AMX Profiling Analysis

叶锦鹏

2026 年 1 月 13 日

中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

Large Language Models (LLMs)

- LLMs 已成为主流 AI 工作负载：生成、推理、对话、代码、搜索增强等。
- 典型结构：Transformer Attention + MLP 堆叠，计算密集、访存压力大。
- 推理特点：
  - Token-by-token 自回归生成，延迟敏感。
  - GEMM/Attention 占据 90%+ 的计算量。
  - 对矩阵乘、Cache、带宽、并行度高度敏感。
- CPU 推理需求增长：边缘部署、成本敏感、可扩展性强。

Intel AMX Hardware Overview

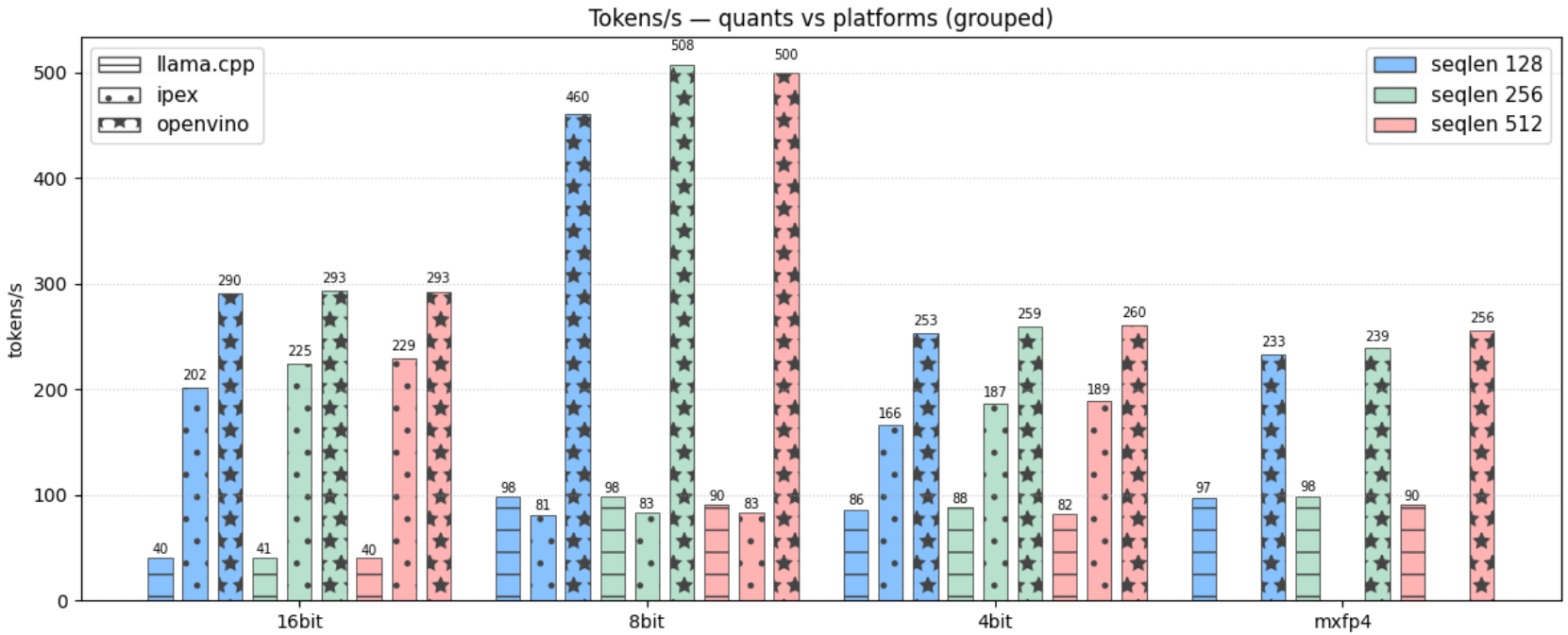
- AMX = Intel 新一代矩阵加速单元（Sapphire Rapids 起）。
- 关键组件：
  - **TILE** 配置：8x(16x64B) tile 寄存器阵列。
  - **TILELOAD/TILESTORE**：高带宽 tile 访存。
  - **TDP/TDPBUSDOT**：BF16/INT8 矩阵乘指令。
- 优势：
  - 大幅提升 GEMM/Attention 性能（比 AVX512 提升 2 – 4x）。
  - 适合 LLM 中的 QKV、FFN、KV Cache matmul。
- 挑战：
  - Tile 配置开销、调度复杂。
  - 需要框架级优化才能发挥峰值。

LLM CPU Framework Landscape

- **llama.cpp**
  - 轻量、跨平台、社区活跃。
  - AMX 支持有限，主要依赖手写 kernel。
- **IPEX (Intel Extension for PyTorch)**
  - 深度集成 PyTorch，支持 AMX BF16/INT8。
  - 主要优化 dense matmul，KV cache 优化有限。
- **OpenVINO GenAI**
  - 针对 LLM 的图优化 + oneDNN/AMX kernel。
  - KV Cache、Attention、GEMM、Prefill/Decode 全链路优化。
  - 支持 INT8、FP16、BF16、混合精度。

Performance Comparison on AMX

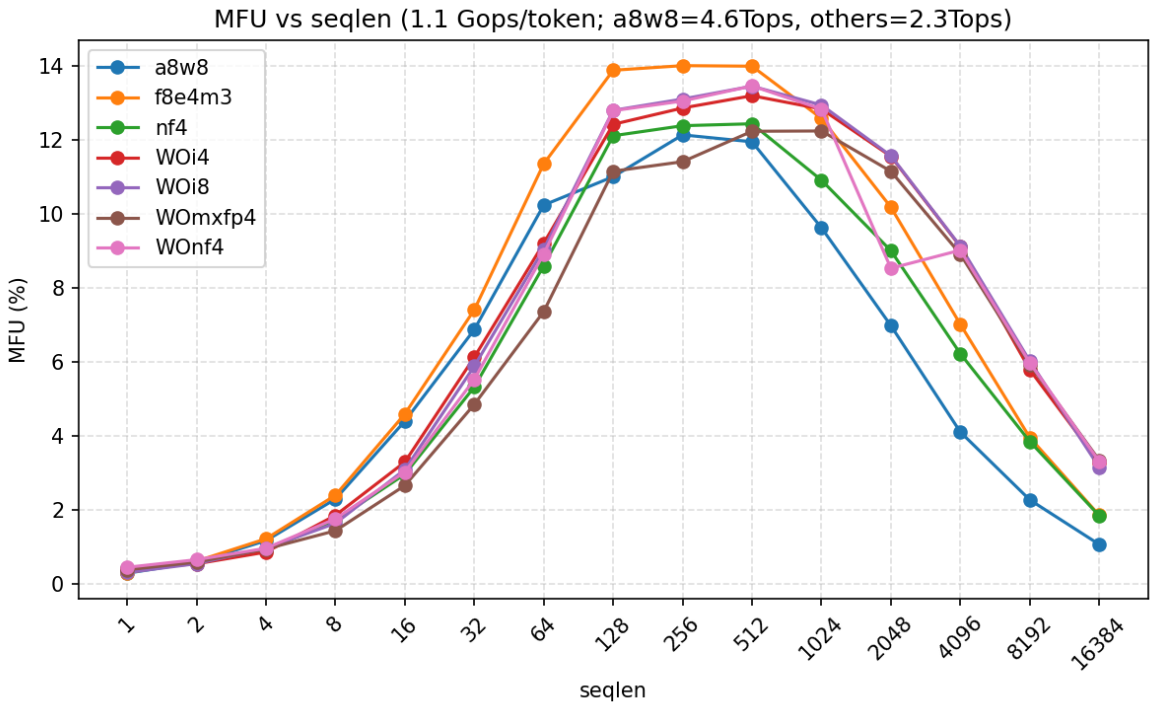
- 测试平台： Intel Xeon 8580 / AMX / Llama3.2-1B



