OpenPPL Arm Server 技术分享

商汤 AI 推理引擎 OpenPPL 实战训练营

邱君仪 2022.03.03

主讲人介绍

邱君仪

商汤科技高性能计算部门异构计算工程师

- 上海交通大学本科毕业，获 UC San Diego 硕士学位
- 曾任职于 Intel, Falcon Computing 和 Google
- 目前在商汤科技负责高性能推理引擎 PPL ARM Server 架构的相关研发

课程提纲

OpenPPL ARM Server 是针对高性能 ARM 服务器优化的深度学习推理引擎。本次分享将结合 ARM 服务器的架构特点，以基于鲲鹏 KP920 芯片的研发和优化为例，对推理引擎的核心卷积技术的算法设计及性能优化进行介绍。

1. OpenPPL ARM Server 简介
2. ARMv8 及 Neon(Advanced SIMD) 指令集
3. ARM 服务器微架构特点
4. OpenPPL ARM Server 卷积设计解析
5. OpenPPL ARM Server 综合性能展示

实战训练营

Outline

Outline

OpenPPL Arm Server 简介

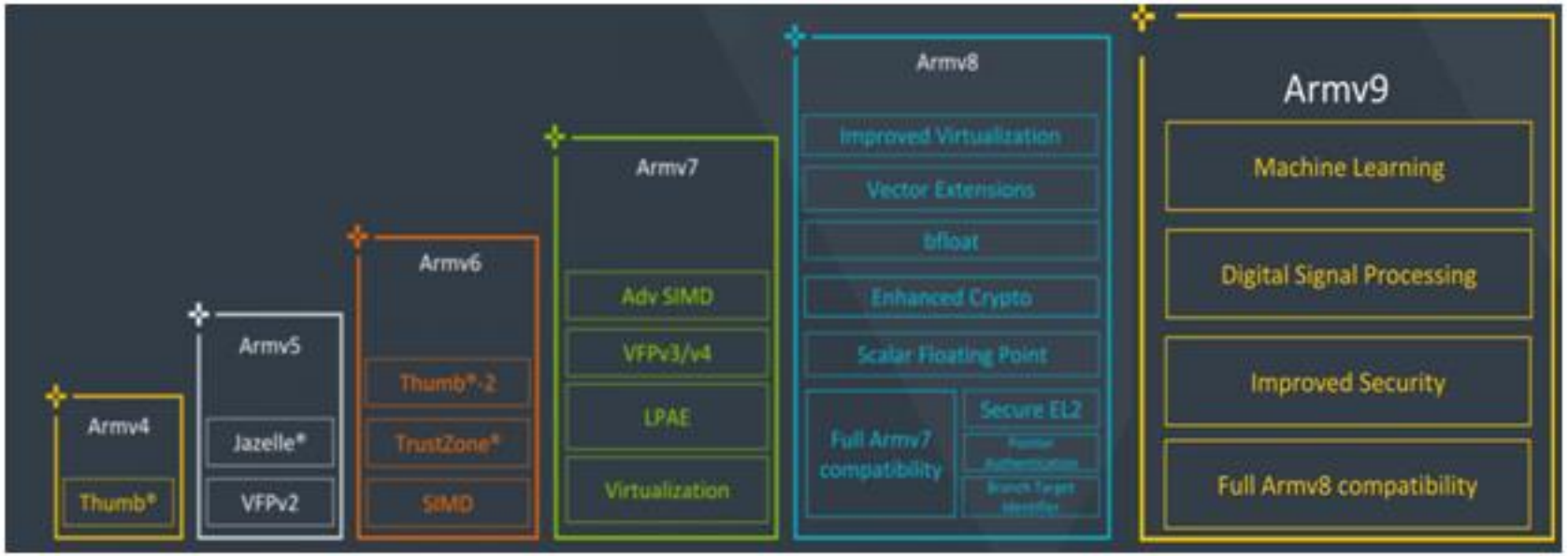
OpenPPL Arm Server

- 针对高性能ARM架构服务器处理器优化的深度学习推理引擎
- 目前支持 FP32 及 FP16 两种浮点数精度格式
- 支持常见的分类网络以及部分超分、分割网络推理
- 支持多batch数据、多线程、多推理实例等不同并行方式

Outline

Arm架构及A64指令集、NEON指令简介

- Arm
  - 精简指令集 (RISC) 处理器架构
  - 通常使用定长指令集 (32-bit)
  - 广泛应用在移动端、嵌入式设备

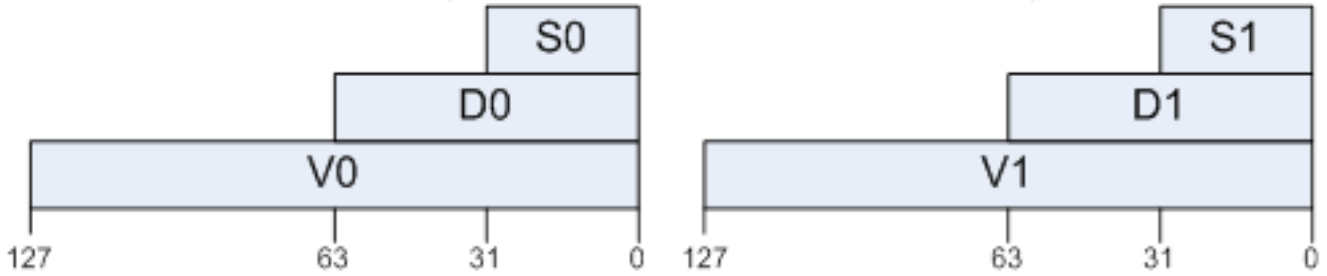


Arm架构及A64指令集、NEON指令简介

- Arm v8
  - 首次支持A64指令集架构（64位）
  - 更新NEON指令
- Arm v8.2
  - 支持 FP16 运算
  - （可选）支持SVE
- Arm v8.4
  - 支持有/无符号整数点积指令(SDOT, UDOT)
- Arm v8.6
  - 支持通用矩阵乘法（GEMM，[2x4]×[4x2]，混合精度）
  - 支持 BF16 运算
  - SIMD矩阵乘法运算
- Arm v9
  - 基于Arm v8.5特性
  - 支持SVE 2指令集

