PR #39505 完整报告

vllm-project/vllm

[compile] Add FlashInfer FP8 async TP fusion and preserve allreduce fusion ordering #27893

合并时间：2026-05-01 13:08

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/39505

执行摘要

- 一句话：FlashInfer FP8 GEMM AsyncTP 融合，提升 B200 性能
- 推荐动作：值得精读。该 PR 展示了如何在 torch.compile 框架下通过模式匹配实现计算 - 通信融合，并充分利用 PyTorch 的 SymmetricMemory 原语。设计决策（如使用 VllmPatternReplacement、避免多余抽象层）具有良好的可扩展性，可为未来类似优化提供参考。

功能与动机

Issue #27893 报告在 B200 上使用 FP8 量化模型时，AsyncTP pass 无法识别 vllm.bmm_fp8 操作，导致矩阵乘法与集体通信（all-gather / reduce-scatter）未被融合，推理性能显著下降。本 PR 通过注册新的模式匹配规则，使 AsyncTP pass 支持 FlashInfer FP8 GEMM 的融合，恢复并提升推理性能。

实现拆解

1. 核心融合模式注册：在 vllm/compilation/passes/fusion/collective_fusion.py 中新增 FlashInferBMMFP8ReduceScatterPattern 和 FlashInferAllGatherBMMFP8Pattern，继承 VllmPatternReplacement 并注册到 AsyncTPPass。这些模式识别计算图中的 bmm_fp8 + reduce_scatter 与 all_gather + bmm_fp8 组合，替换为 SymmetricMemory 提供的融合操作（_fused_scaled_matmul_reduce_scatter_impl 等）。
2. FlashInfer FP8 输出操作：在 vllm/utils/flashinfer.py 中新增 flashinfer_scaled_fp8_mm_out 函数，提供 out-place 版本的 FP8 矩阵乘法，供融合 op 调用。该函数调用 flashinfer.bmm_fp8 并写入预先分配的 out 张量。
3. 自定义 op 注册：在 collective_fusion.py 中通过 direct_register_custom_op 注册 fused_flashinfer_scaled_matmul_reduce_scatter 及其 fake 版本，fake 版本仅返回适当形状的空张量以支持模式匹配的静态分析。
4. Blackwell 序列并行适配：修改 vllm/compilation/passes/fusion/sequence_parallelism.py，为 Blackwell 系列（sm100/sm103 等）添加 SP_MIN_HIDDEN_SIZE（8192）和更保守的 SP_MIN_PER_GPU_SIZE_MB（32 MB）。使用 is_device_capability_family(100) 统一处理多个 Blackwell 变体。
5. 测试与配置调整：删除 test_tp2_async_tp.py 中对 Blackwel 禁用 FlashInferFP8ScaledMMLinearKernel 的 workaround；在 tests/compile/conftest.py 中

添加 is_device_capability_family 的 mock 实现；在 e2e conftest 中当 attention 后端为 FlashInfer 时调整注意力量化融合匹配计数。

关键文件：

- vllm/compilation/passes/fusion/collective_fusion.py（模块 编译优化；类别 source；类型 core-logic；符号 _flashinfer_scaled_mm_out, fused_flashinfer_scaled_matmul_reduce_scatter_fake, fused_flashinfer_scaled_matmul_reduce_scatter, fused_all_gather_flashinfer_scaled_matmul_fake）：核心变更文件。新增 FlashInfer FP8 的 ReduceScatter 与 AllGather 融合模式，重构 AsyncTPPass 使用 VllmFusionPatternMatcherPass，并注册自定义 op。
- vllm/utils/flashinfer.py（模块 工具函数；类别 source；类型 core-logic；符号 flashinfer_scaled_fp8_mm_out）：新增 flashinfer_scaled_fp8_mm_out 函数，提供 out-place FP8 矩阵乘法，供融合 op 调用。
- vllm/compilation/passes/fusion/sequence_parallelism.py（模块 序列并行；类别 source；类型 core-logic）：为 Blackwell 系列添加序列并行阈值配置，并使用 is_device_capability_family 统一处理。
- tests/compile/conftest.py（模块 测试配置；类别 test；类型 test-coverage；符号 is_device_capability_family）：添加 is_device_capability_family 的 mock 实现，支持编译测试中模拟 Blackwell 平台。
- tests/compile/fusions_e2e/test_tp2_async_tp.py（模块 端到端测试；类别 test；类型 test-coverage）：移除 Blackwell 上禁用 FlashInferFP8ScaledMMLinearKernel 的 workaround，表明该 PR 已支持融合。

