PR #35760 完整报告

sgl-project/sglang

[Perf] Tune the W4AFP8 DeepEP low-latency requant launch geometry

合并时间：2026-08-30 07:03

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35760

执行摘要

- 一句话：重调 W4AFP8 requant 启动几何，decode 内核提速最高 12x
- 推荐动作：值得精读。三个设计决策有普适价值： 1) tile 大小以 bytes-per-lane 而非元素数定义，天然兼容 wave32/wave64 厂商差异； 2) 行估计必须向下取整到 2 的幂，与 dispatch_a 的上取整形成互补而不是叠加（否则每个 2 的幂边界 m-grid 翻倍）； 3) 两阶段行调度（per-expert strided + 全局溢出共享）解决“平均值掩盖峰值”的经典调度问题，并用 row_cap slack 2x 在均匀负载代价（13-20%）与偏斜回归之间取平衡。阅读时建议对照 PR body 的性能表和提交历史中的 “share overflow rows”、“tile per lane” 两个关键演进提交。

功能与动机

PR body 明确指出：fp8_per_token_to_per_tensor_quant_triton() 是纯流式内核（每元素两次乘法、移动两个字节），但启动形状的三处属性比算术本身更贵——1024 元素 tile 配 8 warps 只有 4 B/thread，正好停在 Triton 停止向量化的阈值上，编译产物只有 ld.global.b32 / st.global.b32；scale 按元素轴寻址，在 DeepEP column-major 布局下形成每行重复 128 次的 8 地址 gather；m-grid 固定每专家 32 个 program，低延迟 decode 行数稀疏时大部分 program 只做 masked load 就退出——256 个本地专家、每专家 1 行时内核花 35 µs 移动约 3 µs 的数据。

实现拆解

1. 内核 tile 重构：从扁平元素到整 scale 组

python/sglang/kernels/ops/moe/ep_moe_kernels.py 中 _fp8_per_token_quant_to_per_tensor_quant_kernel 把 payload 从扁平元素段改为 [G_BLOCK_SIZE, 128] 的二维分块，K_BLOCK_SIZE 更名为 G_BLOCK_SIZE（constexpr 单位从隐藏元素数变成 128 宽 scale 组数）。每组的 scale 由一次标量 load 获得，快轴保持连续，部分 k 改为整组掩码。warps 由 8 降至 4，使每线程在飞字节从 4 B 升到 8 B，恢复 v2.b32 向量访问。

2. 两阶段行调度：row_cap + 全局溢出共享

新抽出 _requant_row 逐行写函数，phase 1 与 phase 2 共用，保证两条路径写出的行逐位一致。Phase 1 中每个专家按 m-grid 步长处理 min(masked_m, row_cap) 以下的行；Phase 2 通过 tl.cumsum 前缀和把扁平溢出索引映射回 (expert, row)，让 hot expert 的溢出行共享整

个 launch 的 program，避免均值估计（expected_m）掩盖偏斜路由、把单个专家串行化。均匀负载时溢出行数为 0，只付出一次归约开销。

3. 启动几何启发式：requant_launch_geometry

该函数返回 (G_BLOCK_SIZE, m_grid, row_cap) 三个 launch hint。关键设计：tile 以元素表达再除以 group 宽度，保证"每线程字节数"（真正的调优量）不被 group 宽度悄悄改变；expected_rows 向下取整到 2 的幂，因为 dispatch_a 上报 (rows + E) // E 已向上取整，二次上取整会在每个 2 的幂边界把 m-grid 翻倍；m-grid 上限保持历史 32、下限 4，行数低于 64 且专家轴未填满时按 1024 program 目标收缩；requant_warp_size 从 device props 读真实 warp 宽度（lru_cache 缓存），以适配 AMD wave64。

4. 调用链打通

expected_m 早已在 host 侧 dispatch_a 中算好、且对已捕获的 CUDA graph 恒定。本 PR 把它从 w4afp8.py 的 apply_deepep_ll 经 cutlass_w4a8_moe_deepep_ll 透传到 fp8_per_token_to_per_tensor_quant_triton 的 expected_rows，仅作 launch hint，正确性不依赖它。