Arm架构及A64指令集、NEON指令简介

- NEON指令（Advanced SIMD)
  - 128-bit 向量数据处理，支持 1、2、4、8字节不同长度的元素
  - 支持全长 (128-bit) 或者半长 (64-bit) 向量计算，同时用于标量浮点数计算
  - 提供了32个向量寄存器




```
1  EOR  V0.16B,  V0.16B,  V0.16B
2  FMOV  V6.H[0],  6.0e+0
3  LD1  {V0.16B}, [X9]
4  FADD  V0.4S,  V0.4S,  V0.4S
5  FMLA  V0.8H,  V8.8H,  V16.8H
6  FMLA  V17.4S,  V9.4S,  V0.S[1]
```

Outline

Arm Server 微架构特点——以鲲鹏920处理器为例

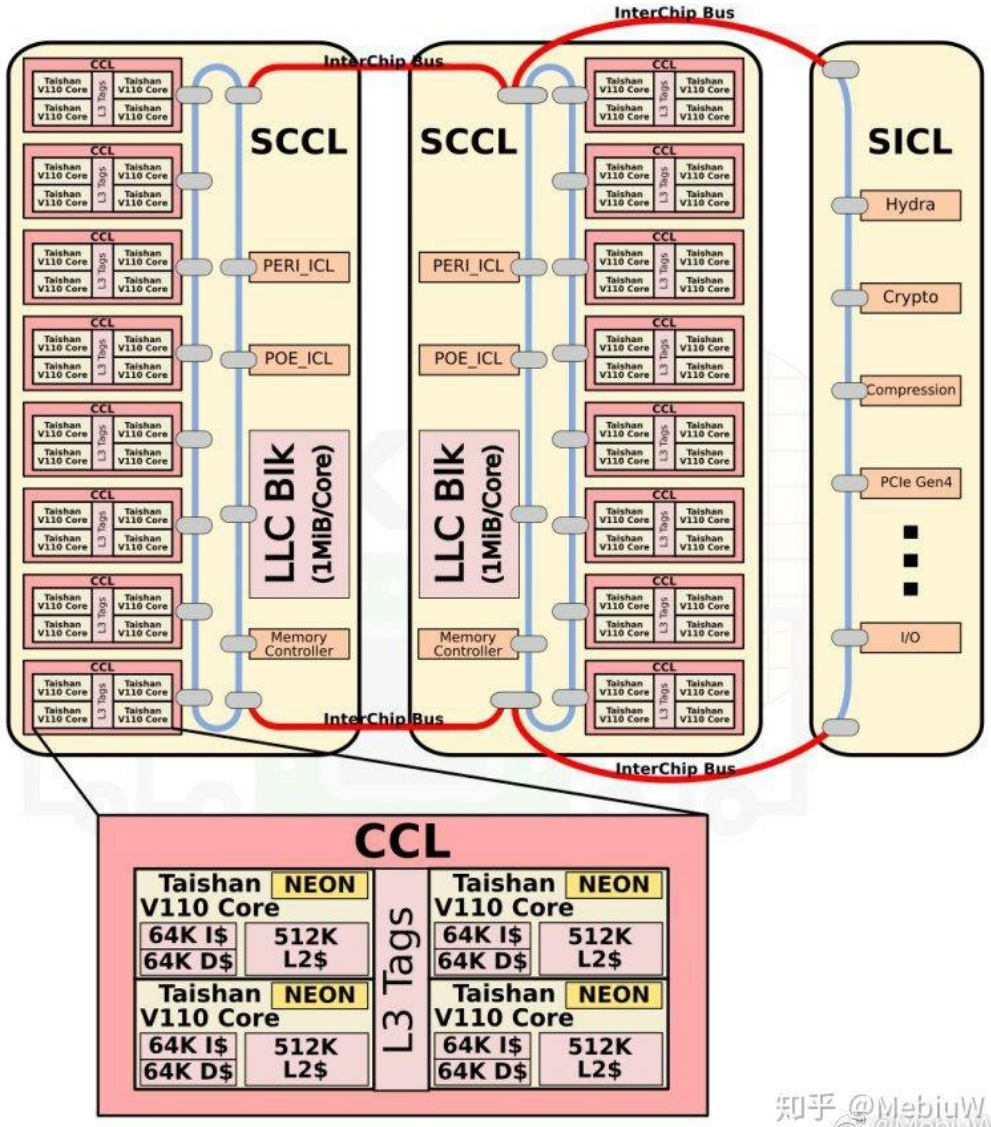
- 鲲鹏920处理器的体系结构特点——核心架构
  - TaiShan v110（Armv8.2-A ISA，非公版设计）
  - 前端：
    - 每周期最多取指4条
    - 支持静态、动态（两级）分支预测
  - 后端：
    - 乱序执行
    - 整数运算单元：3条ALU流水线，1条MDU流水线（整数乘除法运算）
    - AdvSIMD与浮点运算单元：两条FSU流水线(AdvSIMD)
    - Load/Store单元：两个L/S端口，含硬件自动预取器

Arm Server 微架构特点——以鲲鹏920处理器为例

- 鲲鹏920处理器的体系结构特点——核心架构
  - 缓存：
    - L1 ICache: 4路，64KiB (private)
    - L1 DCache: 4路，64KiB (private)
    - L2 Cache: 8路，512KiB (private)
    - L1 ITLB: 全关联，32表项（支持4KB~1GB页大小）
    - L1 DTLB: 全关联，32表项（支持4KB~1GB页大小）
    - L2 TLB: 4路组关联，1152表项
    - LLC/L3 Cache: 32~64MiB (1 MiB per core, shared)

Arm Server 微架构特点——以鲲鹏920处理器为例

- 鲲鹏920处理器的体系结构特点——片上系统
  - 处理器内核集群 CCL（Core CLuster）
    - 包含4个处理器内核及其L1 I/DCache、L2 Cache
    - 包含1个L3 Cache Tag
  - 超级内核集群 SCCL （Super CCL）
    - 包含6个/8个内核集群（CCL）
    - 包含2个I/O集群，4个DDR控制器
    - 包含若干L3 Cache Data，多核缓存一致性模块(HHA)
    - 环形总线互联
  - 相应I/O部件：I/O集群，超级I/O集群



