PR #27896 完整报告

sgl-project/sglang

[Perf] Skip per-call mat_a/scales_a padding in cutlass FP8 blockwise GEMM

合并时间：2026-06-13 00:01

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27896

执行摘要

- 一句话：消除 FP8 blockwise GEMM 每调用填充开销
- 推荐动作：推荐精读该 PR，特别是 sglang_per_token_group_quant_fp8_row_padded 的实现与注释。它清晰地表达了优化动机、正确性论证（行独立性保证）以及 fallback 策略。测试文件 test_fp8_blockwise_row_padding.py 展示了如何通过对比 legacy 路径保证位精确性，值得参考。

功能与动机

在 decode-heavy 负载（Qwen3.5-122B-A10B-FP8, TP=4, NEXTN 推测解码）中，每步 GEMM 的 kernel 序列为 group_quant -> fill -> cat -> fill -> cat -> copy -> copy -> cutlass_fp8_gemm，其中两次 fill 和两次 cat 完全来自 fp8_blockwise_scaled_mm 内部对 mat_a 和 scales_a 的 4 行对齐填充。由于激活张量在每个调用时重新分配，因此 padding 无法跨调用复用。PR 作者分析并提出在量化步骤提前分配对齐缓冲区，使填充在 wrapper 层短路。

实现拆解

实现分为以下步骤：

1. 新增行对齐量化函数（python/sglang/srt/layers/quantization/fp8_kernel.py）：添加 sglang_per_token_group_quant_fp8_row_padded，该函数将量化输出缓冲区分配为 4 行对齐（m_pad = ceil_align(m, 4)），然后对前 m 行运行 v2 量化核，尾行保持未初始化。当 v2 核不可用（group_size 不在 {16,32,64,128}）时 fallback 到原有 sglang_per_token_group_quant_fp8，确保兼容性。
2. 修改 GEMM 封装函数（python/sglang/srt/layers/quantization/fp8_utils.py）：cutlass_w8a8_block_fp8_linear_with_fallback 将量化调用替换为新的行对齐版本，并在 GEMM 输出后根据原始行数切片去除填充行（output = output[: input_2d.shape[0]]）。同时更新导入以包含新函数。
3. 新增单元测试（test/registered/quant/test_fp8_blockwise_row_padding.py）：覆盖 M ∈ {1,2,3,4,5,7,13,16,31,64,256}，验证行对齐缓冲区形状正确性、行对齐路径与 legacy 路径的位精确性，以及 FP8 线性结果与 bf16 参考的近似一致性。

关键文件：

- test/registered/quant/test_fp8_blockwise_row_padding.py（模块 量化测试；类别 test；类型 test-coverage；符号 _quant_weight_blockwise, _legacy_cutlass_linear, TestFP8BlockwiseRowPadding, setUpClass）：新增单元测试，覆盖多种 M 值，验证行对齐缓冲区的形状、位精确性和与 bf16 参考的近似一致性，是质量保障关键。
- python/sglang/srt/layers/quantization/fp8_kernel.py（模块 量化核心；类别 source；类型 core-logic；符号 sglang_per_token_group_quant_fp8_row_padded）：核心新增函数 sglang_per_token_group_quant_fp8_row_padded 所在，实现行对齐量化缓冲区分配，是性能优化的关键。
- python/sglang/srt/layers/quantization/fp8_utils.py（模块 量化工具；类别 source；类型 dependency-wiring；符号 cutlass_w8a8_block_fp8_linear_with_fallback）：修改了 cutlass_w8a8_block_fp8_linear_with_fallback 函数，集成行对齐量化并添加输出切片逻辑，是变更的集成点。

关键符号：sglang_per_token_group_quant_fp8_row_padded, cutlass_w8a8_block_fp8_linear_with_fallback

关键源码片段

python/sglang/srt/layers/quantization/fp8_kernel.py

核心新增函数 sglang_per_token_group_quant_fp8_row_padded 所在，实现行对齐量化缓冲区分配，是性能优化的关键。

```
# 位于 python/sglang/srt/layers/quantization/fp8_kernel.py
# 新增函数：将量化输出分配为行对齐缓冲区，使 GEMM wrapper 的 pad_tensor 短路

def sglang_per_token_group_quant_fp8_row_padded(
    x: torch.Tensor,
    group_size: int,
    eps: float = 1e-10,
    row_alignment: int = 4,
) -> Tuple[torch.Tensor, torch.Tensor]:
    """Per-token-group quant writing into row-padded buffers (col-major scales)."""
    assert x.dim() == 2, "row-padded quant expects a 2D input"
    assert (
        x.shape[-1] % group_size == 0
    ), "the last dimension of `x` must be divisible by `group_size`"
    assert x.is_contiguous(), "`x` is not contiguous"

    # 如果 v2 核不可用，回退到原有非对齐路径，让 GEMM wrapper 自行填充
    if not (enable_sgl_per_token_group_quant_8bit and group_size in (16, 32, 64, 128)):
        return sglang_per_token_group_quant_fp8(
            x, group_size, eps, column_major_scales=True
        )

    m, k = x.shape
    m_pad = ceil_align(m, row_alignment) # 对齐后的行数
    # mat_a 缓冲区 : (m_pad, k) 行主序
```

```
    x_q = torch.empty((m_pad, k), device=x.device, dtype=fp8_dtype)
    # scales_a 缓冲区：列主序（步长 0 为 1），形状 (m_pad, k // group)
    # 先分配 (k//group, m_pad) 再转置，保证 stride(0)==1 以满足 kernel 契约
    x_s = torch.empty(
        (k // group_size, m_pad), device=x.device, dtype=torch.float32
    ).transpose(0, 1)

    if m > 0:
        # 只对前 m 行执行量化（v2 核）
        sgl_per_token_group_quant_8bit(
            x,
            x_q[:m],
            x_s[:m],
            group_size,
            eps,
            fp8_min,
            fp8_max,
            False, # scale_ue8m0
            False, # fuse_silu_and_mul
            None, # masked_m
            enable_v2=True,
        )
    return x_q, x_s
```

