.4s, v24.4s, v19.s[1]
0.01 │          fmla   v7.4s, v24.4s, v19.s[2]
```

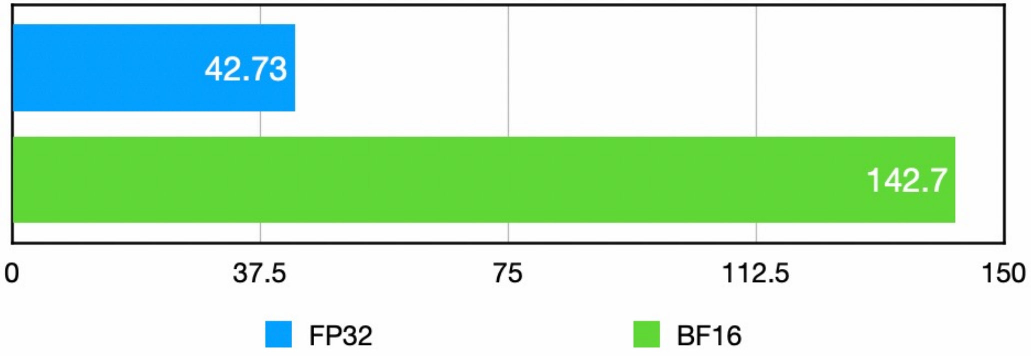


```
5.67 │         bfmmla v8.4s, v0.8h, v4.8h
0.01 │         bfmmla v14.4s, v1.8h, v4.8h
0.38 │         bfmmla v11.4s, v0.8h, v5.8h
0.44 │         bfmmla v17.4s, v1.8h, v5.8h
0.57 │         ldr    q6, [x20]
3.33 │         bfmmla v20.4s, v2.8h, v4.8h
0.05 │         bfmmla v23.4s, v2.8h, v5.8h
0.34 │         ldr    q7, [x20, #16]
1.72 │         bfmmla v26.4s, v3.8h, v4.8h
0.02 │         bfmmla v29.4s, v3.8h, v5.8h
```

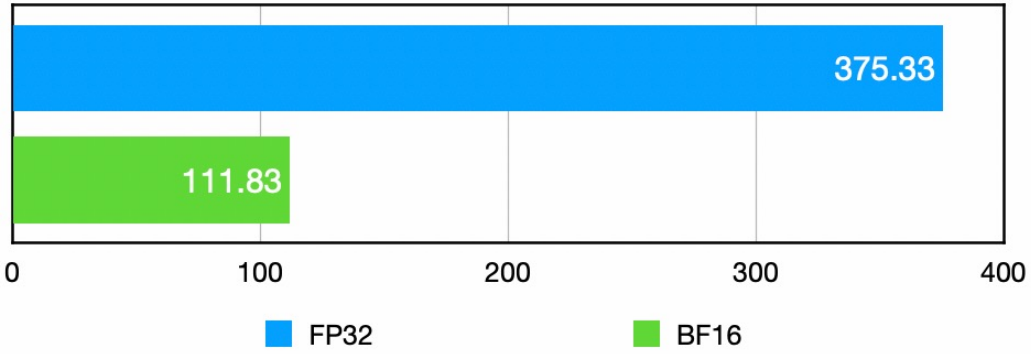
性能测试

- OpenBLAS 矩阵乘法测试
    - 阿里云 ECS g8y（倚天 710）单线程 FP32 vs BF16
    - MNK：2000, 2000, 2000
- GFLOPS
    - 每秒浮点数计算次数，G（十亿次）
    - 2MNK / Times

FP32 和 BF16 GEMM GFLOPS 对比

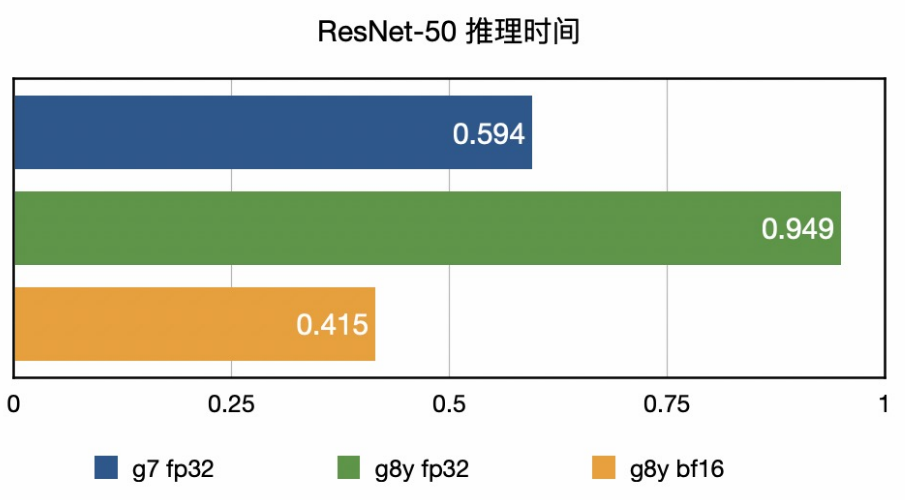


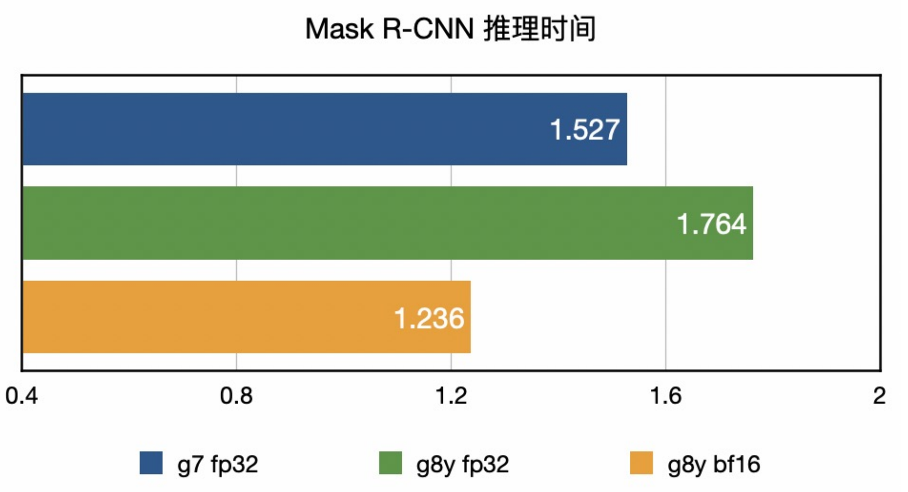
