PR #30924 完整报告

sgl-project/sglang

[JIT] Trait-driven per_token_group_quant: unify the quant kernel family (flat + masked)

合并时间：2026-07-22 08:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30924

执行摘要

- 一句话：统一 per_token_group_quant 多种实现为 trait-driven JIT kernel
- 推荐动作：此 PR 值得精读，尤其是 trait-driven kernel 的设计和性能调优细节（32B/lane tiling、FMUL2、PDL）。合并前应修复 Hopper 性能回退和负索引安全问题。如你正在使用 Hopper GPU，建议暂缓升级此 PR 的 commit。

功能与动机

per_token_group_quant 家族积累了多个并行实现（AOT、JIT、Triton），维护负担重且性能不足。本 PR 引入一个 trait-driven JIT kernel，统一所有 variant，消除重复代码，并利用编译时特化优化性能。

实现拆解

1. 定义 QuantTrait 编译时参数：在 per_token_group_quant.cuh 中，将输入 / 输出 dtype、group size、scale 格式（fp32/UE8M0）、scale 布局（行 / 列主序）、融合 silu_and_mul、调度模式（flat/masked）作为编译时 trait 参数，同一 kernel body 通过模板特化生成所有变体。
2. 新增 Python 入口 per_token_group_quant.py：实现 _jit_module 根据 trait 参数调用 load_jit 编译并加载对应的 CUDA kernel；_infer_scale_layout 从输出 tensor 的 stride 推断 scale 布局；_per_token_group_quant_custom_op 作为 custom op 入口，支持 masked_m 和 expected_m；_allocate_outputs 负责根据配置分配输出 tensor。
3. 删除旧 kernel 及调度器：删除 per_token_group_quant_8bit.py 及其 CUDA 后端 per_token_group_quant_8bit.cuh，移除运行时调度器 _run_per_token_group_quant_8bit_kernel 中的 v1 分支和环境变量 SGLANG_OPT_USE_JIT_PER_TOKEN_GROUP_QUANT。
4. 迁移所有调用点：修改 deep_gemm.py，将 silu_and_mul_masked_post_quant_fwd 统一为新 kernel，移除 SGLANG_MASKED_GEMM_FAST_ACT 分支；简化 fp8_kernel.py 中的 Triton 回退和 AOT 路由；_varlen_deep_gemm_silu_mul_quant 现在直接调用 per_token_group_quant。
5. 新增正确性测试和性能基准：test_per_token_group_quant.py 包含 92 个用例，UE8M0 路径与纯 torch 参考对比 bit-exact，fp32 路径验证 dequant 误差；新增 bench_per_token_group_quant.py 和 bench_per_token_group_quant_masked.py 用于

性能回归。旧测试和 benchmark 被删除。

关键文件：

- python/sglang/jit_kernel/per_token_group_quant.py（模块 量化内核；类别 source；类型 core-logic；符号 _jit_module, _infer_scale_layout, _per_token_group_quant_custom_op, _allocate_outputs）：核心 Python 入口，负责根据 trait 参数加载 JIT kernel 并统一调度，所有调用点最终汇聚于此。
- python/sglang/jit_kernel/csrc/gemm/per_token_group_quant.cuh（模块 CUDA 内核；类别 other；类型 core-logic）：CUDA kernel 实现，包含 QuantTrait 模板和 flat/masked 调度逻辑，是性能关键所在。
- python/sglang/jit_kernel/per_token_group_quant_8bit.py（模块 旧量化内核；类别 source；类型 deletion；符号 _jit_per_token_group_quant_8bit_module, _per_token_group_quant_8bit_custom_op, per_token_group_quant_8bit）：旧 kernel 入口，被彻底删除，代表本 PR 的核心重构行动。
- test/registered/jit/test_per_token_group_quant.py（模块 测试套件；类别 test；类型 test-coverage；符号 _group_amax, _quantize, ref_fp8_fp32_scale, ref_int8）：92 个正确性测试，使用纯 torch 参考，确保新 kernel 在所有变体上的行为正确，UE8M0 路径 bit-exact。
- python/sglang/srt/layers/moe/moe_runner/deep_gemm.py（模块 MoE 运行器；类别 source；类型 dependency-wiring；符号 silu_and_mul_masked_post_quant_fwd, _varlen_deep_gemm_silu_mul_quant）：主要调用点迁移，将 silu_and_mul 与 quant 整合，移除旧分支，体现架构简化。
- test/registered/jit/benchmark/bench_per_token_group_quant_masked.py（模块 性能基准；类别 test；类型 test-coverage；符号 _jit_v2, _current, benchmark）：新增 masked 调度 benchmark，验证低 token 加速效果（16.4x @ 1 token），覆盖 DeepSeek-V3/V4 等模型。

关键符号：per_token_group_quant, _jit_module, _infer_scale_layout, _per_token_group_quant_custom_op, _allocate_outputs, per_token_group_quant_8bit, per_token_group_quant_8bit_v2, ref_fp8_ue8m0, ref_fp8_fp32_scale, ref_int8, silu_and_mul_masked_post_quant_fwd