鲲鹏920微架构性能测试

- 计算部分：

Instruction	Latency (cycles)	Throughput (IPC)
IADD (vector)	2	2
FADD (vector)	5	2
FMUL (vector)	5	2
FMADD (scalar)	5	2
FMLA (vector)	5	2
FDIV (vector)	17	0.08225

- Little's law：需要 5 × 2 = 10 条独立（无数据依赖）的FMLA指令，才能充分利用Neon单元

鲲鹏920微架构性能测试

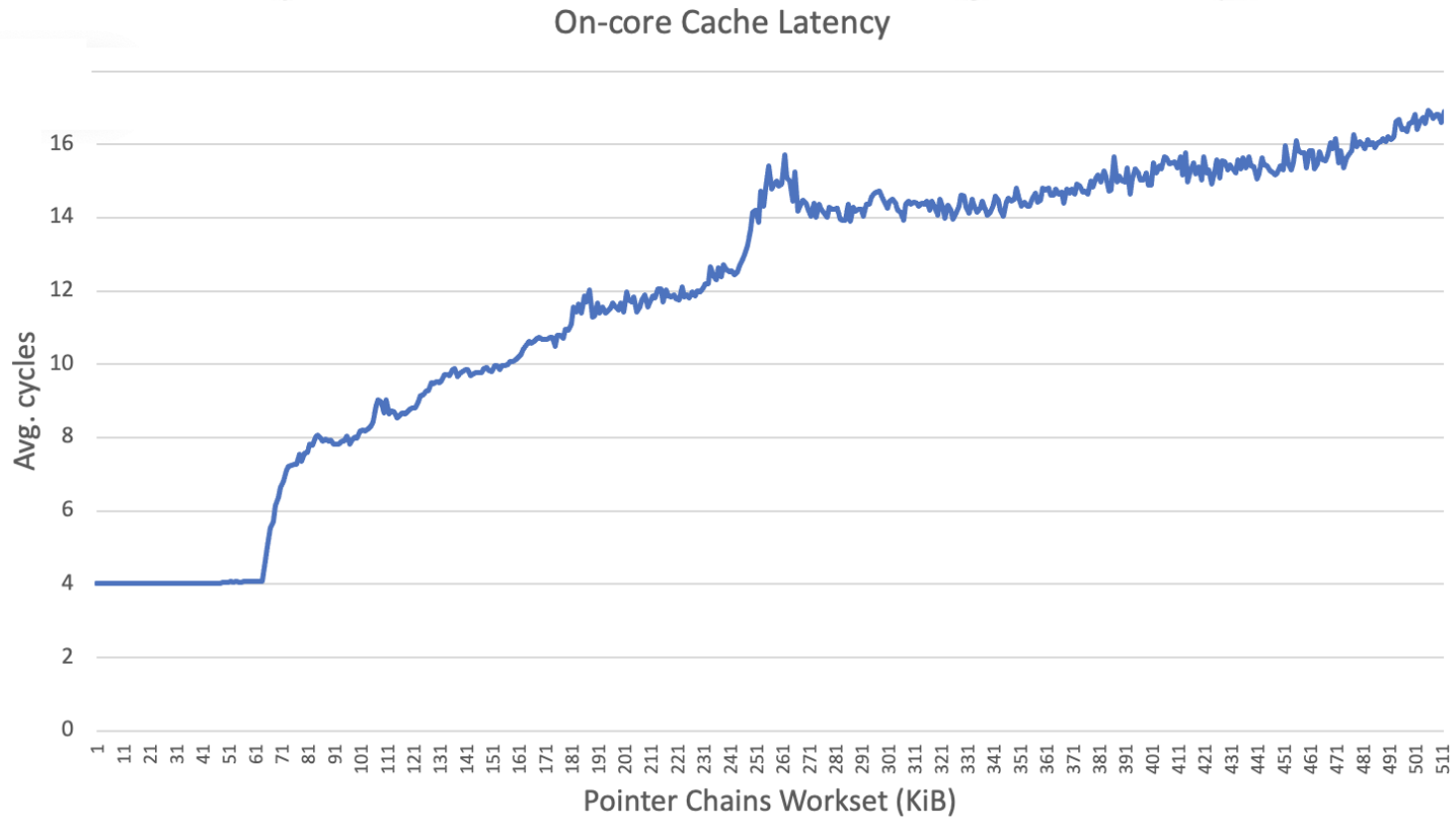
- 访存部分：

Instruction	Latency (cycles)	Throughput (IPC)
LDR/LD1 (16 bytes)	~4 (L1 cache hit)	1.85
STR/ST1 (16 bytes)		0.95

Location	Latency (cycles)
L1D Cache	~4
L2 Cache	8~9 (L1 TLB hit) 14~16 (L1 TLB miss)
L3 Cache	25~33 (local) 47~82 (remote)

鲲鹏920微架构性能测试

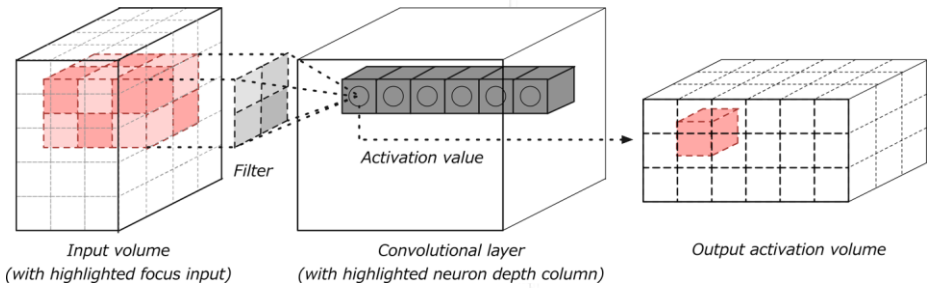
- 访存部分：
  - 使用一般的pointer chasing无法看到特征的L2, L3 Cache以及内存的load to use latency
  - 进一步的测试显示，在4K页的配置下，L2 Cache latency受到了L1 TLB miss的影响。

