PR #23319 完整报告

sgl-project/sglang

[AMD] Use bpreshuffle FP8 blockscale GEMM to replace ABScale GEMM

合并时间：2026-04-23 16:51

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23319

执行摘要

- 一句话：用 bpreshuffle GEMM 替换 ABScale GEMM，提升 AMD FP8 块缩放性能。
- 推荐动作：本 PR 值得精读。其核心设计模式"一次性预处理（preshuffle）换取每次推理加速"在算子优化中具有代表性。AMD 平台性能验证充分，但需关注精度回归是否在可接受范围内。建议关注未来 Triton 路径扩展后，bpreshuffle 路径的覆盖变化。

功能与动机

AITER 的 gemm_a8w8_blockscale_bpreshuffle 是更快的 FP8 块缩放 GEMM，需要预先洗牌权重。通过在加载时完成一次洗牌，将布局转换成本前置，从而加速每一次前向传播。本 PR 旨在为 AMD 平台引入该优化，提升整体推理性能。

实现拆解

1. 导入新算子（python/sglang/srt/layers/quantization/fp8_utils.py）：将原来的 gemm_a8w8_blockscale 导入替换为 gemm_a8w8_blockscale_bpreshuffle，并移除冗余导入。
2. 权重量化处理（python/sglang/srt/layers/quantization/fp8.py）：在 process_weights_after_loading 方法中添加条件判断——对于使用 aiter_w8a8_block_fp8_linear 且非 Triton 调优路径的层，调用 shuffle_weight(layer.weight, (16, 16)) 完成一次洗牌，并将结果写回。
3. 前向传播分发（python/sglang/srt/layers/quantization/fp8_utils.py）：在 aiter_w8a8_block_fp8_linear 函数中，当选择非 Triton 路径时，将 GEMM 算子切换为 gemm_a8w8_blockscale_bpreshuffle，并适配输入 scale 的维度（transpose_scale）。
4. 条件导入保护（python/sglang/srt/layers/quantization/fp8.py）：确保仅在启用 AITER 且为 AMD 平台时导入相关符号，避免在其他平台上引入不必要的依赖。

关键文件：

- python/sglang/srt/layers/quantization/fp8_utils.py（模块 量化层；类别 source；类型 dependency-wiring；符号 aiter_w8a8_block_fp8_linear, use_aiter_triton_gemm_w8a8_tuned_gfx950）：核心路由逻辑：将非 Triton 路径的 GEMM 算子从 gemm_a8w8_blockscale 替换为 gemm_a8w8_blockscale_bpreshuffle，并适配输入 scale 转换。

- python/sglang/srt/layers/quantization/fp8.py（模块 量化层；类别 source；类型 dependency-wiring；符号 process_weights_after_loading, process_weights_after_loading_block_quant）：负责在权重加载后执行一次 preshuffle，确保后续前向传播使用洗牌后的权重。同时添加条件导入保护。

关键符号：aiter_w8a8_block_fp8_linear, use_aiter_triton_gemm_w8a8_tuned_gfx950, process_weights_after_loading (in fp8.py)

