PR #44572 完整报告

vllm-project/vllm

[Perf] SM90 cutlass fp8 mm supports odd M by swap_ab, 180~290% kernel performance improvement

合并时间：2026-06-14 03:05

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44572

执行摘要

- 一句话：SM90 FP8 GEMM 支持奇数 M，性能提升 180~290%
- 推荐动作：值得精读。本 PR 展示了如何通过 kernel 模板参数交换（swap_ab）来绕过 cutlass 对 M 维度的对齐限制，是一种通用的性能优化技巧。同时代码清理（移除 padding hack）也值得借鉴。建议关注 C++ kernel 中模板参数对布局的调整方式。

功能与动机

之前版本（#43706）通过 Python 层 padding 支持奇数 M，引入额外拷贝和计算开销。本 PR 在 kernel 层面直接使用 swap_ab 交换 A 和 B 的角色，使 cutlass 原生支持奇数 M，从而消除 padding，显著提升性能。PR body 中给出了 benchmark 数据，加速比达 1.87x-2.88x。

实现拆解

1. C++ kernel 模板扩展：在 csrc/libtorch_stable/quantization/w8a8/cutlass/c3x/scaled_mm_blockwise_sm90_fp8_dispatch.cuh 中，为 cutlass_3x_gemm_fp8_blockwise 结构体新增 swap_ab_ 模板参数（默认 false）。根据 swap_ab 将矩阵 A/B 的布局、对齐、缩放配置等进行交换，使 kernel 内部从数学上等价于转置后的 GEMM，从而正确处理奇数 M。
2. Python 层清理：在 vllm/model_executor/kernels/linear/scaled_mm/cutlass.py 中：
   - 移除 direct_register_custom_op 导入和 _padded_cutlass、_padded_cutlass_fake、_dynamic_padded_cutlass 等 padding 辅助函数。
   - CutlassFp8BlockScaledMMKernel.__init__ 移除 weight_group_shape 和 is_hopper 属性；apply_block_scaled_mm 方法被简化为直接调用 ops.cutlass_scaled_mm，不再依赖 dynamic_padded_cutlass 自定义 op。
   - 全局函数 cutlass_scaled_mm 同样简化，不再包含 padding 分支。
3. 测试调整：在 tests/kernels/quantization/test_cutlass_scaled_mm.py 中，删除 if m % 4 != 0 and current_platform.has_device_capability(100): return 这一提前返回条件，使得 SM100 上奇数 M 也能正常测试（实际上本 kernel 只针对 SM90，但测试框架仍会执行）。

关键文件：

- vllm/model_executor/kernels/linear/scaled_mm/cutlass.py（模块 量化 Kernel；类别 source；类型 data-contract；符号 _padded_cutlass, _padded_cutlass_fake, _dynamic_padded_cutlass, run_padded）：核心 Python 层，移除 padding 逻辑和动态 op 注册，简化 apply_block_scaled_mm
- csrc/libtorch_stable/quantization/w8a8/cutlass/c3x/scaled_mm_blockwise_sm90_fp8_dispatch.cuh（模块 CUDA Kernel；类别 other；类型 core-logic；符号 EpilogueScheduler）：C++ kernel 模板添加 swap_ab 参数，实现核心 swap 逻辑
- tests/kernels/quantization/test_cutlass_scaled_mm.py（模块 测试套件；类别 test；类型 test-coverage）：测试调整，移除对 SM100 奇数 M 的跳过条件

关键符号：apply_block_scaled_mm, cutlass_3x_gemm_fp8_blockwise, cutlass_scaled_mm

关键源码片段

vllm/model_executor/kernels/linear/scaled_mm/cutlass.py

核心 Python 层，移除 padding 逻辑和动态 op 注册，简化 apply_block_scaled_mm

```
class CutlassFp8BlockScaledMMKernel(Fp8BlockScaledMMLinearKernel):
    # 经过清理后的 kernel，移除了 padding 逻辑
    def __init__(self, config: FP8ScaledMMLinearLayerConfig) -> None:
        super().__init__(config)
        act_scale_descriptor = config.activation_quant_key.scale
        self.quant_fp8 = QuantFP8(
            static=act_scale_descriptor.static,
            group_shape=act_scale_descriptor.group_shape,
            num_token_padding=self.get_output_padding(),
            use_ue8m0=False,
            column_major_scales=True,
        )
        # 注意：移除了 self.is_hopper 和 self.weight_group_shape,
        # 因为现在 kernel 层直接通过 swap_ab 处理奇数 M,
        # 不再需要 Python 层分支和 padding 信息

    def apply_block_scaled_mm(
        self,
        A: torch.Tensor,
        B: torch.Tensor,
        As: torch.Tensor,
        Bs: torch.Tensor,
    ) -> torch.Tensor:
        out_dtype = self.config.out_dtype
        # 直接调用底层 cutlass_scaled_mm，不再区分 is_hopper,
        # 底层 C++ kernel 会根据 GPU 能力自动选择是否使用 swap_ab
        return ops.cutlass_scaled_mm(
            A,
            B.T,
            out_dtype=out_dtype,
            scale_a=As,
```

```
            scale_b=Bs.T,
        )
```

csrc/libtorch_stable/quantization/w8a8/cutlass/c3x/scaled_mm_blockwise_sm90_fp8_dispatch.cuh