结论： OpenVINO 通过图优化 + oneDNN AMX kernel ， 实现当前 矩阵乘扩展 Intel CPU 上最强的 LLM 推理性能。

OpenVINO AMX MFU 现状

- 尽管 OpenVINO 在 AMX 上进行了大量优化（图优化、算子融合、AMX kernel、KV cache 优化等），但实际测得的 **MFU(Model Flops Utilization)** 在不同量化模型（BF16、FP16、INT8、混合精度）上，均未达到 AMX 理论峰值。
- 这表明：**OpenVINO** 的 **AMX** 性能仍有较大优化空间。



AMX MFU 偏低的潜在原因

- 软件层面瓶颈
  - 图优化不完全：部分算子未融合、未进入矩阵乘子图。
  - Kernel 选择 suboptimal：未完全匹配 AMX tile 配置或数据排布。
  - 没有选择多层融合的激进层融合优化，如 Flash Attention，Gate-Up Fuse 等
- 硬件层面瓶颈
  - 内存带宽不足导致 AMX compute pipeline 空转。
  - L2/L3 命中率不足，KV cache 访问导致 stall。
- 结论
  - MFU 偏低是软硬件共同作用的结果。
  - 需要从图优化、kernel 生成、调度、tile 配置、数据布局等多方面深入分析。

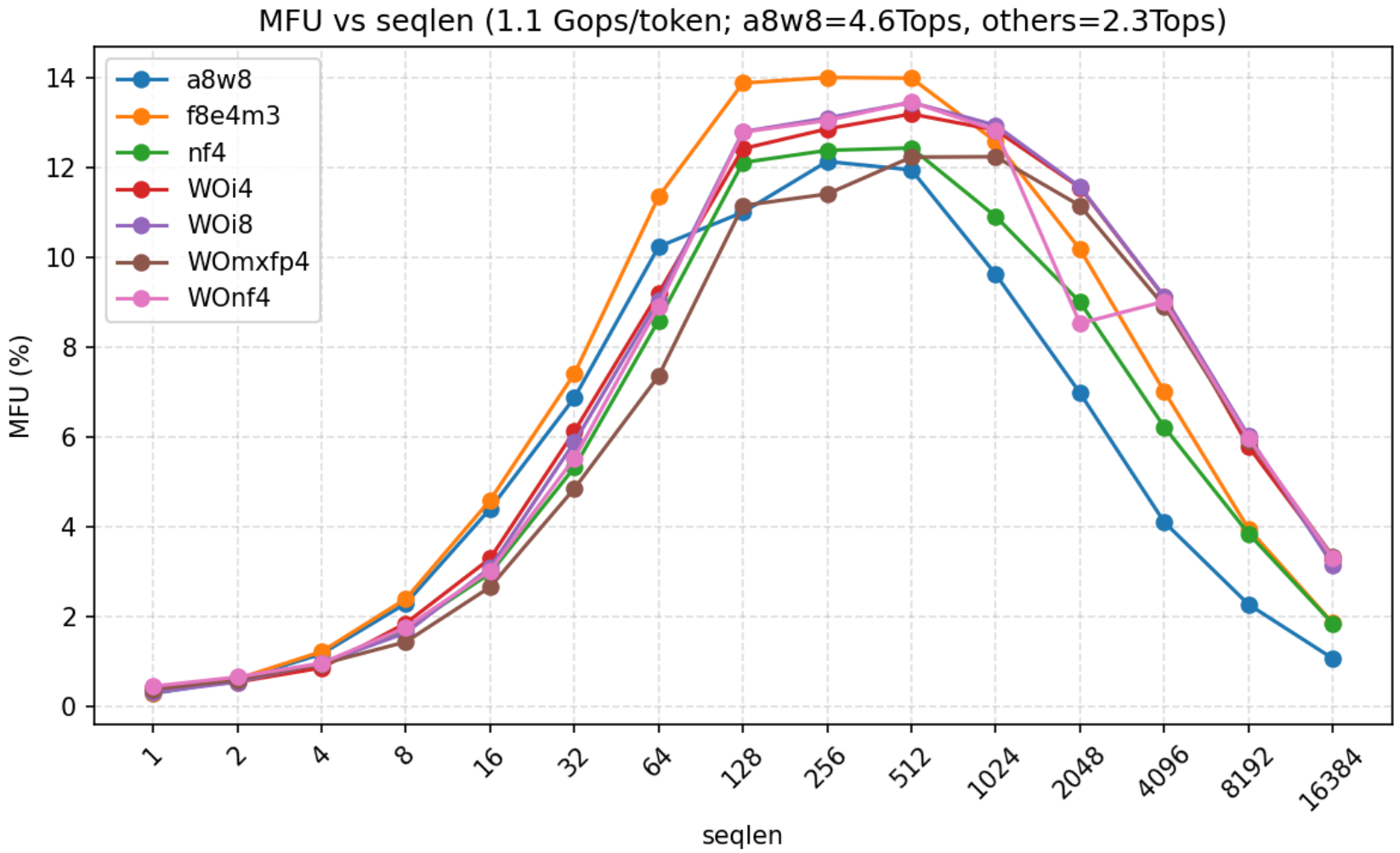
OpenVINO LLM 推理的执行流程

- 输入格式：ONNX / OpenVINO IR / GGUF（GenAI）
- **1.** 图优化（**Graph Optimization**）
  - 算子融合（MatMul+Bias、QKV、MLP 等）
  - 常量折叠、形状推理、子图重写
  - 生成更适合 AMX/oneDNN 的计算图
- **2. Kernel** 选择（**Kernel Selection**）
  - 根据数据排布（NCHW/NCX/blocked）选择最佳 kernel
  - 根据硬件特性选择 AMX / AVX512 / JIT kernel
  - 为算子添加必要的 reorder、layout 转换
- **3.** 即时编译（**JIT Compilation**）
  - 生成 AMX tile 配置、TDP/TDPBUSDOT 指令序列
  - 生成 BRGEMM 子图的二进制代码
- **4.** 推理执行（**Execution**）
  - Prefill + Decode 阶段执行
  - KV cache 管理、调度、并行化