5. 测试、基准与 CI 配套

- 新增 CPU 单元测试 test_w4afp8_requant_geometry.py（12 个用例）：锁定 m-grid 上界 / 单调性 / 向下取整、行 cap 契约、每线程字节数跨 warp 宽度不变、tile 不超 payload。
- 扩展 GPU 单元测试 test_fp8_per_token_to_per_tensor_quant.py：覆盖 column-major scale、40/128 专家 cap 生效、全 launch 几何组合逐位一致、空专家跳过。
- 新增 base-b-kernel-benchmark 条目 bench_fp8_per_token_to_per_tensor_quant.py：tuned 与 legacy-geometry 同图对比，含 skewed 路由轴。

关键文件：

- python/sglang/kernels/ops/moe/ep_moe_kernels.py（模块 MoE 内核；类别 source；类型 core-logic；符号 _requant_row, _floor_pow2, requant_warp_size, requant_launch_geometry）：核心变更文件：重构 requant Triton 内核的 tile 形状、warp 数与 m-grid，新增两阶段行调度和 launch geometry 启发式，是本 PR 全部性能收益的来源。
- test/registered/unit/layers/moe/test_w4afp8_requant_geometry.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestRequantLaunchGeometry, test_cap_leaves_ordinary_variation_to_the_owning_expert, test_cap_never_exceeds_the_payload, test_unknown_row_count_keeps_every_row_with_its_expert）：新增 CPU 启发式契约测试，12 个用例锁定 m_grid 上界 / 单调性 / 向下取整、row_cap 边界、每线程字节数跨 warp 宽度不变等关键不变量，是评审中反复打磨的对象。
- test/registered/kernels/ops/moe/test_fp8_per_token_to_per_tensor_quant.py（模块 单元测试；类别 test；类型 test-coverage；符号 _build, test_masked_rows_and_group_tail, _assert_output, _run_and_check）：GPU 正确性测试扩展：覆盖 column-major scale、40/128 专家 cap 生效、穷举 launch 几何组合逐位一致、空专家跳过，直接守护两阶段调度中最易 off-by-one 的前缀和映射。

- test/registered/kernels/benchmark/moe/bench_fp8_per_token_to_per_tensor_quant.py（模块 基准测试；类别 test；类型 test-coverage；符号 _expected_m, _row_counts, _build, _tuned）：新增 in-tree 基准，同图对比 tuned 与 legacy-geometry 两种 launch，并引入 skewed 路由轴覆盖 expected_m 均值掩盖的偏斜场景，是 CI 中防止性能回退的锚点。
- python/sglang/srt/layers/quantization/w4afp8.py（模块 量化层；类别 source；类型 core-logic；符号 apply_deepep_ll）：调用链入口：从 DeepEP dispatch output 中解出 expected_m 并透传给 cutlass_w4a8_moe_deepep_ll，改动虽小但打通了 host-side 行估计到 kernel 的链路。
- python/sglang/srt/layers/moe/cutlass_w4a8_moe.py（模块 MoE 算子；类别 source；类型 core-logic；符号 cutlass_w4a8_moe_deepep_ll）：把 expected_m 作为可选 launch hint 透传到 requant wrapper，并在 docstring 中明确“any value is correct”，强调正确性不依赖该估计。

关键符号：fp8_per_token_to_per_tensor_quant_triton, _fp8_per_token_quant_to_per_tensor_quant_kernel, _requant_row, requant_launch_geometry, requant_warp_size, _floor_pow2, cutlass_w4a8_moe_deepep_ll, apply_deepep_ll