python/sglang/srt/layers/quantization/fp8_utils.py

修改了 cutlass_w8a8_block_fp8_linear_with_fallback 函数，集成行对齐量化并添加输出切片逻辑，是变更的集成点。

```
# 位于 python/sglang/srt/layers/quantization/fp8_utils.py
# 修改后的 cutlass_w8a8_block_fp8_linear_with_fallback 函数

def cutlass_w8a8_block_fp8_linear_with_fallback(
    input: torch.Tensor,
    weight: torch.Tensor,
    block_size: List[int],
    weight_scale: torch.Tensor,
    input_scale: Optional[torch.Tensor] = None,
    bias: Optional[torch.Tensor] = None,
) -> torch.Tensor:
    assert input_scale is None

    # 仅当形状满足条件时才走 cutlass 路径，否则回退到 Triton
    shape_supported = weight.shape[0] % 128 == 0 and weight.shape[1] % 128 == 0
    if not shape_supported:
        return triton_w8a8_block_fp8_linear(
            input, weight, block_size, weight_scale, input_scale, bias
        )

    input_2d = input.view(-1, input.shape[-1])
```

```
    output_shape = [*input.shape[:-1], weight.shape[0]]

    # 使用行对齐量化，使 sgl-kernel wrapper 的 pad_tensor 短链（每 GEMM 省 2 fill + 2 cat）
    # weight_scale.T 仍保持 K 主序视图，因为 kernel 内部会做其自有拷贝
    q_input, x_scale = sglang_per_token_group_quant_fp8_row_padded(
        input_2d, block_size[1]
    )
    output = fp8_blockwise_scaled_mm(
        q_input, weight.T, x_scale, weight_scale.T, out_dtype=input_2d.dtype
    )
    if output.shape[0] != input_2d.shape[0]:
        # GEMM 是在填充后的缓冲区上执行的，丢弃多余行
        output = output[: input_2d.shape[0]]
    if bias is not None:
        output += bias
    return output.to(dtype=input_2d.dtype).view(*output_shape)
```

评论区精华

review 中 gemini-code-assist[bot]指出新增函数的断言错误消息写反了：条件检查 `x.shape[-1] % group_size == 0` 表示“必须被整除”，但消息写成了“cannot be divisible by group_size”。作者 yuan-luo立即回复 “Good catch, fixed.” 并修正了消息文本。

- 断言错误消息纠正 (correctness): 作者 yuan-luo 确认并修正了消息文本。

风险与影响

- 风险：
  1. 回归风险：当 v2 量化核不可用时，sglang_per_token_group_quant_fp8_row_padded 会 fallback 到原有路径，行为完全不变；当 v2 核可用时，新路径产生与 legacy 路径位精确相同的结果（测试验证了 max_abs_diff == 0）。
  2. 安全风险：填充行未初始化，但 GEMM 后立即切片丢弃，不会被读取；硬件行独立性保证了不对前 m 行产生副作用。
  3. 兼容性：仅影响 cutlass_w8a8_block_fp8_linear_with_fallback 调用链；Triton 和 DeepGEMM 路径未修改。
  4. 性能风险：额外分配对齐缓冲区可能略微增加显存峰值，但解码场景 M 很小，影响可忽略。 - 影响：对 用户：使用 cutlass FP8 blockwise 线性路径且满足 v2 量化核条件的模型（如 Qwen-FP8 系列）在解码阶段可观测到约 2% 的吞吐提升，单请求延迟无变化。对 系统：显著降低了 GPU 内核启动数（fill+cat 从 ~130k 降至 ~600），可能略微降低 GPU 利用率。对 团队：提供了一个“提前分配避免后续填充”的优化模式，可供其他类似场景参考。测试用例可作为 FP8 量化测试的模板。 - 风险标记：依赖 v2 量化核可用性，新增路径中未初始化尾行需注意

关联脉络

- 暂无明显关联 PR