关键符号：_flashinfer_scaled_mm_out, fused_flashinfer_scaled_matmul_reduce_scatter_fake, fused_flashinfer_scaled_matmul_reduce_scatter, fused_all_gather_flashinfer_scaled_matmul_fake, fused_all_gather_flashinfer_scaled_matmul, FlashInferBMMFP8ReduceScatterPattern.get_inputs, FlashInferAllGatherBMMFP8Pattern.get_inputs, flashinfer_scaled_fp8_mm_out

关键源码片段

vllm/compilation/passes/fusion/collective_fusion.py

核心变更文件。新增 FlashInfer FP8 的 ReduceScatter 与 AllGather 融合模式，重构 AsyncTPPass 使用 VllmFusionPatternMatcherPass，并注册自定义 op。

```
# vllm/compilation/passes/fusion/collective_fusion.py
# （片段：展示 _flashinfer_scaled_mm_out 适配器与 fused_flashinfer_scaled_matmul_reduce_scatter）

def _flashinfer_scaled_mm_out(
    A: torch.Tensor,
    B: torch.Tensor,
    *,
```

```
    scale_a: torch.Tensor,
    scale_b: torch.Tensor,
    out: torch.Tensor,
    bias: torch.Tensor | None = None,
    scale_result: torch.Tensor | None = None,
    out_dtype: torch.dtype | None = None,
    use_fast_accum: bool = False,
) -> None:
    # 延迟导入避免循环依赖
    from vllm.utils.flashinfer import flashinfer_scaled_fp8_mm_out

    # FlashInfer 适配器当前不支持 bias、result scaling 和 fast_accum
    assert bias is None, "FlashInfer symm_mem adapter does not support bias"
    assert scale_result is None, "... does not support result scaling"
    assert not use_fast_accum, "... does not support use_fast_accum"
    assert A.ndim == 2 and B.ndim == 2 and out.ndim == 2
    # 仅支持 per-tensor scalar scale
    assert scale_a.numel() == 1 and scale_b.numel() == 1, \
        "FlashInfer symm_mem adapter only supports tensor-wise FP8 scales"

    flashinfer_scaled_fp8_mm_out(
        A, B, scale_a, scale_b,
        out=out,
        out_dtype=out_dtype or out.dtype,
    )


def fused_flashinfer_scaled_matmul_reduce_scatter(
    A: torch.Tensor,
    B: torch.Tensor,
    A_scale: torch.Tensor,
    B_scale: torch.Tensor,
    reduce_op: str,
    orig_scatter_dim: int,
    scatter_dim_after_maybe_reshape: int,
    group_name: str,
    output_shape: list[int],
    out_dtype: torch.dtype | None = None,
) -> torch.Tensor:
    # 当前仅支持 scatter_dim=0
    assert orig_scatter_dim == 0 and scatter_dim_after_maybe_reshape == 0
    world_size = c10d._resolve_process_group(group_name).size()
    assert A.ndim == 2 and B.ndim == 2
    assert A.is_contiguous()
    assert A_scale.numel() == 1 and B_scale.numel() == 1
    assert A.shape[0] % world_size == 0

    kwargs = {
        "scale_b": B_scale,
```

```
        "bias": None,
        "scale_result": None,
        "out_dtype": out_dtype,
        "use_fast_accum": False,
    }
    # 委托给 PyTorch SymmetricMemory 的通用融合实现
    return torch.distributed._symmetric_memory._fused_scaled_matmul_reduce_scatter_impl(
        mm_out_op=_flashinfer_scaled_mm_out,
        A=A, B=B, A_scale=A_scale, kwargs=kwargs,
        out_dtype=out_dtype, reduce_op=reduce_op,
        orig_scatter_dim=orig_scatter_dim,
        scatter_dim_after_maybe_reshape=scatter_dim_after_maybe_reshape,
        group_name=group_name,
        output_shape=output_shape,
    )
```