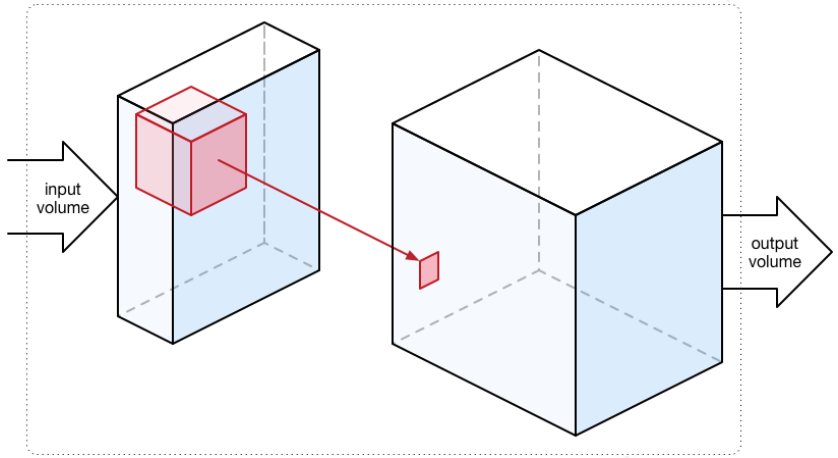


Outline

OpenPPL Arm Server 卷积实现分析

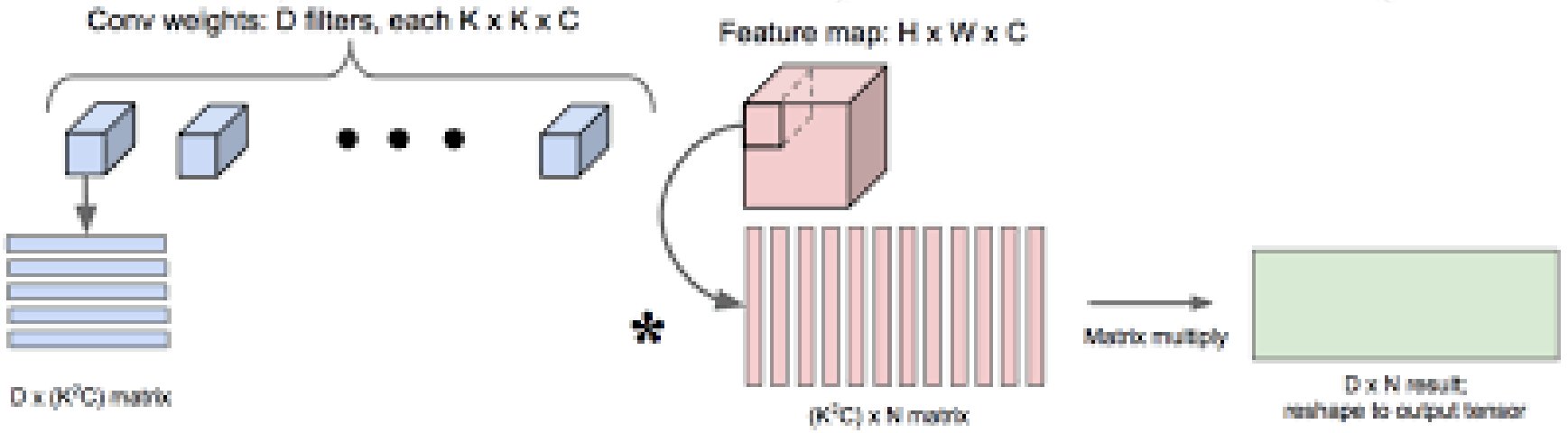
- 卷积
  - input: (N, C_i , H_i , W_i), output: (N, C_o , H_o , W_o), weights: (C_o , C_i , K_h , K_w)
  - 计算量： $C_o \times C_i \times K_h \times K_w \times H_o \times W_o \times 2$ FLOP
  - 并行维度： C_o , H_o , W_o
  - 累加维度： C_i , K_h , K_w





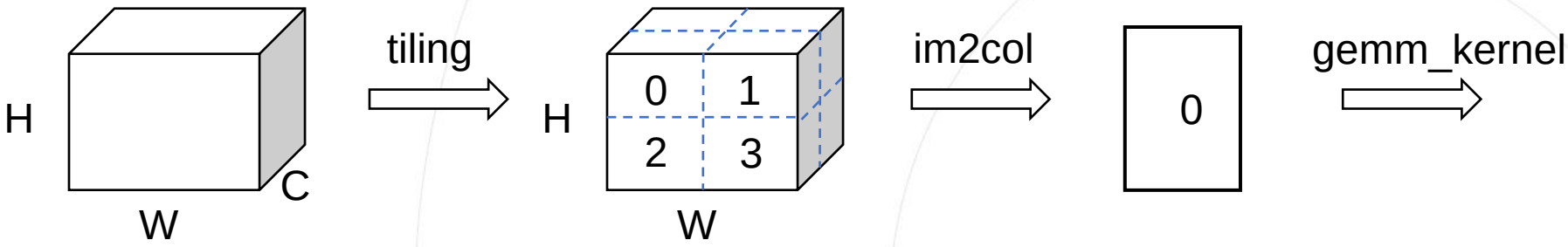
OpenPPL Arm Server 卷积实现分析

- 卷积—— 基础 Im2col-GEMM 实现
  - 矩阵乘法：$\mathcal{A}$ (weights) × $\mathcal{B}$ (im2col) = $\mathcal{C}$ (output)
  - $[C_o, C_i \times K_h \times K_w]$ $[C_i \times K_h \times K_w, H_o \times W_o]$ $[C_o, H_o \times W_o]$