这一流程的每一步都可能影响 **AMX** 的 **MFU**。

图优化后的 OP 占比分析

OpenVINO 的 AMX 性能仍有较大优化空间

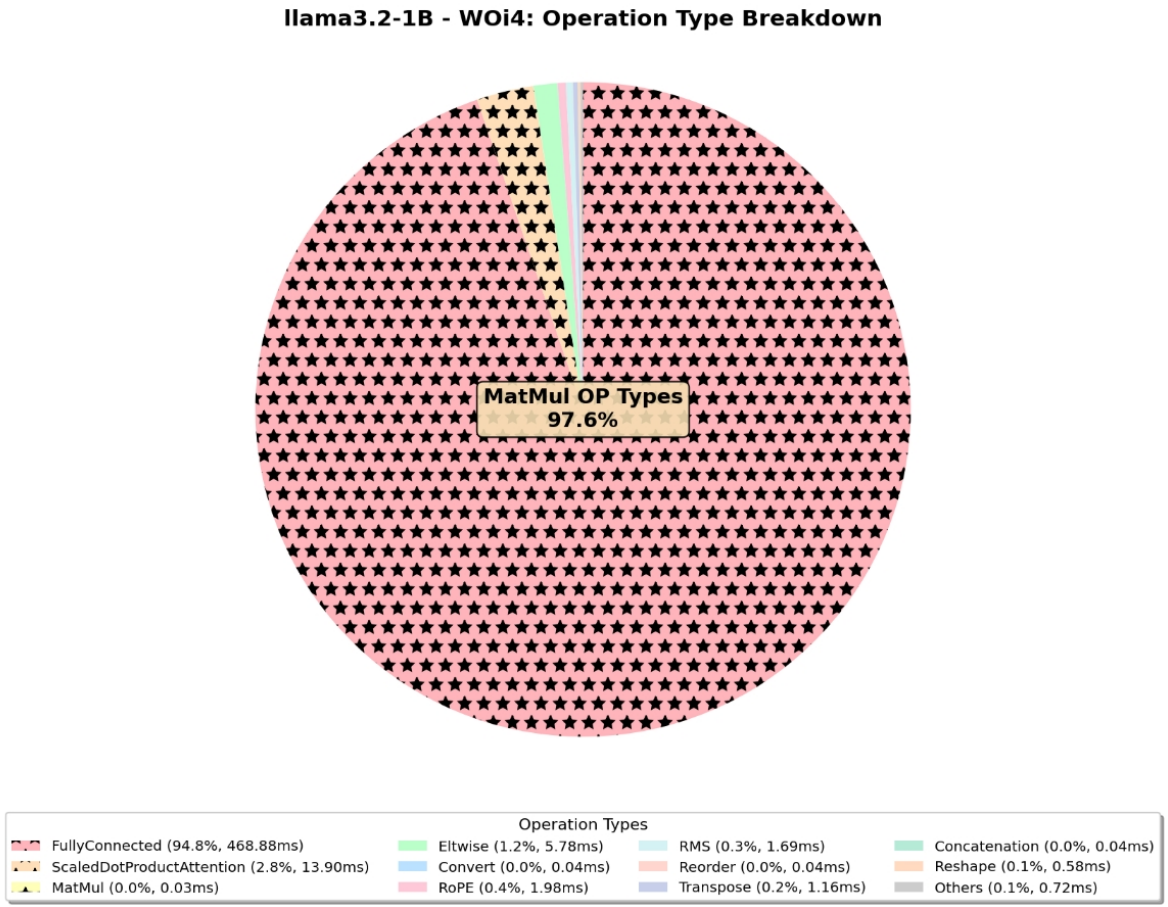


图优化后的 OP 占比分析

- 我们首先从最高层次观察：图优化（Graph Optimization）之后，各类算子在执行图中的占比。
- 展示三张饼图：分别对应 BF16、Weight-Only mxfp4、A8W8（SmoothQuant）三种量化模型。
- 观察结果：
  - MatMul / GEMM 仍然占据绝对主导地位。
  - 优化 MatMul 的软硬件路径仍然是最关键的性能突破口。
  - OpenVINO 已将大部分量化/反量化操作成功融合，但仍有部分后处理算子未完成融合。
  - OpenVINO 仍缺乏先进的层融合优化技术如 FA(Flash attention) 等。

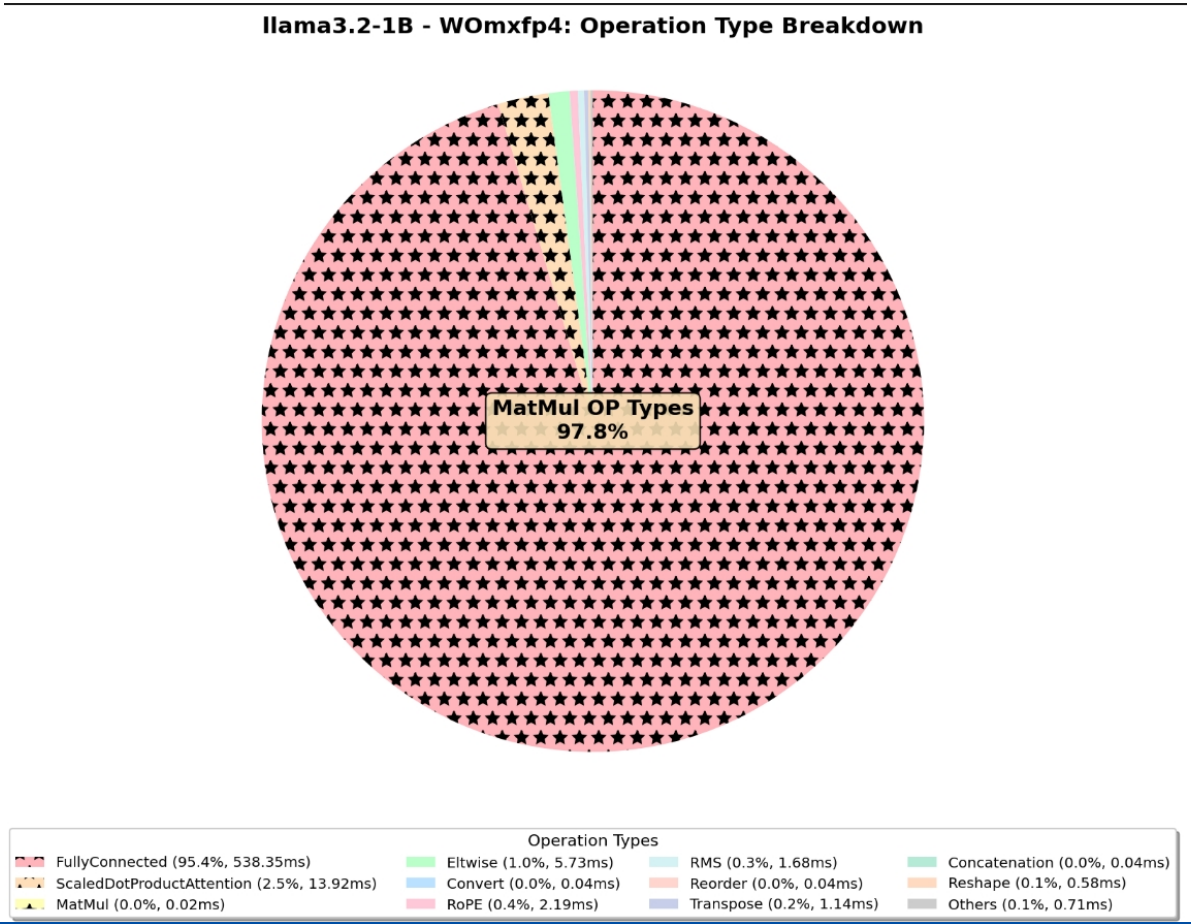
图优化后的 OP 占比分析

MatMul / GEMM 仍然占据绝对主导地位。OpenVINO 已将大部分量化/反量化操作成功融合进 MatMul。但 Gate-Up Fusion 的 Pass 未生效。



