PR #36456 完整报告

sgl-project/sglang

Fix OOB read in mxfp4 MoE weight scales on Hopper

合并时间：2026-08-27 04:36

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36456

执行摘要

- 一句话：修复 Hopper 上 mxfp4 权重缩放越界读取问题
- 推荐动作：建议精读。该 PR 展示了如何通过简洁的填充修复一个由上游 kernel 未掩码加载导致的隐蔽越界读取问题，并附有详细的根因分析、性能对比和验证数据，值得深入了解其调试思路和修复策略。代码中也清晰标记了上游修复的退役条件。

功能与动机

PR 描述中明确指出 test_gpt_oss_4gpu_mxfp4.py 在 4-gpu-h100 上间歇性失败，原因是服务器在评估中途崩溃，所有请求返回空结果，导致得分恰好为 0.0（低于阈值 0.58）。根因是 triton_kernels 在 Hopper 上强制 block_k=128，其 HOPPER_SCALE 分支加载权重缩放时未掩码，导致K轴越界读取。该问题仅影响最后一个专家且依赖分配器布局，因此表现为间歇性。

实现拆解

1. 新增填充助手函数：在 python/sglang/srt/layers/quantization/mxfp4.py 中新增 _pad_hopper_mxfp4_scale(scale, k_size) 函数，计算目标长度 round_up(k_size, 128) // 32，若当前长度不足则使用 torch.nn.functional.pad 填充，填充值为中性 E8M0 缩放值 _UE8M0_ONE。
2. 集成到 swizzle 流程：在 _swizzle_mxfp4 中，对于 SM90（Hopper）分支，在转置和布局转换之前根据 quant_tensor.shape[-1] * 2（即 K 大小，因为 fp4 每字节打包 2 个值）调用填充函数，确保缩放张量的长度满足 kernel 的访问需求。
3. 不涉及其他模块：该改动仅影响权重加载后的缩放处理，权重加载器、create_weights 形状均未改动，因此对非 Hopper 平台和 w2 矩阵无影响。未增加单独测试，但通过现有 e2e 测试和 compute-sanitizer 验证。

关键文件：

- python/sglang/srt/layers/quantization/mxfp4.py（模块 量化层；类别 source；类型 core-logic；符号 _pad_hopper_mxfp4_scale）：核心修复文件，新增了 _pad_hopper_mxfp4_scale 函数并集成到 _swizzle_mxfp4 中，解决了 Hopper 上 mxfp4 权重缩放越界读取问题。

关键符号：_pad_hopper_mxfp4_scale

关键源码片段

python/sglang/srt/layers/quantization/mxfp4.py

核心修复文件，新增了 _pad_hopper_mxfp4_scale 函数并集成到 _swizzle_mxfp4 中，解决了 Hopper 上 mxfp4 权重缩放越界读取问题。

```
# python/sglang/srt/layers/quantization/mxfp4.py
def _pad_hopper_mxfp4_scale(scale, k_size):
    # triton_kernels 的 HOPPER_SCALE 分支（matmul_details/_matmul.py）会以 cdiv(k_size, 128)
    # 个 tile 的方式非掩码地加载权重缩放，因此需要将 K 轴长度填充到 128 的倍数。
    # 这里填充为 `round_up(k_size, 128) // 32` 个元素（每个 mxfp4 块 32 个缩放值）。
    mxfp4_block = 32
    want = round_up(k_size, 128) // mxfp4_block
    # 如果已有长度足够则直接返回，避免不必要的内存复制。
    if scale.shape[-1] >= want:
        return scale
    # 使用 `_UE8M0_ONE`（中性 E8M0 缩放值）填充，数值上对结果无影响。
    return torch.nn.functional.pad(scale, (0, want - scale.shape[-1]), value=_UE8M0_ONE)


def _swizzle_mxfp4(quant_tensor, scale, num_warps):
    """weight swizzle for mxfp4 moe, used for OAI mxfp4 kernel"""
    import triton_kernels.matmul_details.opt_flags as opt_flags
    from triton_kernels.numerics import InFlexData
    from triton_kernels.tensor import FP4, convert_layout, wrap_torch_tensor
    from triton_kernels.tensor_details import layout

    value_layout = layout.make_default_matmul_mxfp4_w_layout(mx_axis=-2)
    value_layout_opts = {}
    scale_layout = layout.make_default_matmul_mxfp4_w_scale_layout(
        mx_axis=-2, num_warps=num_warps
    )
    scale_layout_opts = {}
    if is_sm100_supported():
        constraints = {
            "is_persistent": True,
            "epilogue_subtile": 1,
        }
        opt_flags.update_opt_flags_constraints(constraints)
    elif is_sm90_supported():
        # Hopper 分支：设置 split_k=1，并在布局转换前对 scale 进行填充，避免越界读取。
        constraints = {
            "split_k": 1,
        }
        opt_flags.update_opt_flags_constraints(constraints)
        k_size = quant_tensor.shape[-1] * 2 # packed e2m1: 每个字节包含 2 个 fp4 值
        scale = _pad_hopper_mxfp4_scale(scale=scale, k_size=k_size)
    # 转置张量使量化轴位于第 1 维，然后进行布局转换。
    quant_tensor = quant_tensor.transpose(-2, -1)
    scale = scale.transpose(-2, -1)
```

```
    quant_tensor = convert_layout(
        wrap_torch_tensor(quant_tensor, dtype=FP4), value_layout, **value_layout_opts
    )
    scale = convert_layout(wrap_torch_tensor(scale), scale_layout, **scale_layout_opts)
    return quant_tensor, InFlexData(), scale
```

评论区精华

无 review 评论，仅有一个 /rerun-test 命令，由 CI bot 确认测试通过。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。虽然填充会增加约 14 MB/GPU 的内存占用，但数值上填充值为中性 E8M0 缩放值，且与对应权重 w 的掩码一致，因此不会影响正确性。主要风险在于该修复仅针对 Hopper（SM90），对 SM100（Blackwell）的未掩码加载问题（BLACKWELL_SCALE 分支）未处理，但当前 4-gpu-b200 测试通过。此外，填充逻辑依赖 triton_kernels 的 compute_block_k 强制 128 的行为，如果上游修复该问题，此填充可能会变得冗余，但不会造成错误。
- 影响：影响范围集中在使用 mxfp4 MoE 在 Hopper（如 H100/H200）上运行的模型（如 gpt-oss-120b）。修复后，相关 *e2e* 测试从随机崩溃（错误 0.0）变为稳定通过（得分 0.63），大幅提升该类部署的稳定性。对非 Hopper 平台无影响，对内存占用有轻微增加，但可忽略。团队可避免在 H100 上遇到间歇性服务器崩溃问题，提高生产环境的可靠性。
- 风险标记：Hopper 专属修复，依赖上游 kernel 行为，内存占用轻微增加

关联脉络

- 暂无明显关联 PR