OpenPPL Arm Server 卷积实现分析

- 算法优化
  - Tile Im2col：
    - 分块做im2col，减小用于im2col的额外内存开销



  - Direct：
    - 按卷积公式计算，无额外im2col开销
    - 在C_o维度上进行向量化

OpenPPL Arm Server 卷积实现分析

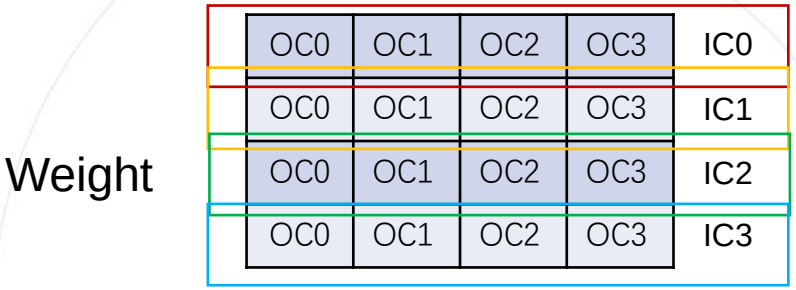
- 算法优化（续）
  - Winograd：
    - 降低卷积计算量，常用于 3×3、5×5 的卷积
    - 减小卷积中的累加维度，增加额外的并行维度
      - 例：BnF3中，卷积核 3^2 次的乘累加，转换成 $(n+2)^2$ 组并行点乘（矩阵乘法）
    - 需要额外的前后转换计算
    - 在计算机浮点数运算中会产生误差
      - 在 FP16 精度下更明显， 切分单元越大则误差越大

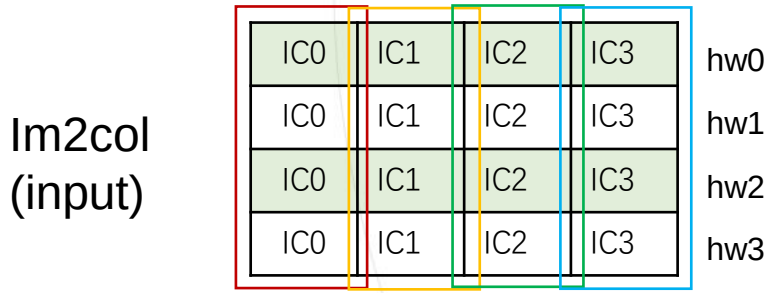
OpenPPL Arm Server 卷积实现分析

- 数据排布优化
  - Ndarray 排布 —— 原始张量格式 (n, c, h, w)
  - NBCX 排布 —— (n, c, h, w) → (n, c/c_{blk}, h, w, c_{blk})
    - 在 im2col转换 / 直接卷积 / winograd前后转换 中，易于利用Neon指令向量化
    - 在channel wise算子（如 pooling2d, resize等）中，充分利用Neon指令

OpenPPL Arm Server 卷积实现分析

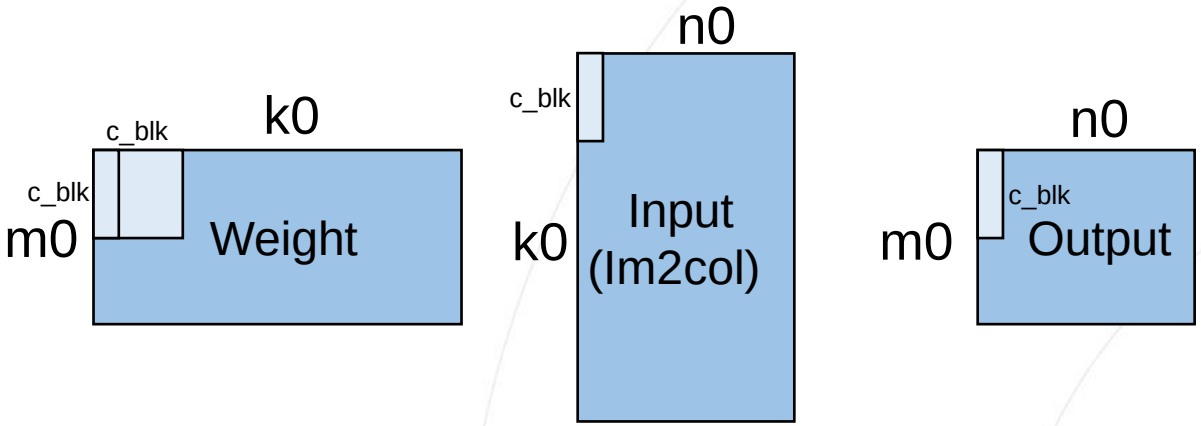
- 数据排布优化
  - NBCX 排布下，对卷积权重进行重排
    - $(C_o, C_i, K_h, K_w) \rightarrow (C_o / c_{blk}, C_i / c_{blk}, K_h, K_w, ic_{blk}, oc_{blk})$
    - 以外积的方式进行计算时，权重访存连续
    - 重排在离线处理/预处理的阶段完成