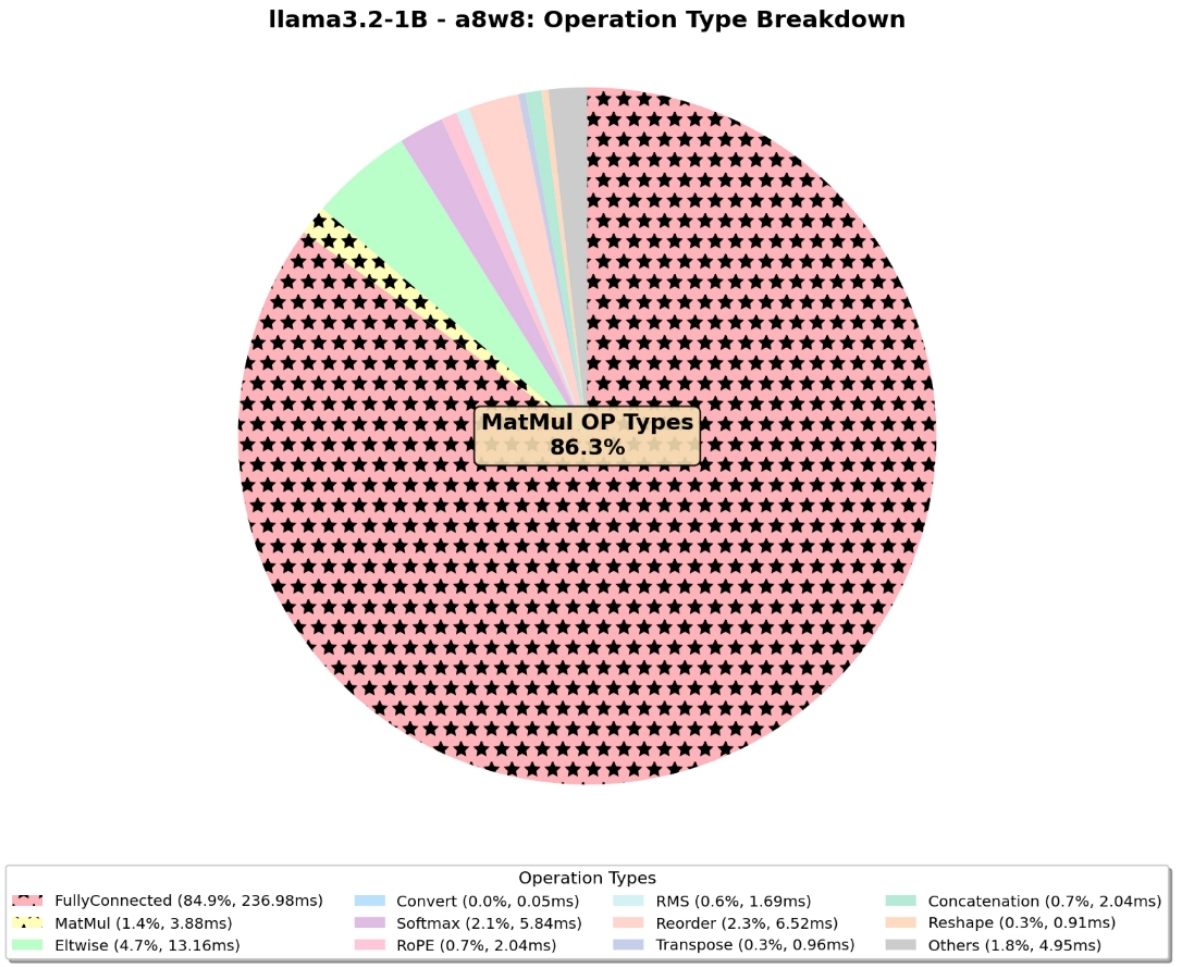
图优化后的 OP 占比分析

MatMul / GEMM 仍然占据绝对主导地位。OpenVINO 已将大部分量化/反量化操作成功融合进 MatMul。但 Gate-Up Fusion 的 Pass 未生效。



图优化后的 OP 占比分析

SDPA 被展开了，被展开成了 Matmul 和 softmax 等操作，smoothquant 等工业界认可的先进量化实现的优化 pass 未实现。



SmoothQuant A8W8 中出现 SDPA：跨层融合缺失

- 在 SmoothQuant A8W8 模型中，我们意外发现：
  - **SDPA（Scaled Dot-Product Attention）**仍然以独立的被分解的 **OP** 存在。
- 这意味着：
  - Q/K/V -> Attention -> Softmax -> MatMul 的跨层融合并未发生。
  - OpenVINO 的 SDPA fusion pass 在该量化路径中未生效。
- 影响：
  - 整个 Attention 层的多级优化无法实施，后续 kernel 选择只能回退到选择基础的矩阵乘 kernel。

层融合仍不彻底：后处理算子未完全融合

- 尽管量化/反量化已被融合，但我们仍观察到大量后处理算子（post-ops）未被融合：
  - Add、Mul、BiasAdd、RoPE 等仍以独立算子存在。
  - 说明当前的层融合 pass 仍不够彻底。
- 这些未融合的 post-ops 会导致：
  - 额外的 kernel 调度开销。
  - 额外的内存读写与 layout 转换。
  - AMX pipeline 被频繁打断，降低 MFU。

Gate-Up Fusion 未生效：激进融合优化缺失

- 阅读 OpenVINO 源码可以看到：
  - **Gate-Up Fusion pass** 是存在的。
- 但在实际执行图中：
  - 并未观察到 gate + up-projection 的融合算子。
  - 说明该 pass 在 LLM 模型中未被触发或未完全实现。
- 影响：
  - MLP 仍然拆分为多个 MatMul + post-ops。
  - 无法形成更大的 BRGEMM 子图。
  - Kernel 选择上，丧失了优化空间。

未发现 FlashAttention：缺乏先进 Attention 优化

- 在所有模型（BF16/MXFP4/A8W8）中，我们均未观察到 FlashAttention 或类似的 fused attention kernel。
- 这意味着：
  - Attention 仍然是传统的 MatMul + Softmax + MatMul 结构。
- 影响：
  - Prefill 阶段的 MFU 无法提升。
  - AMX 的 compute pipeline 无法保持高利用率。