关键源码片段

python/sglang/kernels/ops/moe/ep_moe_kernels.py

核心变更文件：重构 requant Triton 内核的 tile 形状、warp 数与 m-grid，新增两阶段行调度和 launch geometry 启发式，是本 PR 全部性能收益的来源。

```
# 调优单位是“每线程字节数”而非元素数：NVIDIA wave32 与 AMD wave64 的 warp 宽度不同，
# 固定元素数会在 wave64 部件上把每线程字节数减半。
_REQUANT_BYTES_PER_LANE = 16 # 专家数 >= 32 时每个线程搬 16 B
_REQUANT_BYTES_PER_LANE_FEW_EXPERTS = 8 # 专家少、grid 填不满时 tile 减半
_REQUANT_MANY_EXPERTS = 32
_REQUANT_NUM_WARPS = 4 # 4 warps 让每线程 8 B，恢复 v2.b32 向量化
_REQUANT_DEFAULT_WARP_SIZE = 32
_REQUANT_M_GRID_MAX = 32 # 历史固定值，新几何只缩不放
_REQUANT_M_GRID_MIN = 4 # 下限避免单行 batch 把 expert 串行化
_REQUANT_TARGET_PROGRAMS = 1024 # 行稀缺时 (m-grid x expert) 平面上的 program 目标
_REQUANT_ROWS_SATURATED = 64 # 超过此行数，cap 掉的 program 会有真实工作
_REQUANT_ROW_CAP_SLACK = 2 # row_cap = 2x 期望行，容忍专家间普通波动


def _floor_pow2(value: int) -> int:
    return 1 << (max(1, value).bit_length() - 1)


@lru_cache(maxsize=None)
def requant_warp_size(device: torch.device) -> int:
    """每个 warp 的 lane 数，决定 requant 的 tile 宽度。"""
    return torch.cuda.get_device_properties(device).warp_size
```

```
def requant_launch_geometry(
    num_groups: int,
    num_experts: int,
    group_size: int = 128,
    expected_rows: Optional[int] = None,
    warp_size: int = _REQUANT_DEFAULT_WARP_SIZE,
    max_rows: int = 1 << 30,
) -> Tuple[int, int, int]:
    """返回 (每组 program 的 group 数, m_grid, row_cap) 三个 launch hint。

    任何取值都产生相同的字节输出，几何只影响性能。行估计必须向下取整到 2 的幂：
    dispatch_a 上报 (rows + num_experts) // num_experts，在恰好平均时多报 1 行，
    若这里再向上取整，会在每个 2 的幂边界把 m-grid 翻倍。
    """
    # payload 是 fp8，一个元素就是一个字节，所以每线程字节数等于每线程元素数。
    bytes_per_lane = (
        _REQUANT_BYTES_PER_LANE
        if num_experts >= _REQUANT_MANY_EXPERTS
        else _REQUANT_BYTES_PER_LANE_FEW_EXPERTS
    )
    tile_elems = bytes_per_lane * _REQUANT_NUM_WARPS * warp_size
    # tile 用元素表达、除以 group 宽度得到 G_BLOCK_SIZE：不同 group 宽度不会
    # 悄悄改变实测的 tile 大小；再向下取整并 clamp 到 payload。
    g_block = min(_floor_pow2(tile_elems // group_size), _floor_pow2(num_groups))

    if expected_rows is None:
        # 没有行估计就没有 cap 的依据，保持历史 m-grid，所有行留在本专家。
        return g_block, _REQUANT_M_GRID_MAX, max_rows

    m_grid = min(
        _REQUANT_M_GRID_MAX,
        max(_REQUANT_M_GRID_MIN, _floor_pow2(expected_rows)),
    )
    if expected_rows < _REQUANT_ROWS_SATURATED:
        # 行稀缺时用 program 目标限制 m-grid，避免专家轴还没填满机器就空转。
        m_grid = min(
            m_grid, _floor_pow2(_REQUANT_TARGET_PROGRAMS // max(1, num_experts))
        )
    row_cap = min(max_rows, max(1, expected_rows * _REQUANT_ROW_CAP_SLACK))
    return g_block, m_grid, row_cap
```

