PR #43706 完整报告

vllm-project/vllm

[Perf] Optimize cutlass fp8 scaled mm bypassing padding, 20% kernel performance improvement

合并时间：2026-06-01 21:05

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43706

执行摘要

- 一句话：优化 FP8 矩阵乘法绕过 padding，提升 20% 性能
- 推荐动作：该 PR 值得维护者精读，展示了一种优雅的 padding 绕过优化模式：通过检查对齐条件选择不同执行路径，并在 torch.compile 下使用 torch.cond 保持可编译性。同时，review 中的建议进一步优化了完全对齐的情况，体现了合作改进的价值。

功能与动机

在 FP8 量化推理中，Cutlass FP8 Block-Scaled MM 内核要求 M 维度按 4 对齐，原有实现总是先对输入进行 padding，导致额外显存拷贝和计算开销。本 PR 通过动态 padding 策略，在输入已对齐时跳过 padding，直接调用底层 cutlass_scaled_mm，从而获得约 20% 的性能提升。基准测试结果来自 H100 GPU 上多组 (m, n, k) 形状。

实现拆解

1. 在 apply_block_scaled_mm 中添加对齐检查：当 is_hopper 且 A.shape[0] % 4 == 0 时，直接调用 ops.cutlass_scaled_mm 绕过 padding（来自 reviewer lgeiger 的建议）。
2. 新增动态 padding 函数 _dynamic_padded_cutlass：该函数根据输入 M 维度是否被 4 整除决定是否调用 _padded_cutlass（需要 padding）或 cutlass_scaled_mm（无需 padding）。在 torch.compile 路径下使用 torch.cond 保持分支正确性。
3. 注册自定义 Op：通过 direct_register_custom_op 将 _dynamic_padded_cutlass 注册为 dynamic_padded_cutlass 操作，并使用与 padded_cutlass 相同的 fake 实现。
4. 更新入口调用：在 apply_block_scaled_mm 中，对于 Hopper 且 M 不对齐的情况，调用 torch.ops.vllm.dynamic_padded_cutlass 替代原有的 torch.ops.vllm.padded_cutlass。
5. 移除 else 分支中的冗余兜底：原来非 Hopper 的 fallback 路径在 else 中，现在因 if 提前返回而被保留，逻辑未变。 该变更仅涉及单个文件，无新增测试用例，但回归风险低。

关键文件：

- vllm/model_executor/kernels/linear/scaled_mm/cutlass.py（模块 FP8 量化；类别 source；类型 performance；符号 _dynamic_padded_cutlass, run_padded, run_direct）：唯一修改的文件，包含 FP8 Block-Scaled MM 内核的动态 padding 优化实现，新增 _dynamic_padded_cutlass 函数并修改 apply_block_scaled_mm 路由逻辑。

关键符号：_dynamic_padded_cutlass, run_padded, run_direct, apply_block_scaled_mm

关键源码片段

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

唯一修改的文件，包含 FP8 Block-Scaled MM 内核的动态 padding 优化实现，新增 _dynamic_padded_cutlass 函数并修改 apply_block_scaled_mm 路由逻辑。

```
def _dynamic_padded_cutlass(
    qx: torch.Tensor,
    weight: torch.Tensor,
    x_scale: torch.Tensor,
    weight_scale: torch.Tensor,
    block_size: list[int],
    output_dtype: torch.dtype,
) -> torch.Tensor:
    # 定义内部函数，用于 torch.cond 的两个分支
    def run_padded(
        qx: torch.Tensor,
        weight: torch.Tensor,
        x_scale: torch.Tensor,
        weight_scale: torch.Tensor,
    ) -> torch.Tensor:
        # 当 M 不是 4 的倍数时，执行 padding 版本
        return _padded_cutlass(
            qx, weight, x_scale, weight_scale, block_size, output_dtype
        )

    def run_direct(
        qx: torch.Tensor,
        weight: torch.Tensor,
        x_scale: torch.Tensor,
        weight_scale: torch.Tensor,
    ) -> torch.Tensor:
        # 当 M 已经是 4 的倍数时，绕过 padding 直接调用
        return cutlass_scaled_mm(
            qx, weight, x_scale, weight_scale, block_size, output_dtype
        )

    # 在 torch.compile 路径下，使用 torch.cond 动态分支
    if torch.compiler.is_compiling():
        return torch.cond(
            qx.shape[0] % 4 != 0, # 条件：是否需要 padding
            run_padded, # 真分支
            run_direct, # 假分支
            (qx, weight, x_scale, weight_scale),
        )

    # 普通 eager 模式下，直接用 if 判断
    if qx.shape[0] % 4 != 0:
        return run_padded(qx, weight, x_scale, weight_scale)
```

```
    return run_direct(qx, weight, x_scale, weight_scale)

# 注册自定义 Op 以支持 PyTorch 的 dispatcher 和 fake tensor
direct_register_custom_op(
    "dynamic_padded_cutlass",
    _dynamic_padded_cutlass,
    fake_impl=_padded_cutlass_fake,
)
```

同时，在 apply_block_scaled_mm 中将对齐检查前置： defapply_block_scaled_mm(self, A: torch.Tensor, B: torch.Tensor, As: torch.Tensor, Bs: torch.Tensor,) -> torch.Tensor: out_dtype = self.config.out_dtype if self.is_hopper: # 当 M 已对齐 4 时，完全绕过 padding，直接调用底层 MM if A.shape[0] % 4 == 0: return ops.cutlass_scaled_mm(A, B.T, out_dtype=out_dtype, scale_a=As, scale_b=Bs.T,) # 否则使用动态 padding Op return torch.ops.vllm.dynamic_padded_cutlass(A, B, As, Bs, list(self.weight_group_shape), out_dtype,) # 非 Hopper 设备仍使用原始路径 return ops.cutlass_scaled_mm(A, B.T, out_dtype=out_dtype, scale_a=As, scale_b=Bs.T,)

评论区精华

Reviewer @lgeiger 在评论中指出：当 M 已经 4 对齐时，可以完全跳过 padding，直接使用 ops.cutlass_scaled_mm。这一建议在最终实现中被采纳，使得优化更加彻底。作者 @yewentao256 回复 'Nice catch, fixed'。

- 当 M 已对齐时直接跳过 padding (performance): 作者 @yewentao256 采纳建议，在 apply_block_scaled_mm 中添加了对 A.shape[0] % 4 == 0 的检查，对齐时直接走非 padding 路径。

风险与影响

- 风险：该变更为纯性能优化，无接口变更，不会破坏现有功能。主要风险在于：当 M 维度在编译时未知且通过 torch.cond 动态分支时，需要确保 CUDA 图兼容性和编译正确性。此外，dynamic_padded_cutlass 在非 compile 路径下的 if 分支与 compile 路径下的 torch.cond 行为一致，但未通过单元测试覆盖（无新增测试文件）。长期看，若后续有其他内核依赖 padded_cutlass 的固定 padding 行为，可能存在隐性假设。
- 影响：对用户：使用 FP8 量化模型且在 Hopper (SM90) 设备上部署的用户将直接受益，获得约 20% 端到端推理性能提升。对系统：无部署配置变更，backward 兼容。对团队：该设计模式（动态 padding 避免不必要的内存操作）可推广至其他需要对齐约束的内核。
- 风险标记：缺少测试覆盖 , torch.compile 路径下需验证

关联脉络

- 暂无明显关联 PR