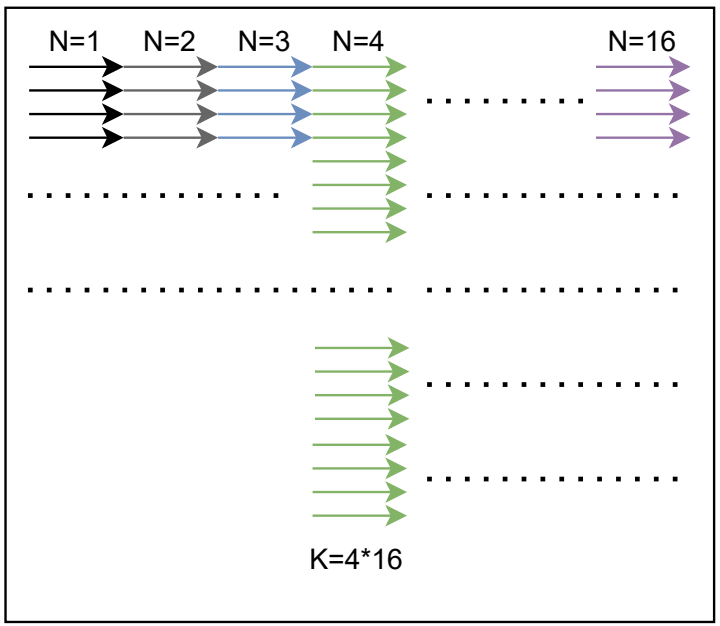
小结：问题 1 — 图优化步骤缺乏先进的层融合优化技术

- 通过自顶向下分析，我们发现：
  - 图优化后的 OP 占比仍然以 MatMul 为主，但 post-ops 未充分融合。
  - SDPA 未被跨层融合，Attention 仍然走基础实现。
  - Gate-Up Fusion pass 存在但未生效。
  - 未发现 FlashAttention 或类似的先进 Attention kernel。

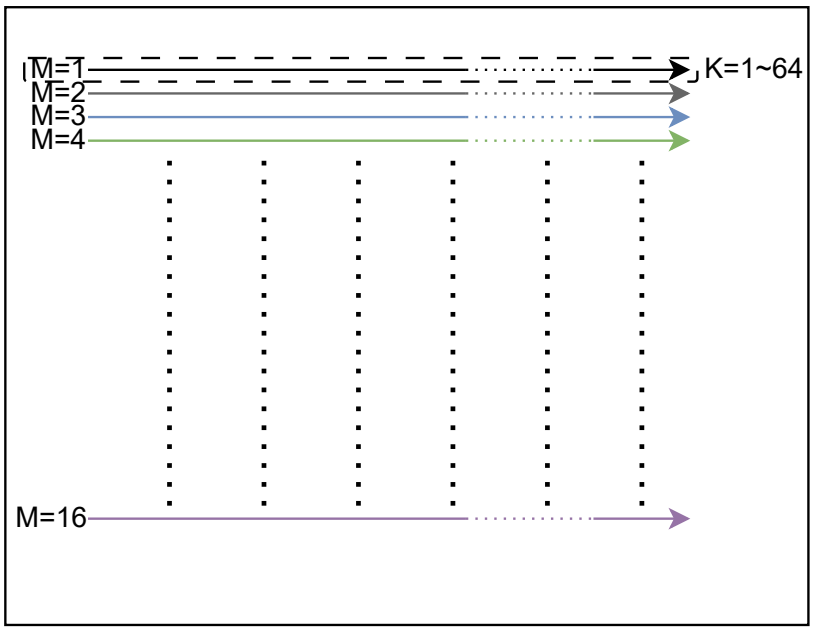
问题 1：OpenVINO 缺乏先进的层融合优化技术与实现。

Kernel Selection：图优化后的 OP 如何映射到 Kernel

- 图优化阶段完成后，OpenVINO 会将每个 OP 映射到具体的执行 kernel。
- 对于 AMX 硬件，kernel 选择受到严格的数据排布（layout）要求：
  - AMX tile 需要特定的 block 格式（如 16x64B tile）。
  - 输入张量必须在进入 AMX 计算单元前完成 layout 对齐。



Tile Src B



Tile Src A

Kernel Selection：图优化后的 OP 如何映射到 Kernel

- 图优化阶段完成后，OpenVINO 会将每个 OP 映射到具体的执行 kernel。
- 对于 AMX 硬件，kernel 选择受到严格的数据排布（layout）要求：
  - AMX tile 需要特定的 block 格式（如 16x64B tile）。
  - 输入张量必须在进入 AMX 计算单元前完成 layout 对齐。
- 这导致在 OP -> Kernel 转换过程中插入大量额外算子：
  - **reorder**（layout 转换）
  - **reshape**（形状调整）
  - **concat/split**（张量拼接与切分）
- 问题在于：这些向量 **kernel** 大多没有被融合优化，形成大量独立 kernel 调度。

AMX 支持的数据格式有限：导致现场反量化（dequant）

- 当前在售的 AMX 硬件仅支持两种输入格式：
  - **INT8**
  - **BF16**
- 虽然 OpenVINO 的 OP 和 Kernel 层面支持多种量化格式（INT4、INT8、MXFP4、NVFP4、混合精度等），但最终执行时仍必须转换为 AMX 支持的格式。
- 这意味着：
  - INT4/MXFP4/NVFP4 等格式在执行前必须进行 现场反量化（**dequant**）。
  - 如果反量化算子没有和 AMX 的计算执行充分 Overlap，将存在性能损失。
- 影响：
  - 增加额外访存与算子开销。
  - 可能破坏 AMX 的连续计算流，降低 MFU。
  - 量化模型的优势被部分抵消。

小结：Kernel-select 阶段的问题（潜在问题）

通过自顶向下分析，我们发现：

- Kernel-select 阶段暴露出两类潜在问题：
  - **(1)** 存在未融合的向量 **kernel**
    - reorder / reshape / concat 等频繁出现
    - 破坏 AMX pipeline 的连续性
  - **(2) AMX** 数据格式受限 -> 现场反量化
    - INT4/FP16/A8W8 等格式最终都需转换为 INT8/BF16
    - dequant kernel 融合，增加额外开销