test/registered/unit/layers/moe/test_w4afp8_requant_geometry.py

新增 CPU 启发式契约测试，12 个用例锁定 m_grid 上界 / 单调性 / 向下取整、row_cap 边界、每线程字节数跨 warp 宽度不变等关键不变量，是评审中反复打磨的对象。

```
class TestRequantLaunchGeometry(CustomTestCase):
    def test_m_grid_never_exceeds_the_previous_fixed_grid(self):
        """行估计只会让 m-grid 收缩，任何 batch 都不会比旧固定网格更差。"""
        for expected_rows in (1, 8, 32, 33, 1024):
```

```
            for num_experts in (8, 56, 256):
                _, m_grid, _ = requant_launch_geometry(
                    DSV3_GROUPS, num_experts, expected_rows=expected_rows
                )
                self.assertLessEqual(m_grid, PREVIOUS_FIXED_M_GRID) # 历史值 32

    def test_dispatcher_round_up_does_not_bump_the_grid(self):
        """dispatch_a 上报 (rows + E) // E，恰好整除时多报 1 行；启发式若也向上
        取整，会在每个 2 的幂边界把 m-grid 翻倍，所以必须向下取整。"""
        for rows in (4, 8, 16, 32):
            exact = requant_launch_geometry(DSV3_GROUPS, 8, expected_rows=rows)[1]
            reported = requant_launch_geometry(
                DSV3_GROUPS, 8, expected_rows=rows + 1
            )[1]
            self.assertEqual(reported, exact, f"rows={rows}")

    def test_tile_holds_bytes_per_lane_across_warp_widths(self):
        """调优单位是每线程字节数：warp 32 的 2048 元素在 warp 64 上必须变成
        4096，否则 wave64 部件拿到一半的每线程字节数。"""
        for warp_size, want_elems in ((32, 2048), (64, 4096)):
            for group_size in (64, 128, 256, 512):
                g_block, _, _ = requant_launch_geometry(
                    num_groups=7168 // group_size,
                    num_experts=56,
                    group_size=group_size,
                    expected_rows=16,
                    warp_size=warp_size,
                )
                self.assertEqual(
                    g_block * group_size, want_elems, f"warp_size={warp_size}"
                )
```

评论区精华

> BBuf: Could we benchmark with the actual expected_m from the dispatcher? An exact average of 8 rows produces 9, which changes m_grid from 8 to 16.

benchmark 的 _expected_m 改为模拟 dispatch_a 的 (rows + num_experts) // num_experts 上取整上报，使基准与生产 launch 对齐。

> BBuf: _legacy still runs the rewritten kernel, so it only restores the old launch parameters, not the old scale-loading behavior.

alexnails 接受并重命名，最终以 legacy-geometry 命名并在文档串中说明其只隔离 launch geometry，不包含 scale 加载方式的重写。

> BBuf: This case does not actually exercise the expert cap. With 128 experts and expected_rows=8, both paths choose m_grid=8.

测试参数改为 (40, 32) 与 (128, 16)，让 cap 真正绑定：40 专家封顶 16、128 专家封顶 8，同时把每专家行数改成互不相同。

> BBuf: expected_rows=None only preserves the old m-grid. The group block size and num_warps still change.

alexnails 承认描述过度，测试更名为 test_unknown_row_count_keeps_the_previous_m_grid，PR body 中同类表述一并修正——expected_rows=None 只保留旧 m-grid。

> BBuf: These constants are tuned on H200 and B200 but apply to all SM90+ GPUs. Do we have an H100 or H20 result as well?

alexnails 补测 H100，并刻意在 torch 2.11 / triton 3.6 与 torch 2.13 / triton 3.7.1 两套工具链上验证（收益依赖 Triton codegen）；H20 无设备，给出单命令复现入口。

> BBuf: expected_m is an average, so with skewed routing a hot expert may be left with m_grid=4/8 and serialize many rows.

benchmark 增加 skew 轴：一个 hot expert 拿 skew * rows，其余专家均分剩余，并保持 dispatch 总量（即 expected_m）不变。

> BBuf: The CI subset only runs E8, so it does not exercise the num_experts >= 32 tile/cap path.