OpenPPL Arm Server 卷积实现分析

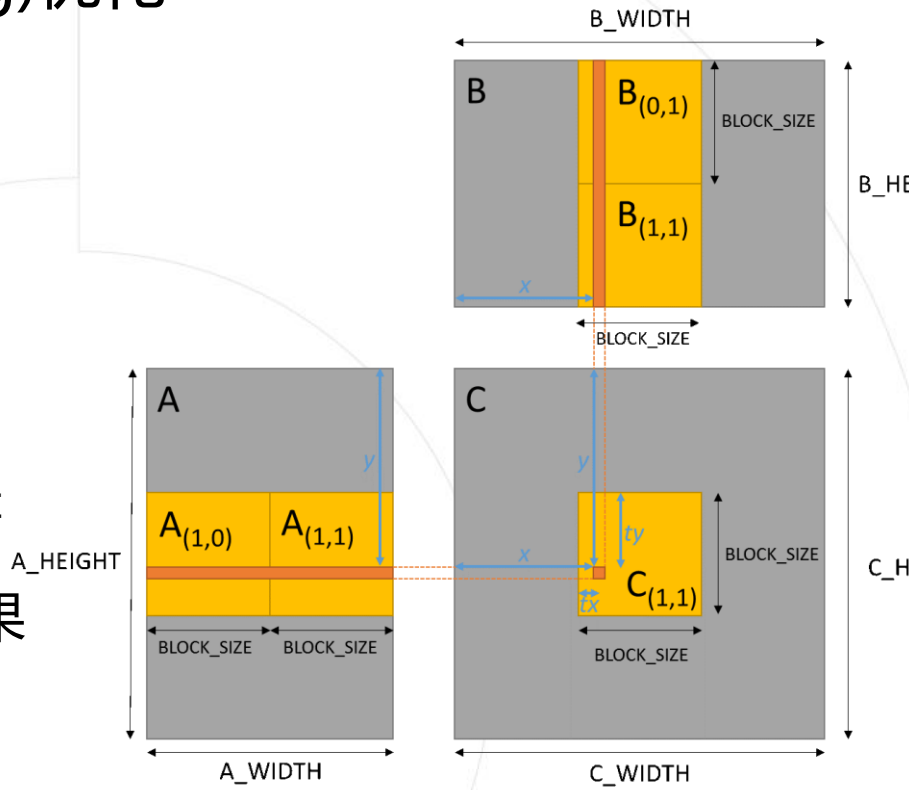
- GEMM分块计算优化 —— 寄存器分配



- 约束：#寄存器 = (m0 / c_{blk}) × n0 + n0 + 2 ~~(m0 / c_{blk})~~ ≤ 32
- 分块1： m0 = c_{blk}, n0 = 15 $\frac{\#VFMA}{\#VLD}$： fp16 – 5.21, fp32 – 3.15
- 分块2： m0 = c_{blk} × 2, n0 = 10 $\frac{\#VFMA}{\#VLD}$： fp16 – 6.15, fp32 – 4.44
- 分块3： m0 = c_{blk} × 3, n0 = 7 $\frac{\#VFMA}{\#VLD}$： fp16 – 5.42, fp32 – 4.42

OpenPPL Arm Server 卷积实现分析

- GEMM分块计算优化 —— 缓存复用(cache blocking)优化
  - 保留三个矩阵各自的一个分块在L1 DCache中
    - (m1 × k1 + k1 × n1 + m1 × n1) × sizeof(type) ≈ 64KB
  - 采用相对固定的分块大小
    - 支持在预处理阶段，搜索性能最优的分块大小
    - 权重数据在预处理阶段时，会按照分块大小进行重排优化访存
  - Kunpeng920上针对L2 Cache的blocking优化没有明显效果



OpenPPL Arm Server 卷积实现分析

- 多线程优化
  - 任务分割
    - 单batch内部，按 C_o , H_o , W_o并行维度分割
    - 多batch，按独立的 batch 分割
    - 分组卷积，按独立的分组分割
  - 任务并行调度 —— 基于OpenMP的多线程实现
  - 多线程绑定到同一NUMA节点内的核心

OpenPPL Arm Server 卷积实现分析

- 多线程 Conv2d – Im2col

for g_{outer} = 0 **to** #*group* **by** seg_{group} **do**
| **for** b_{outer} = 0 **to** #*batch* **by** seg_{batch} **do**
| | **if** *pack input for group* **do**
| | | *copy_from_input_to_buffer*
| | **end if**
| | **for** (g_{inner}, b_{inner}, hw_{outer}, oc_{outer}) = 0 **to** (seg_{group}, seg_{batch}, $H_o \cdot W_o$, C_o) **by** (1, 1, seg_{hw}, seg_{oc}) **do**
| | | *im2col_transform*
| | | *gemm_kernel*
| | **end for**
| | **if not** in-place group convolution **do**
| | | *copy_from_buffer_to_output*
| | **end if**
| **end for**
end for