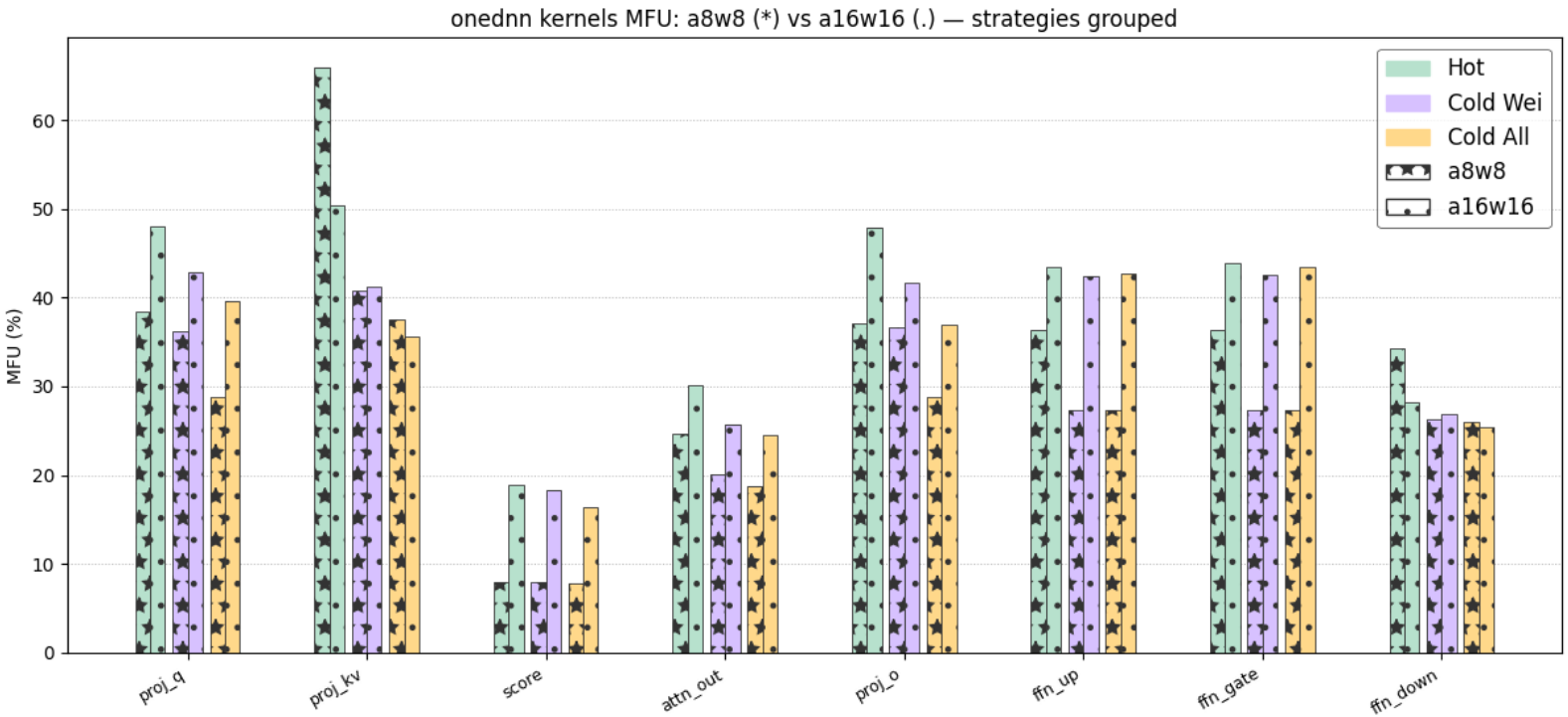
潜在问题：**Kernel-select** 阶段，面对现有 **AMX** 的硬件实现存在一定困难。如若软硬件配合不恰当，则导致 **MFU** 进一步下降。

Kernel Implementation：最终执行的核心 MatMul Kernel

- 图优化与 Kernel-select 完成后，最终的计算会落到两类 kernel 上：
  - **oneDNN** 库中的预编译 **kernel**（DNNL 动态库）
  - **oneDNN JIT** 框架生成的 **AMX kernel**（BRGEMM / MatMul）
- 为了排除所有外部因素，我们构造了最纯净的测试场景：
  - 仅执行核心 MatMul（无 post-op）
  - 无量化/反量化
  - 无 reorder / reshape
  - AMX pipeline 完整、无打断

Kernel Implementation：最终执行的核心 MatMul Kernel

- 结果却显示：即使在最纯净的 **AMX MatMul kernel** 中，**MFU** 仍然显著偏低。
- 这说明：问题不仅来自图优化或 **kernel-select**，连最底层的 **AMX kernel** 本身也未能充分利用硬件。



小结：问题 3 — AMX 核心 Kernel 的 MFU 本身偏低

- 通过对 oneDNN 的核心 MatMul kernel 进行独立测试，我们发现：
  - 即使没有任何 层融合
  - 即使没有任何 layout 转换或反量化
  - 即使 AMX pipeline 完整、无打断
- **AMX kernel 的 MFU 仍然无法接近理论峰值。**
- 这意味着：
  - AMX tile 配置、访存模式、调度策略可能存在结构性瓶颈。
  - oneDNN 的 AMX kernel 实现可能尚未达到硬件的最佳利用率。
  - MFU 偏低不仅是“外部因素”，也是“内部 kernel 实现”的问题。
- 因此我们得到：

问题 3：**AMX 核心 kernel 的实现本身存在性能瓶颈，MFU 偏低。**

Top-to-Down 自顶向下分析总结：三个关键问题

- 我们发现 OpenVINO 在 AMX 上 MFU 偏低并非单点问题，而是多层次瓶颈叠加的结果。
- 问题 1：图编译步骤缺乏先进的层融合优化技术
  - 部分 post-ops 未融合（Add/Mul/Activation 等）。
  - Gate-Up Fusion pass 存在但未生效。
  - 未发现 FlashAttention 或类似的高性能 Attention kernel。
- 潜在问题 2：**Kernel-select** 阶段选择困难
  - 为满足 AMX tile layout，插入大量 reorder/reshape/concat。
  - 这些向量 kernel 未融合，导致 pipeline 频繁中断。
  - AMX 仅支持 INT8/BF16 多种量化格式需现场反量化。
- 问题 3：**AMX 核心 kernel 实现本身 MFU 偏低**
  - 即使在最纯净的 MatMul（无 post-op、无量化、无 reorder）中，AMX kernel 的 MFU 仍远低于理论值。
  - oneDNN AMX kernel 在 tile 配置、访存模式、调度上存在结构性瓶颈。
- 三个问题共同导致：**AMX** 的理论性能无法在 **OpenVINO** 中充分释放。

细粒度分析：从 Kernel OP Task 的 MFU 下降链路

- 本章将带着前文的三个问题，逐层分析 MFU 下降的根因。
- 我们对每个 LLM task（如 proj_q、proj_k、MLP、SDPA 等）测量：
  - oneDNN 核心 kernel 的 MFU
  - MatMul OP 的 MFU（含部分 post-op）
  - OpenVINO 完整 task 的 MFU（含 reorder、dequant、KV cache 等）
- 通过对比这三层 MFU，我们可以清晰看到：