FP32 和 BF16 GEMM 耗时对比（单位毫秒）



性能测试

- 平台
  - 阿里云 ECS g8y vs g7（Ice Lake）
  - 8vCPUs
- TensorFlow 推理测试
  - ResNet-50
  - batch size 32
- PyTorch 推理测试
  - Mask R-CNN
  - 每张图像耗时





最佳实践

- TensorFlow
  - Official（latest）
    - pip install tensorflow == 2.10.1 或者 == 2.11.0
  - 阿里云
    - Docker 镜像
    - accc-registry.cn-hangzhou.cr.aliyuncs.com/tensorflow/tensorflow
    - Tag: latest
- PyTorch
  - Official（latest）
    - pip install torch==1.13.0
  - 阿里云
    - Docker 镜像
    - accc-registry.cn-hangzhou.cr.aliyuncs.com/pytorch/pytorch
    - Tag: torch_openblas, torch_openblas_modelzoo

PyTorch BF16 加速演示

- 拉取镜像

    -
      ```
      docker pull accc-registry.cn-hangzhou.cr.aliyuncs.com/pytorch/pytorch:torch1.13.0_openblas_modelzoo
      ```

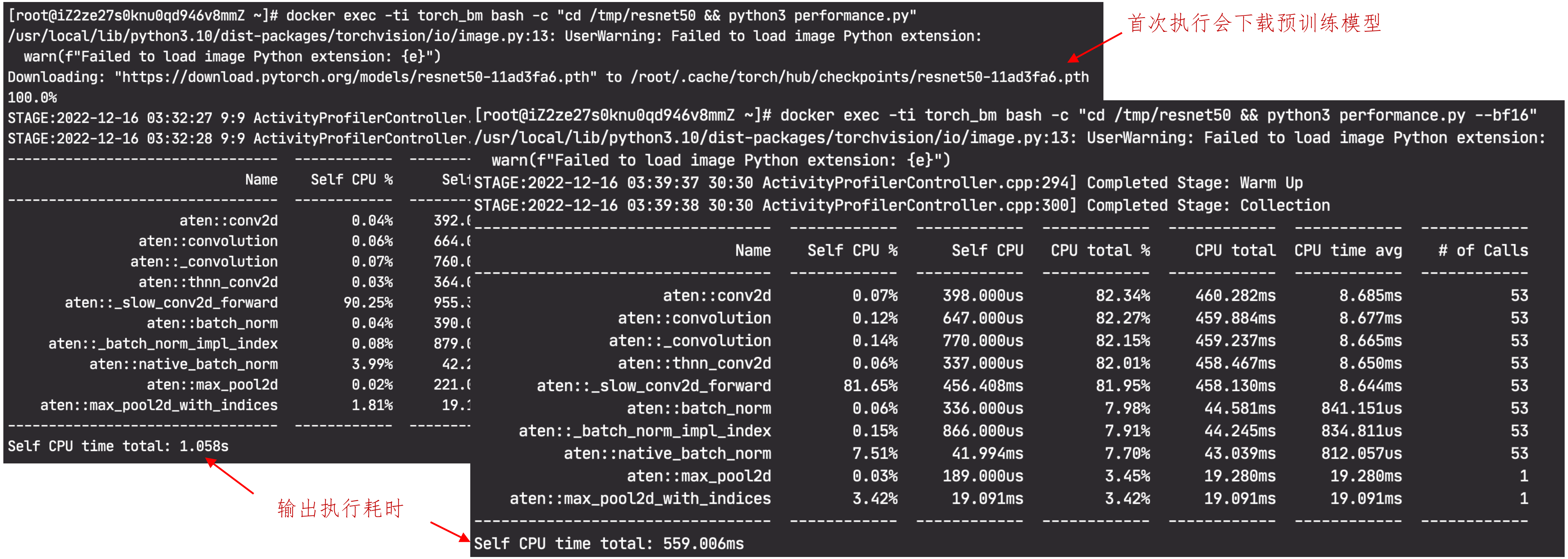
- 启动容器

    -
      ```
      docker run -d --name torch_bm -ti accc-registry.cn-hangzhou.cr.aliyuncs.com/pytorch/pytorch:torch1.13.0_openblas_modelzoo
      ```

PyTorch BF16 加速演示

Python for Good
»» PyCon China 2022

- 测试 ResNet-50 推理性能
  - docker exec -ti torch_bm bash -c "cd /tmp/resnet50 && python3 performance.py”
  - docker exec -ti torch_bm bash -c "cd /tmp/resnet50 && python3 performance.py --bf16"



```
[root@iZ2ze27s0knu0qd946v8mmZ ~]# docker exec -ti torch_bm bash -c "cd /tmp/resnet50 && python3 performance.py"
/usr/local/lib/python3.10/dist-packages/torchvision/io/image.py:13: UserWarning: Failed to load image Python extension:
  warn(f"Failed to load image Python extension: {e}")
Downloading: "https://download.pytorch.org/models/resnet50-11ad3fa6.pth" to /root/.cache/torch/hub/checkpoints/resnet50-11ad3fa6.pth
100.0%
STAGE:2022-12-16 03:32:27 9:9 ActivityProfilerController.