C++ kernel 模板添加 swap_ab 参数，实现核心 swap 逻辑

```
// 添加 swap_ab_ 模板参数，控制是否交换 A/B 的角色
template <class OutType, int ScaleGranularityM, int ScaleGranularityN, int ScaleGranularityK,
          class MmaTileShape, class ClusterShape,
          class EpilogueScheduler, class MainloopScheduler,
          bool swap_ab_ = false> // 新增参数，默认不交换
struct cutlass_3x_gemm_fp8_blockwise {
    static constexpr bool swap_ab = swap_ab_;

    // 根据 swap_ab 动态选择矩阵布局
    using LayoutA = conditional_t<swap_ab,
        typename cutlass::layout::LayoutTranspose<cutlass::layout::RowMajor>::type, // 转置
        cutlass::layout::RowMajor>;
    using LayoutB = conditional_t<swap_ab,
        typename cutlass::layout::LayoutTranspose<cutlass::layout::ColumnMajor>::type,
        cutlass::layout::ColumnMajor>;
    // 缩放配置也相应调整 GMMA::Major 顺序
    using ScaleConfig = conditional_t<swap_ab,
        cutlass::detail::Sm90BlockwiseScaleConfig<
            ScaleGranularityM, ScaleGranularityN, ScaleGranularityK,
            cute::GMMA::Major::K, cute::GMMA::Major::MN>,
        cutlass::detail::Sm90BlockwiseScaleConfig<
            ScaleGranularityM, ScaleGranularityN, ScaleGranularityK,
            cute::GMMA::Major::MN, cute::GMMA::Major::K>>;

    // 集体主循环也根据 swap_ab 使用不同的构建参数
    using CollectiveMainloop = conditional_t<swap_ab,
        typename cutlass::gemm::collective::CollectiveBuilder<
            ArchTag, OperatorClass,
            ElementB, cute::tuple<LayoutB, LayoutSFA>, AlignmentB, // 交换 A/B
            ElementA, cute::tuple<LayoutA, LayoutSFB>, AlignmentA,
            ElementAccumulator, MmaTileShape, ClusterShape,
            StageCountAutoCarveout<...>, MainloopScheduler>::CollectiveOp,
        ... // 非 swap 时保持原有顺序
    >;
};
```

评论区精华

仅有 mgoin 的一条审批评论：'Thank god we finally can get rid of this hack, great work!!' 表明团队对移除 padding hack 的认可。无其他技术讨论。

- 移除 padding hack 的认可 (other): PR 被批准，无需修改

风险与影响

- 风险：
  1. 数值精度：swap_ab 不改变数学结果，但浮点运算的累加顺序可能略有不同，需验证精度满足要求（测试中 rtol=0.5, atol=0.15 已覆盖）。
  2. 硬件兼容性：该优化仅对 SM90（Hopper）生效，其他 GPU 回退到原始路径，但 Python 层删除了条件分支，可能导致非 SM90 GPU 也走新路径？实际上 apply_block_scaled_mm 之前是通过 self.is_hopper 分支，现在去掉了分支，对所有 GPU 都直接调用 ops.cutlass_scaled_mm，但 ops.cutlass_scaled_mm 底层对于非 SM90 可能不支持 swap_ab？这需要确认。不过原代码在非 SM90 也是走 ops.cutlass_scaled_mm 的 padding 路径，现在去掉了 padding，但 ops.cutlass_scaled_mm 本身可能不要求 padding，所以应该没问题，只是性能可能不如原来。但需要进一步验证。
  3. 回归风险：删除了大量 padding 逻辑，可能遗漏某些边界情况。测试用例已覆盖常见的 M/N/K 组合，但仍建议增加随机形状的 fuzz 测试。 - 影响：用户端：使用 SM90 GPU（如 H100）且启用了 CUTLASS_FP8_BLOCK_SCALED 的模型，推理性能可提升 1.87~2.88 倍（具体取决于 M 是否为奇数）。系统端：减少了 Python 层的 padding 和拷贝开销，降低了 GPU 内存带宽消耗。团队端：代码量减少约 118 行，移除了一个自定义 op 注册，简化了维护。兼容性：无 API 变更，仅内部实现优化。 - 风险标记：核心 kernel 变更，硬件特定优化，精度需验证

关联脉络

- PR #43706 [Perf] Odd M support via padded cutlass: 本 PR 是 #43706 的 kernel 级别实现，替代了之前的 Python padding 方案。