  MFU 从 60% -> 30% -> 10% 的逐层下降链路

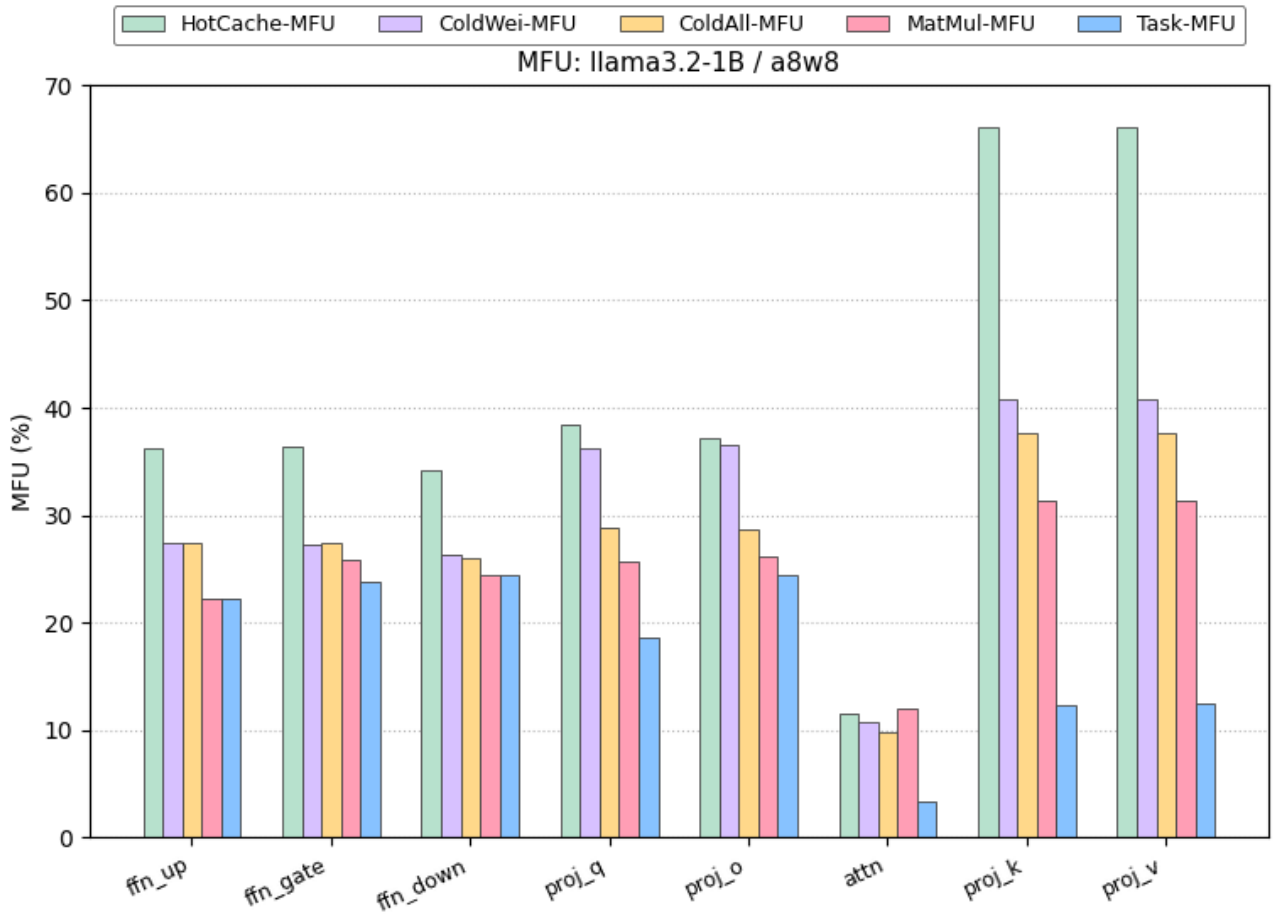
- 这将帮助我们对 MFU 最终掉到 10% 左右进行归因。

A8W8（SmoothQuant）性能变化：供数能力不足 + 后操作未融合

- 展示 A8W8 模型的 MFU 变化图（oneDNN MatMul Task）。
- 观察结果：
  - proj_q / proj_k 的 MFU 明显下降。
  - 主要原因：
    - 供数能力不足（memory-bound）
    - 部分 post-op（如 bias、activation）未融合
- 结论：**a8w8** 瓶颈主要来自数据供给不足 + 层融合不彻底。

A8W8（SmoothQuant）性能变化：供数能力不足 + 后操作未融合

a8w8 瓶颈主要来自数据供给不足 + 层融合不彻底。

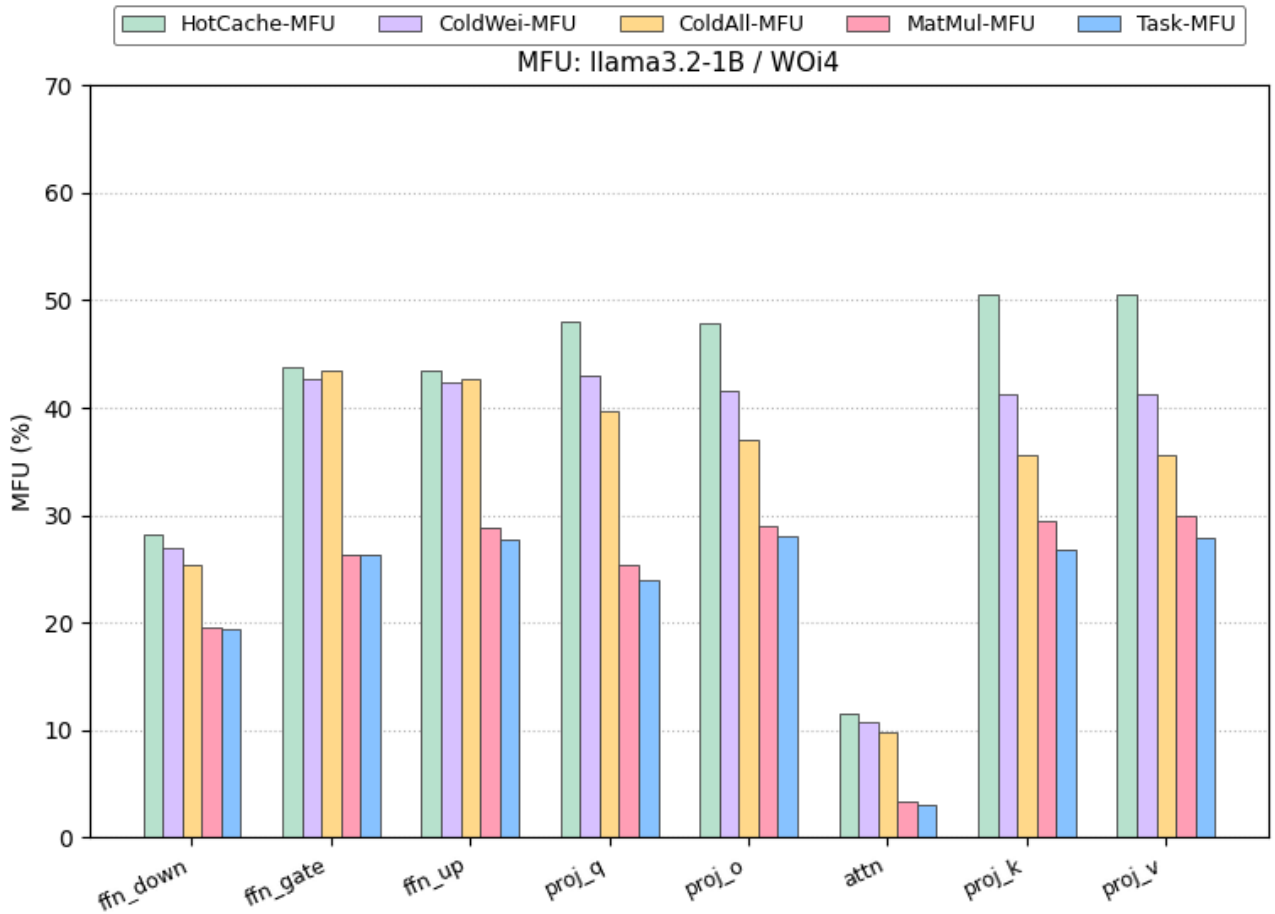


WOI4 性能变化：反量化融合后 MFU 明显下降

- 展示 WOI4 的 MFU 变化图。
- 观察结果：
  - oneDNN kernel MFU 尚可，但 OP 层开始急剧下降。
  - 主要原因：
    - 反量化（**dequant**）融合后，**AMX** 无法直接处理 **INT4 BF16/INT8** 转换
    - dequant kernel 无法融合进主计算 kernel
- 结论：**WOI4** 的瓶颈来自反量化开销 + **AMX** 数据格式受限。