STAGE:2022-12-16 03:32:28 9:9 ActivityProfilerController.
-------------------------------  ------------  --------
                           Name    Self CPU %      Self
-------------------------------  ------------  --------
                   aten::conv2d         0.04%     392.0
              aten::convolution         0.06%     664.0
             aten::_convolution         0.07%     760.0
              aten::thnn_conv2d         0.03%     364.0
      aten::_slow_conv2d_forward        90.25%     955.3
               aten::batch_norm         0.04%     390.0
   aten::_batch_norm_impl_index         0.08%     879.0
        aten::native_batch_norm         3.99%      42.2
               aten::max_pool2d         0.02%     221.0
  aten::max_pool2d_with_indices         1.81%      19.1
-------------------------------  ------------  --------
Self CPU time total: 1.058s
```

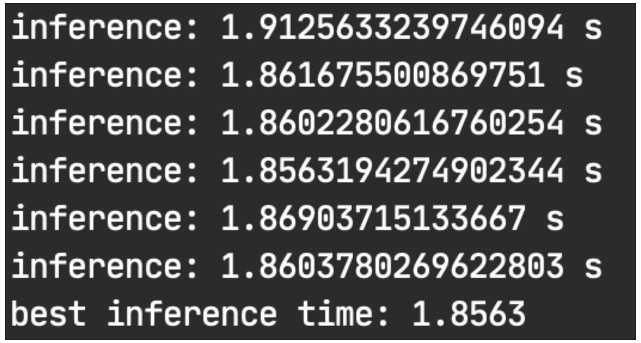
首次执行会下载预训练模型

```
[root@iZ2ze27s0knu0qd946v8mmZ ~]# docker exec -ti torch_bm bash -c "cd /tmp/resnet50 && python3 performance.py --bf16"
/usr/local/lib/python3.10/dist-packages/torchvision/io/image.py:13: UserWarning: Failed to load image Python extension:
  warn(f"Failed to load image Python extension: {e}")
STAGE:2022-12-16 03:39:37 30:30 ActivityProfilerController.cpp:294] Completed Stage: Warm Up