parallel region

parallel region

parallel region

OpenPPL Arm Server 卷积实现分析

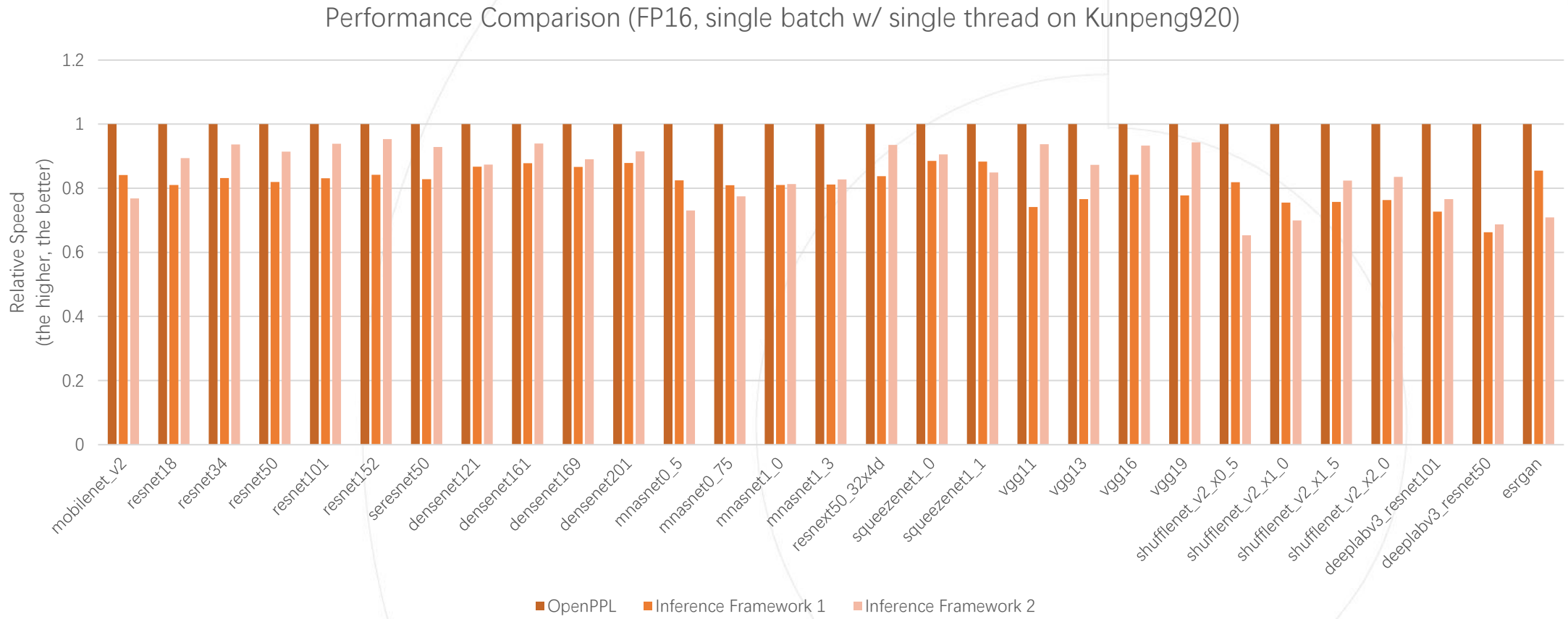
- 多线程 Conv2d - Winograd

for $g = 0$ **to** $\#group$ **by** 1 **do**
| **for** $idx_{wg\text{-}block} = 0$ **to** $\#wg\text{-}block$ **by** $seg_{wg\text{-}block}$ **do**
| | **for** $ic_{outer} = 0$ **to** C_i **by** seg_{ic} **do**
| | | *wg_preproccess* — parallel region
| | |
| | | **for** $oc_{outer} = 0$ **to** C_o **by** seg_{oc} **do**
| | | | *wg_gemm_kernel* — parallel region
| | | | **if** *last* C_i *segment* **do**
| | | | | *wg_postproccess* — parallel region
| | | | **end if**
| | | **end for**
| | **end for**
| **end for**
end for