关键源码片段

python/sglang/jit_kernel/per_token_group_quant.py

核心 Python 入口，负责根据 trait 参数加载 JIT kernel 并统一调度，所有调用点最终汇聚于此。

```
# python/sglang/jit_kernel/per_token_group_quant.py
# 根据编译时 trait 加载 JIT kernel

@cache_once
def _jit_module(
    in_dtype: torch.dtype,
    out_dtype: torch.dtype,
```

```
    group_size: int,
    scale_ue8m0: bool, # UE8M0 指数编码 vs FP32 浮点 scale
    row_major: bool, # scale 布局：行主序 vs 列主序
    aligned: bool, # 组数是否为 4 的倍数（仅 UE8M0 有效）
    fuse_silu_and_mul: bool,
    masked_layout: bool, # 是否启用 masked 调度（EP-MoE）
    use_pdl: bool, # 是否使用 PDL（programmatic dependent launch）
) -> Module:
    # 验证输入合法性
    assert in_dtype in _SUPPORTED_INPUT_DTYPES
    assert out_dtype in _SUPPORTED_OUTPUT_DTYPES
    assert group_size in _SUPPORTED_GROUP_SIZES
    # 将 Python 参数转换为 C++ 模板参数
    trait_args = make_cpp_args(
        in_dtype, out_dtype, group_size, scale_ue8m0,
        row_major, aligned, fuse_silu_and_mul, use_pdl,
    )
    # 根据调度模式选择不同的 kernel 启动器
    launcher = (
        "PerTokenGroupQuantMaskedKernel"
        if masked_layout else "PerTokenGroupQuantFlatKernel"
    )
    return load_jit(
        "per_token_group_quant", *trait_args,
        "masked" if masked_layout else "flat",
        cuda_files=["gemm/per_token_group_quant.cuh"],
        cuda_wrappers=[("per_token_group_quant",
                        f"{launcher}<{trait_args}>::run")],
        extra_cuda_cflags=["--use_fast_math"],
    )
```

test/registered/jit/test_per_token_group_quant.py

92 个正确性测试，使用纯 torch 参考，确保新 kernel 在所有变体上的行为正确，UE8M0 路径 bit-exact。

```
# test/registered/jit/test_per_token_group_quant.py
# UE8M0 路径的纯 torch 参考（用于 bit-exact 对比）


def ref_fp8_ue8m0(x, gs):
    """生成 fp8 码点和 UE8M0 指数字节（[..., ng]）。
    乘数 2^-e 在 fp32 中精确表示，因此码点与参考 bit-exact。
    """
    amax = _group_amax(x, gs) # 每组的 absmax，不低于 EPS
    raw = (amax / FMAX).contiguous() # FMAX = 448
    bits = raw.view(torch.int32)
    # 向上取整为 UE8M0 指数：若尾数非零则指数 +1
    exp = ((bits >> 23) & 0xFF) + ((bits & 0x7FFFFF) != 0).to(torch.int32)
    # 构造量化 scale = 2^(127 - (exp-127)) = 2^(254 - exp)
    quant_scale = ((127 + 127 - exp) << 23).view(torch.float32)
```

```
    q = _quantize(x, gs, quant_scale, fp8_dtype, -FMAX, FMAX)
    return q, exp.to(torch.uint8)
```

评论区精华

Gemini Code Assist 负索引安全审查：在 per_token_group_quant.cuh 中三处使用 -1、-2 作为 TensorView::size() 和 stride() 参数（通常接受 size_t），可能产生未定义行为。建议使用 ndim() - 1 替代。此问题在合并前未修复。

用户报告 Hopper H20 性能回退：用户 whybeyoung 反馈新 kernel 导致 GLM5.2 性能下降，并提供了 revert commit 链接。作者 DarkSharpness 表示仅一次 Hopper 测试无回归，但未深入调查。该问题在合入前未完全解决。

- 负索引安全性问题 (security): 作者未在合并前修复该问题。
- Hopper H20 性能回退 (performance): 未解决，用户通过 revert 恢复性能。

风险与影响

- 风险：
  - Hopper 性能回退风险：用户报告 H20 上性能下降，PR 未充分验证 Hopper，可能影响 Hopper 集群用户。
  - 负索引安全性风险：三处负索引传入 TensorView 方法，参数类型为 size_t，会导致极大无符号数，可能引起越界访问或 UB。PR 合并前未修复。
  - 旧代码删除不可回滚：v1/v2 kernel 及配套测试、benchmark 被彻底删除，若新 kernel 出现未覆盖的边界情况，恢复旧行为需 revert 大量改动。
  - int8 码位差异：约 0.2% 的元素与旧 AOT 有 1 的码位偏移（由 --use_fast_math 除法边界引起），但 scale 一致，影响可控。
  - 影响：用户：B200 用户获得显著性能提升（flat 最高 27%，masked 低 token 加速 10x+）；Hopper 用户可能面临性能回退。系统：代码库量化 kernel 数量从 7+ 减少到 1，维护成本降低；调用点 API 简化，不再需要传入 eps/mix/max。团队：旧 API（如 per_token_group_quant_8bit）被移除，依赖旧 API 的分支需迁移；未来新增量化变体只需扩展 trait 参数，无需重复实现 kernel。
- 风险标记：Hopper 性能回退风险，负索引未修复，旧代码删除不可回滚，int8 码位差异

关联脉络

- PR #30838 [JIT] Refactor dtype traits into DTypeTrait and unify warp reductions: 本 PR 依赖于该 PR 提供的 DTypeTrait 和 warp reduction 原语。
- PR #30784 RFC Phase 2.5: Move quantization kernels to sglang.kernels.ops.quantization: 本 PR 在此之上 rebase，确保所有入口路径一致。