PR #36237 完整报告

sgl-project/sglang

[MegaMoE] Respect padded MXFP8 scale row strides in pre-dispatch

合并时间：2026-08-25 15:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36237

执行摘要

- 一句话：修复 MegaMoE pre-dispatch 缩放行步长填充问题
- 推荐动作：该 PR 值得精读，特别是内核地址计算和校验逻辑，展示了处理外部库非连续布局的通用方案。建议关注后续 DeepGEMM 是否暴露物理步长，以及是否可简化。

功能与动机

在 DeepGEMM 中，MXFP8 scale 输出每个 token 行会为 TMA 对齐填充到 16 字节的倍数。原 pre-dispatch 内核假设 scale 行是紧凑的（连续布局），导致当 H 不是 512 的倍数时（如 H=2304），第二个 token 的 scale 行会写在错误的位置，覆盖到 padding 或错误的数据。PR 描述提到“Packed addressing misplaces scale rows after token 0”，并给出了具体 padding 例子。

实现拆解

1. 修改内核参数结构：在 MegaMoEPreDispatchParams 中新增 uint64_t buf_x_sf_stride_bytes 字段，用于传递物理行步长（字节）。
2. 修改内核写 scale 逻辑：在 mega_moe_pre_dispatch_kernel 中，将原先紧凑地址 token_id * num_groups + group_id 改为 static_cast<uint64_t>(token_id) * params.buf_x_sf_stride_bytes + group_id，以正确跳过填充。
3. 修改 TensorMatcher 校验：将 buf_x_sf 的 stride 从默认连续改为 with_strides({-1, 1})，允许第一维有填充，但要求内层维度连续。
4. 增加运行时校验：在 MegaMoEPreDispatchKernel 中新增对 buf_x_sf.stride(0) 的逻辑宽度校验、16 字节对齐校验（行步长和 base 地址）。
5. 新增测试：在 test_mega_moe_pre_dispatch.py 中新增 test_mxfp8_scale_output_uses_padded_row_stride，构造带填充的 scale 缓冲，验证逻辑 scale 字节被写入且 padding 和未使用行不被覆盖。

关键文件：

- python/sglang/kernels/jit/csrc/deepseek_v4/mega_moe_pre_dispatch.cuh（模块 内核；类别 other；类型 core-logic）：核心逻辑修复，涉及内核参数、地址计算和校验逻辑
- test/registered/kernels/ops/moe/test_mega_moe_pre_dispatch.py（模块 预调度；类别 test；类型 test-coverage；符号 test_mxfp8_scale_output_uses_padded_row_stride）：新增回归测试，验证填充行步长下 scale 写入正确

关键符号：mega_moe_pre_dispatch_kernel, MegaMoEPreDispatchKernel::operator(), test_mxfp8_scale_output_uses_padded_row_stride

关键源码片段

python/sglang/kernels/jit/csrc/deepseek_v4/mega_moe_pre_dispatch.cuh

核心逻辑修复，涉及内核参数、地址计算和校验逻辑

```
struct MegaMoEPreDispatchParams {
    const float* __restrict__ topk_weights; // [num_tokens, top_k]
    fp8_e4m3_t* __restrict__ buf_x; // [padded_max, hidden]
    int32_t* __restrict__ buf_x_sf; // row-major int32 [P, G/4]，可能带填充
    int64_t* __restrict__ buf_topk_idx; // [padded_max, top_k]
    float* __restrict__ buf_topk_weights; // [padded_max, top_k]
    uint64_t buf_x_sf_stride_bytes; // 物理行步长（字节），用于跳转填充
    uint32_t num_tokens;
    uint32_t padded_max;
    uint32_t hidden;
    // ...
};

// 写 scale 值的核心逻辑：
// 每个线程组写入一个 UE8M0 字节，使用物理步长定位行起始。
const uint32_t group_id = chunk / kThreadsPerGroup;
const uint32_t within_group_id = chunk % kThreadsPerGroup;
if (within_group_id == 0 && group_id < params.num_groups) {
    // 使用 buf_x_sf_stride_bytes 计算字节偏移，跳过每行的填充部分
    const uint64_t byte_off = static_cast<uint64_t>(token_id) * params.buf_x_sf_stride_bytes +
    group_id;
    reinterpret_cast<uint8_t*>(params.buf_x_sf)[byte_off] = static_cast<uint8_t>(ue8m0_exp);
}
```

test/registered/kernels/ops/moe/test_mega_moe_pre_dispatch.py

新增回归测试，验证填充行步长下 scale 写入正确

```
import torch
from sglang.kernels.ops.attention.dsv4 import mega_moe_pre_dispatch

# 使用 H=2304，物理行 80B，逻辑行 72B，产生 8B 填充
num_tokens, padded_max, hidden, top_k = 5, 8, 2304, 8
num_groups = hidden // 32 # 72
logical_scale_int32 = num_groups // 4 # 18
scale_stride_int32 = 20 # 物理行 80B / 4B
marker = 0xA5

# 构造带填充的 scale 缓冲，填充部分用 marker 标记
scale_bytes = torch.full(
    (padded_max, scale_stride_int32 * 4), marker,
    device="cuda", dtype=torch.uint8)
```

```
buf_x_sf = scale_bytes.view(torch.int32)[:, :logical_scale_int32]
# 验证步长为 20
assert buf_x_sf.stride() == (scale_stride_int32, 1)

# 调用内核
mega_moe_pre_dispatch(x, topk_idx, topk_weights, buf_x, buf_x_sf,
                      buf_topk_idx, buf_topk_weights)
torch.cuda.synchronize()

# 验证：逻辑字节部分已被写入（非 marker），填充部分保持 marker
logical_scale_bytes = logical_scale_int32 * 4
assert torch.all(scale_bytes[:num_tokens, :logical_scale_bytes] != marker)
assert torch.all(scale_bytes[:num_tokens, logical_scale_bytes:] == marker)
assert torch.all(scale_bytes[num_tokens:] == marker)
```

评论区精华

本 PR 没有 review 评论。主要讨论体现在 PR 描述中，关于布局的详细解释和需求，以及原 PR #36007 被替换的原因（CI rebase 门禁阻塞）。

- 暂无高价值评论线程

风险与影响

- 风险：该改动影响 Megatron MoE pre-dispatch 内核，如果错误可能导致 scale 数据错位或越界，但增加了校验。主要风险包括：
  - 新校验可能拒绝合法但未对齐的布局（但符合 DeepGEMM 要求）。
  - 使用 stride(0) 而非物理步长字节可能引入类型转换错误，但代码中乘以 sizeof(int32_t)。
  - 测试仅在 Blackwell GPU 上运行，其他平台兼容性未验证。
  - 影响：影响范围集中在 DeepSeek-V4 等使用 MegaMoE pre-dispatch 的模型，修复在部分隐藏维度下 scale 错位问题，提升正确性。对性能影响极小，因为只是地址计算变化。对用户而言，能解决潜在的错误结果。团队需要保持与 DeepGEMM 布局变化同步。
  - 风险标记：核心路径变更，缺少测试覆盖（其他平台），外部库布局耦合

关联脉络

- 暂无明显关联 PR