关键源码片段

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

核心路由逻辑：将非 Triton 路径的 GEMM 算子从 gemm_a8w8_blockscale 替换为 gemm_a8w8_blockscale_bpreshuffle，并适配输入 scale 转换。

```
# python/sglang/srt/layers/quantization/fp8_utils.py
# 条件导入：仅在使用 AITER 时引入 bpreshuffle 算子
if _use_aiter:
    import aiter
    from aiter import (
        gemm_a8w8_blockscale_bpreshuffle,
        gemm_a8w8_bpreshuffle,
        get_hip_quant,
    )
    from aiter.ops.triton.gemm_a8w8_blockscale import (
        gemm_a8w8_blockscale as triton_gemm_a8w8_blockscale,
    )

    aiter_per1x128_quant = get_hip_quant(aiter.QuantType.per_1x128)

def aiter_w8a8_block_fp8_linear(...):
    # ... 省略前处理
    n, k = weight.shape
    if _use_aiter_gfx95:
        use_triton = use_aiter_triton_gemm_w8a8_tuned_gfx950(n, k)
    else:
        use_triton = True

    # 对于 input_scale 非 None 的情况（预量化输入），非 Triton 路径需要转置 scale
    if input_scale is not None:
        q_input = input_2d
        x_scale = input_scale
        if not use_triton:
            x_scale = x_scale.transpose(-1, -2).contiguous().view(*x_scale.shape)
    else:
        q_input, x_scale = aiter_per1x128_quant(
            input_2d,
            quant_dtype=aiter.dtypes.fp8,
            transpose_scale=not use_triton,
        )
```

```
    # 非 Triton 路径使用 bpreshuffle 算子
    if use_triton:
        gemm_a8w8_blockscale_op = triton_gemm_a8w8_blockscale
    else:
        # TODO(1am9trash), to deal with chance of this branch changes
        gemm_a8w8_blockscale_op = gemm_a8w8_blockscale_bpreshuffle

    output = gemm_a8w8_blockscale_op(
        q_input,
        weight,
        x_scale,
        weight_scale,
        dtype=torch.bfloat16 if input_scale is not None else input.dtype,
    )
    # ... 后处理
```

python/sglang/srt/layers/quantization/fp8.py

负责在权重加载后执行一次 preshuffle，确保后续前向传播使用洗牌后的权重。同时添加条件导入保护。

```
# python/sglang/srt/layers/quantization/fp8.py - 部分 extract
# 在 process_weights_after_loading 方法中添加如下条件块
if (
    _use_aiter_gfx95
    and self.w8a8_block_fp8_linear is aiter_w8a8_block_fp8_linear
):
    n, k = layer.weight.shape
    # 仅当当前形状不在 Triton 调优列表中时，才进行 preshuffle
    if not use_aiter_triton_gemm_w8a8_tuned_gfx950(n, k):
        # TODO(1am9trash), to deal with case that this branch chance
        # drops as use_aiter_triton_gemm_w8a8_tuned_gfx950() expands
        t = shuffle_weight(layer.weight, (16, 16))
        layer.weight.copy_(t)
        del t
```

评论区精华

Reviewer HaiShaw 要求在 fp8.py 中确保新符号 aiter_w8a8_block_fp8_linear 仅在使用 AITER 的 AMD 平台上导入（即 _use_aiter 为 True 且 _is_hip）。作者通过 if _use_aiter: 条件导入满足该要求。该讨论已解决，无未解决疑虑。

- 条件导入保护 (design): 作者通过 if _use_aiter: 条件导入，并在 fp8.py 中新增条件导入块，满足要求。

风险与影响

- 风险：
  1. 精度略微下降：GSM8K 测试显示准确率从 0.949 降至 0.943，下降约 0.6 个百分点。可能是 bpreshuffle 数值行为差异导致，需评估是否可接受。

2. 未覆盖所有 GEMM 形状：use_aiter_triton_gemm_w8a8_tuned_gfx950 仅对特定 (n,k) 组合返回 False 才走 bpreshuffle 路径，其他形状仍使用 Triton 版本，灵活性较好。
3. 仅影响 AMD gfx950：通过 _use_aiter_gfx95 开关控制，不影响其他硬件后端。 - 影响：用户：仅影响 AMD GPU（特别是 MI355X）上使用 FP8 块缩放量化的用户，预期获得 8-11% 吞吐量提升和 8-10% TPOT 下降，但可能伴随微小精度损失。系统：无外部 API 或接口变更，权重加载阶段增加一次洗牌操作（约 13-21us 额外开销），但可被后续多次前向传播分摊。团队：代码量小，但引入了新的 AITER 算子依赖（gemm_a8w8_blockscale_bpreshuffle），需要确认该算子在所有目标 AMD 版本上可用。

- 风险标记：核心量化路径变更，精度略微下降，仅 AMD 平台生效

关联脉络

- 暂无明显关联 PR