Outline

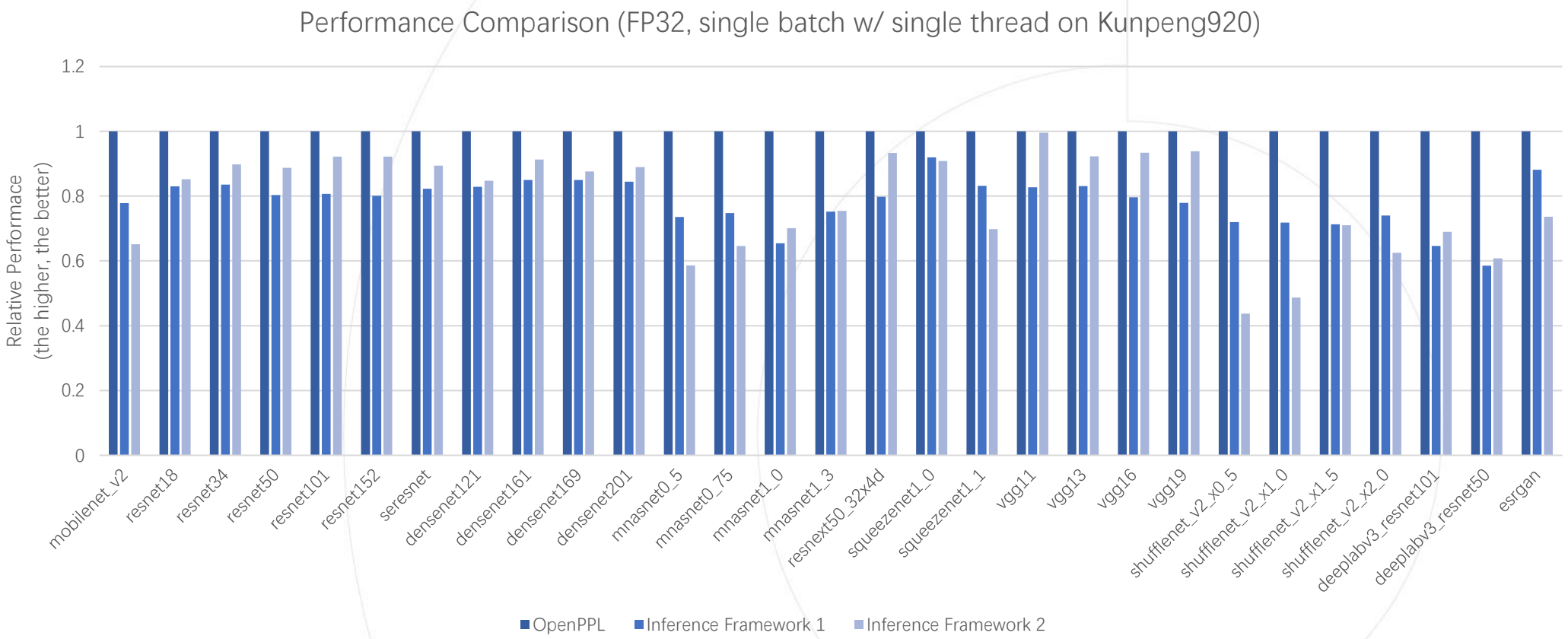
OpenPPL Arm Server 性能综合展示

- 推理性能——单线程单batch



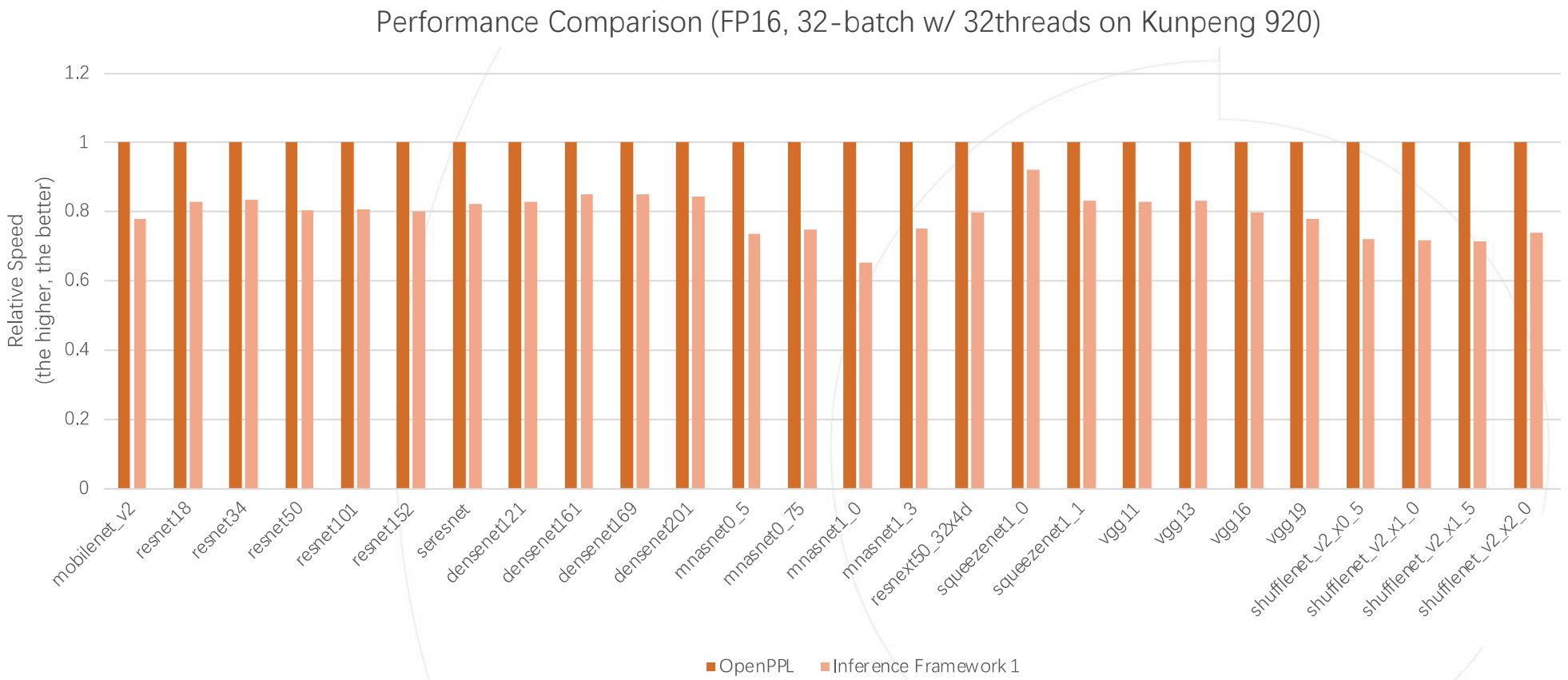
OpenPPL Arm Server 性能综合展示

- 推理性能——单线程单batch



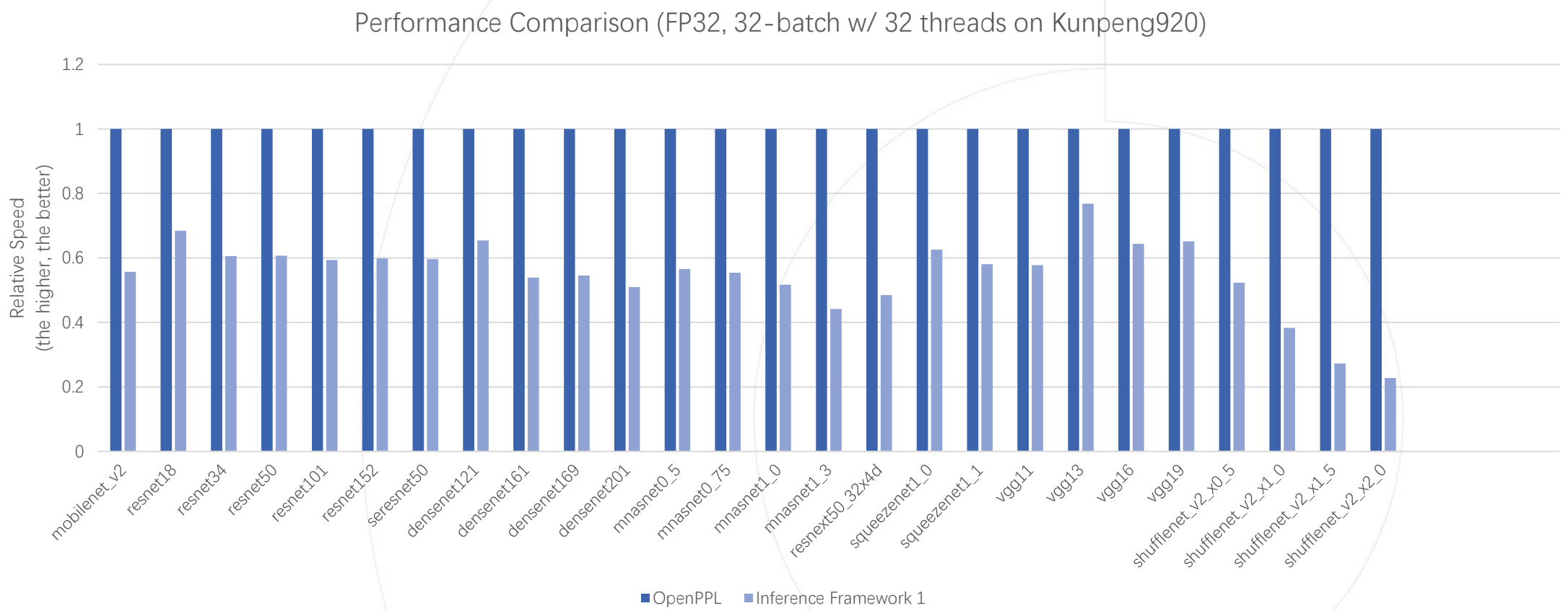
OpenPPL Arm Server 性能综合展示

- 推理性能——多线程多batch※



OpenPPL Arm Server 性能综合展示

- 推理性能——多线程多batch※



OpenPPL Arm Server 开发计划与展望

- 支持更多的网络模型和算子
  - 语义分割、目标检测等网络
- 支持
  - Int8 数据精度推理
  - Arm v9 特性支持

THANK YOU

Q & A

OpenPPL 微信公众号

OpenPPL QQ 交流群

- OpenPPL 官网主页：https://openppl.ai/
- OpenPPL GitHub 主页：https://github.com/openppl-public
- OpenPPL 知乎账号：https://www.zhihu.com/people/openppl

Thanks for listening!

Q&A Time

实战训练营