STAGE:2022-12-16 03:39:38 30:30 ActivityProfilerController.cpp:300] Completed Stage: Collection
-------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                           Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
-------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                   aten::conv2d         0.07%     398.000us        82.34%     460.282ms       8.685ms            53
              aten::convolution         0.12%     647.000us        82.27%     459.884ms       8.677ms            53
             aten::_convolution         0.14%     770.000us        82.15%     459.237ms       8.665ms            53
              aten::thnn_conv2d         0.06%     337.000us        82.01%     458.467ms       8.650ms            53
     aten::_slow_conv2d_forward        81.65%     456.408ms        81.95%     458.130ms       8.644ms            53
               aten::batch_norm         0.06%     336.000us         7.98%      44.581ms     841.151us            53
   aten::_batch_norm_impl_index         0.15%     866.000us         7.91%      44.245ms     834.811us            53
        aten::native_batch_norm         7.51%      41.994ms         7.70%      43.039ms     812.057us            53
               aten::max_pool2d         0.03%     189.000us         3.45%      19.280ms      19.280ms             1
  aten::max_pool2d_with_indices         3.42%      19.091ms         3.42%      19.091ms      19.091ms             1
-------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 559.006ms
```

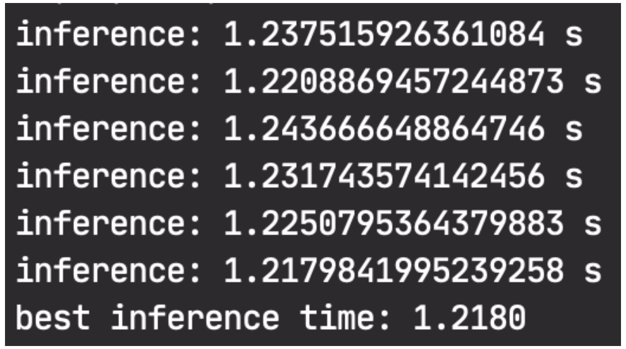
输出执行耗时

PyTorch BF16 加速演示

- 测试 Mask R-CNN 推理性能
  - `docker exec -ti torch_bm bash -c "cd /tmp/maskrcnn && python3 performance.py”`
  - `docker exec -ti torch_bm bash -c "cd /tmp/maskrcnn && python3 performance.py --bf16"`



```
inference: 1.9125633239746094 s
inference: 1.861675500869751 s
inference: 1.8602280616760254 s
inference: 1.8563194274902344 s
inference: 1.86903715133667 s
inference: 1.8603780269622803 s
best inference time: 1.8563
```

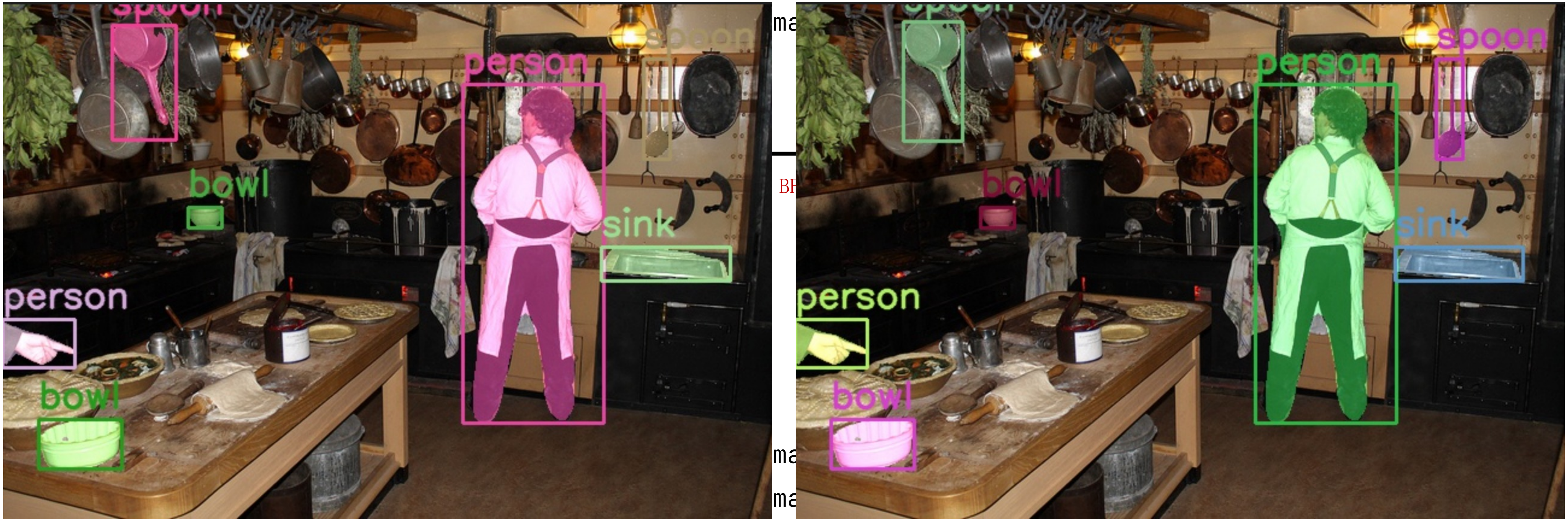


```
inference: 1.237515926361084 s
inference: 1.2208869457244873 s
inference: 1.243666648864746 s
inference: 1.231743574142456 s
inference: 1.2250795364379883 s
inference: 1.2179841995239258 s
best inference time: 1.2180
```

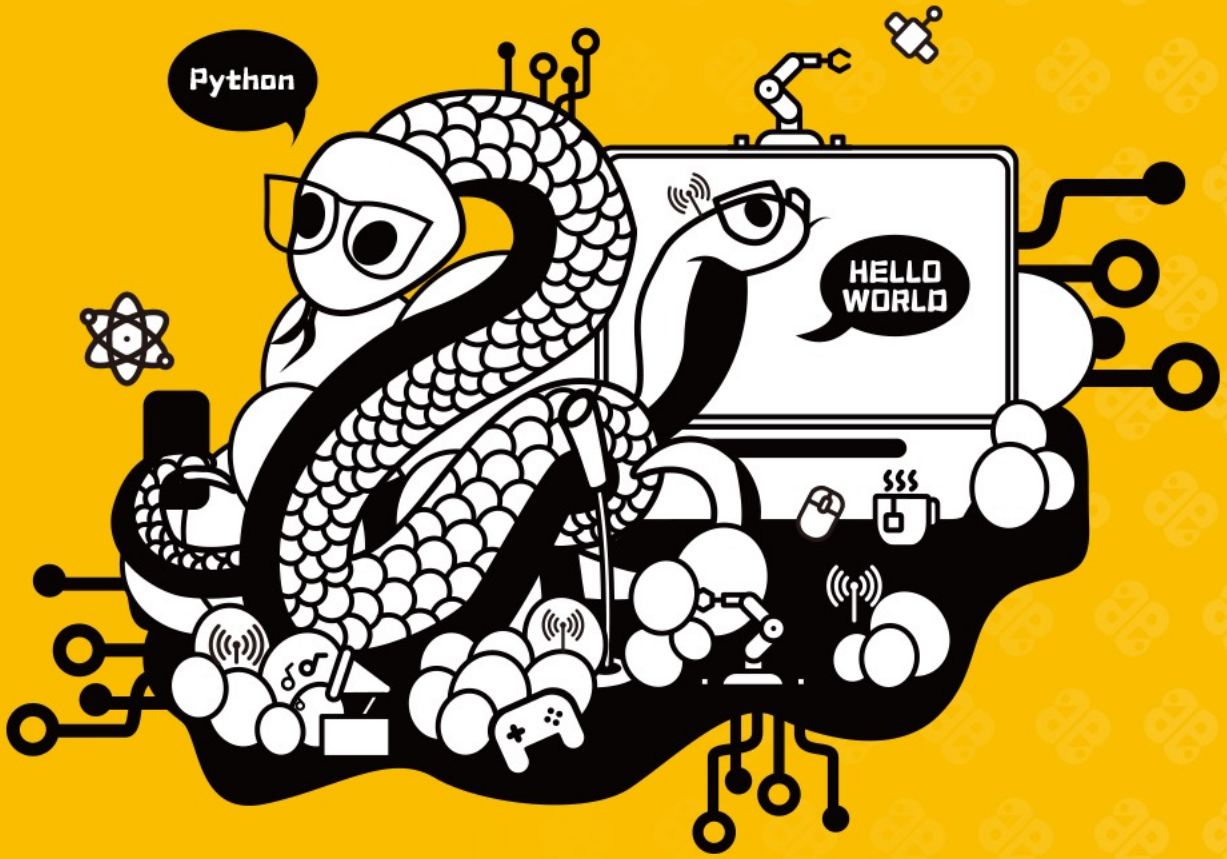
- 验证 Mask R-CNN 预测结果
  - `docker exec -ti torch_bm bash -c "cd /tmp/maskrcnn && python3 validate.py”`
  - `docker exec -ti torch_bm bash -c "cd /tmp/maskrcnn && python3 validate.py --bf16"`

PyTorch BF16 加速演示

- 测试 Mask R-CNN 推理性能
  - `docker exec -ti torch_bm bash -c "cd /tmp/maskrcnn && python3 performance.py”`



预测结果，左图FP32，右图BF16，提升性能的同时不影响模型预测结果



Thanks!

感谢观看