WOI4 性能变化：反量化融合后 MFU 明显下降

WOI4 的瓶颈来自反量化开销 + **AMX** 数据格式受限。

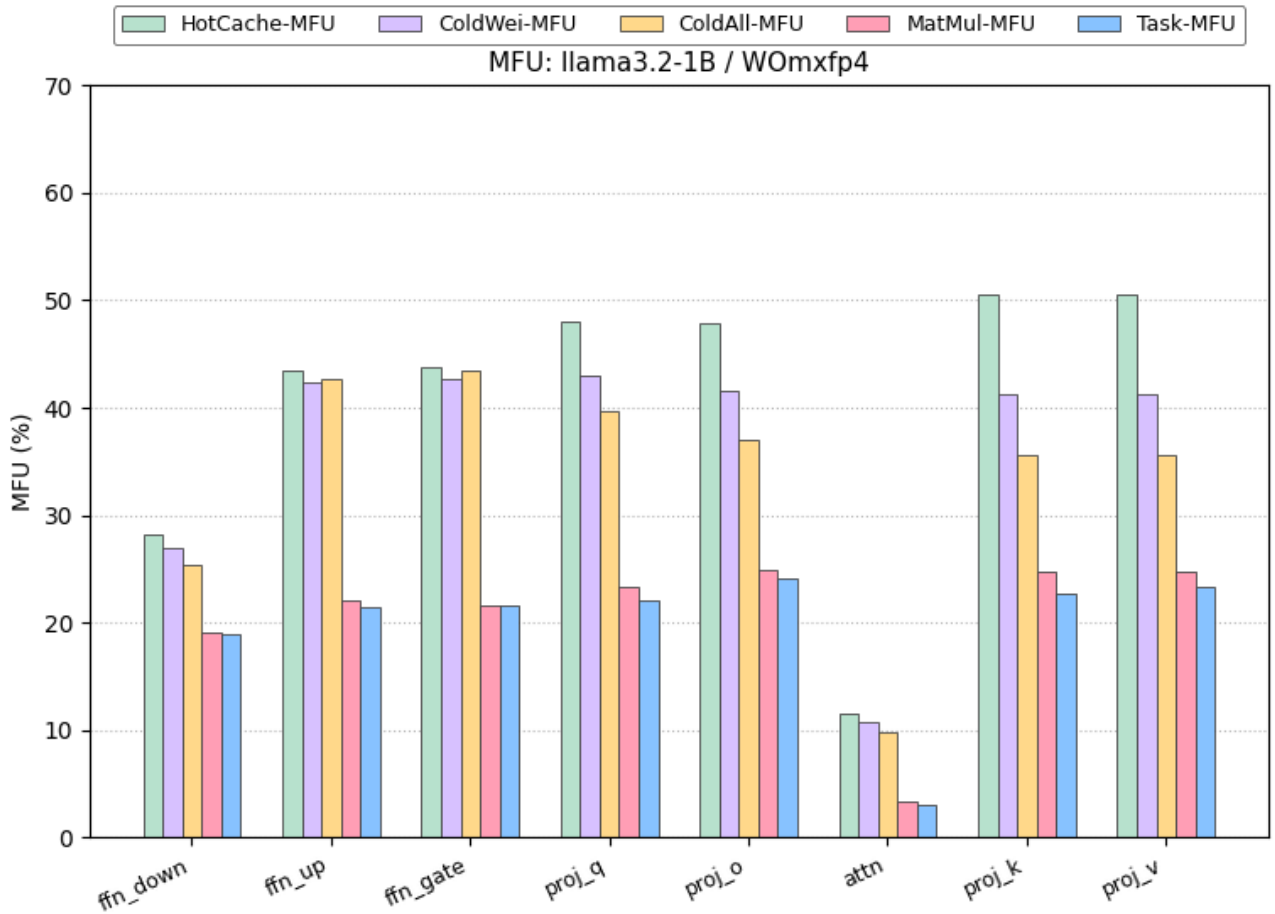


WOMXFP4 性能变化：SDPA 融合算子未生效

- 展示 WOMXFP4 的 MFU 变化图。
- 观察结果：
  - SDPA 的融合算子存在于代码中，但实际执行中：并未触发高效融合实现
  - Attention 仍然走 fallback kernel，导致 MFU 大幅下降。
- 结论：
  WOMXFP4 的瓶颈来自 **SDPA** 融合缺失 + 反量化开销。

WOMXFP4 性能变化：SDPA 融合算子未生效

WOMXFP4 的瓶颈来自 **SDPA** 融合缺失 + 反量化开销。



三个问题被完全验证：进一步重整为四个子问题

- 通过 A8W8、WOI4、WOMXFP4 的细粒度分析，我们确认：

  前三章提出的三个问题全部是真问题。

- 并进一步重整为四个更细的子问题：
  1. 缺乏激进/先进的层融合优化 **Pass**
  2. **oneDNN** 的 **AMX kernel MFU** 本身偏低
  3. 部分 **task** 是向量瓶颈，需要更强的向量部件
  4. **AMX** 硬件能力受限，导致大量向量任务拖慢性能
- 这四个子问题构成了 MFU 下降的完整因果链。

体系结构优化建议（软件 + 硬件）

- 针对问题 1：层融合不足
  - 建议 OpenVINO 尽快实现有效的融合算子（SDPA、Gate-Up、post-op 融合）。(CUTE 已实现)
- 针对问题 2：oneDNN kernel MFU 偏低
  - 软件：XJN 的预取优化使 MatMul 可稳定达到 70% MFU (all hot)。
  - 硬件：建议增加 AMX 暂存区大小（CUTE 已实现）。
- 针对问题 3：向量瓶颈
  - 建议 AMX 增加更多数据格式支持（Micro 2025 已实现解压缩加速器）。
  - 增加 exp2 等硬件 (CUTE 正在实现)，利用 SMT 实现向量-矩阵 overlap（CUTE 已实现）。
- 针对问题 4：AMX 数据格式受限
  - 建议统一输入/输出数据排布，减少向量任务（CUTE 已实现）。
  - 尝试融合更多后操作（CUTE 已实现）。

本章总结：MFU 下降的完整归因链

- 通过 Kernel OP Task 的 MFU 逐层分析，我们得到：**MFU** 下降 = 层融合不足 + **layout** 转换 + 反量化开销 + **AMX kernel** 本身偏弱
- 三个问题 -> 四个子问题 -> 软件/硬件优化建议