CI 子集默认加入 (3584, 56, 256) 形状与 rows=32、skew=16 等代表组合。

- benchmark 应使用 dispatcher 实际上报的 expected_m (correctness): _expected_m 改为 (dispatched_rows + num_experts) // num_experts，模拟 dispatch_a 的上取整上报。
- legacy 基准的命名与基线有效性 (testing): 重命名为 legacy-geometry，并在文件 docstring 说明其只隔离 launch geometry 变化。
- test_many_experts 未真正触发 expert cap (testing): 参数化为 (40, 32) 与 (128, 16)，使 cap 真正绑定，并把 m 提高到 48、每专家行数互不相同。
- expected_rows=None“保持旧几何”的措辞误导 (documentation): 测试改名 test_unknown_row_count_keeps_the_previous_m_grid，PR body 中同类表述一并修正。
- 调优常量的硬件覆盖（H100/H20） (performance): H100 在 torch 2.11/triton 3.6 与 2.13/3.7.1 两套工具链补测，收益成立；H20 无设备，留 benchmark 单命令复现入口。
- K_BLOCK_SIZE 更名 G_BLOCK_SIZE 的原因 (question): 因为单位从隐藏元素数变为 128 宽 scale 组数，tile 现在按“组”计量。
- skewed 路由缺乏性能基准覆盖 (performance): benchmark 增加 skew 轴：一个 hot expert 拿 skew*rows、其余均分，并保持 dispatch 总量（即 expected_m）不变。
- CI benchmark 子集未覆盖大专家数路径 (testing): CI 子集加入 (3584, 56, 256) 代表形状，rows=32、skew=16 等组合进入默认参数表。

风险与影响

- 风险：

1. 性能回归风险（主要风险）：m-grid 启发式为经验拟合，PR 自述在低延迟 decode 区间距每格最优 0-5%，超高专家数 sub-4 μs 形状有 8-12% 离群，B200 在 8 行场景还有 0.03 μs 的微逆单元格；“只缩不放”设计（m-grid ≤ 历史 32）兜底了大部分 batch，但 tile 宽度与 warp 数对所有调用者无条件改变（即使 expected_rows=None），test_w4afp8_requant_geometry.py 明确锁定了这一契约。H20 未测，其余架构（尤其非 SM90 部件）常量可能非最优。
2. 正确性风险低但有边界：Phase 2 前缀和映射是最可能 off-by-one 的地方，测试已覆盖空专家、全零 masked_m、cap 与 payload 边界，三厂商输出 bitwise 一致；但 PR 自述多节点端到端 decode 仍未跑。
3. Triton codegen 版本敏感：收益依赖 8 B/thread 触发 v2.b32 的代码生成行为，triton 3.6 与 3.7.1 在 7168 形状上已出现 +12% → 持平的方向性差异，未来 Triton 升级可能改变当前最优几何（PR 自述 closing the remaining few percent 需要 autotuning）。 - 影响：用户侧：W4AFP8 + DeepEP low-latency 部署（DeepSeek-V3/R1、Kimi-K3 等 W4A8 模型）decode 阶段显著受益：E2E A/B 显示 4x H200、EP4 场景吞吐 +0.83~1.48%、median TPOT -0.84~1.44%；并发越低、本地专家越多收益越大（kernel 级单行 12.1x）。无接口、flag 或精度变化，输出 bitwise 一致，升级无需配置改动。 系统侧：新增 3 个测试 / 基准入口——CPU 几何契约测试（base-a-test-cpu，约 5 s）、GPU 单元测试（base-b-kernel-unit，est 60 s）、kernel benchmark（base-b-kernel-benchmark，est 45 s，1-gpu-large）。 团队侧：沉淀了可复用的 kernel 调优方法学——bytes-per-lane 调优单位、双向上取整抵消、均值估计与溢出共享调度组合；所有常量集中定义并带注释，为后续 autotune 留了明确接口。

- 风险标记：核心内核路径变更，启发式经验拟合，Triton codegen 版本敏感，H20 未覆盖，多节点 E2E 未验证

关联脉络

- PR #36985 test: re-enable FlashInfer per-token NVFP4 coverage: 未改相同文件，但同属 W4A8/NVFP4 MoE 量化内核的正确性与性能保障生态，可视为该领域测试矩阵持续扩展的背景。