vllm/utils/flashinfer.py

新增 flashinfer_scaled_fp8_mm_out 函数，提供 out-place FP8 矩阵乘法，供融合 op 调用。

```
# vllm/utils/flashinfer.py
#  （片段：flashinfer_scaled_fp8_mm_out 函数）

def flashinfer_scaled_fp8_mm_out(
    a: torch.Tensor,
    b: torch.Tensor,
    scale_a: torch.Tensor,
    scale_b: torch.Tensor,
    out: torch.Tensor,
    out_dtype: torch.dtype | None = None,
) -> torch.Tensor:
    assert a.ndim == 2 and b.ndim == 2 and out.ndim == 2
    assert a.shape[1] == b.shape[0]
    assert out.shape == (a.shape[0], b.shape[1])
    assert scale_a.numel() == 1 and scale_b.numel() == 1
    assert a.dtype == torch.float8_e4m3fn and b.dtype == torch.float8_e4m3fn
    assert out.device.type == "cuda"
    assert a.is_contiguous()

    from flashinfer import bmm_fp8 as bmm_fp8_

    # unsqueeze 为 batch=1 调用 FlashInfer 的 bmm_fp8
    bmm_fp8_(
        a.unsqueeze(0),
        b.unsqueeze(0), # FlashInfer 期望权重保持转置布局
        scale_a,
        scale_b,
        out_dtype or out.dtype,
        out.unsqueeze(0),
        "auto",
```

```
    )
    return out
```

评论区精华

Review 中主要讨论包括：

- 文件拆分争论：作者最初创建了单独的 flashinfer_collective_fusion.py，但 ProExpertProg 认为不必要，应复用现有 collective_fusion.py，最终合并。
- View-like 操作处理：作者手动枚举了 view/reshape/squeeze 等操作，ProExpertProg 指出应依赖 Inductor 将此类操作规范化为 reshape，只需在模式中处理 reshape 即可。
- 阈值机制：作者在 sequence_parallelism.py 中添加了多个硬编码阈值，ProExpertProg 建议使用已有的 compile_ranges_endpoints 和 sp_min_token_num 机制，避免重复逻辑。
- Group Name 解析：作者添加了 _resolve_symm_mem_group_name 等辅助函数，ProExpertProg 建议直接在替换中使用 self.tp.device_group.group_name 以简化。
- 自定义 op 注册位置：作者将自定义 op 放在 parallel_state.py，ProExpertProg 要求移动到 collective_fusion.py。

所有讨论均以合入前解决，最终代码干净简洁。

- 是否拆分单独文件 flashinfer_collective_fusion.py (design): 合并到 collective_fusion.py ，删除独立文件。
- View-like 操作处理方式 (design): 移除多余列举，只保留 reshape 支持。
- Min_token 阈值机制 (design): 移除硬编码阈值，完全依赖 compile_ranges 机制。
- Group name 解析简化 (design): 移除辅助函数，直接引用 device_group.group_name。
- 自定义 op 注册位置 (design): 移动到 collective_fusion.py。

风险与影响

- 风险：
  1. 性能回归风险：融合操作在 Hopper（sm90）上未经充分 benchmark，可能因 SMs 调度变化导致小 batch 性能下降。
  2. Scale 假设：当前实现仅支持 per-tensor scalar scale，若未来 FlashInfer 支持 per-token 或 block scale，断言会失败，需扩展适配。
  3. SymmetricMemory 依赖：融合操作依赖 torch.distributed._symmetric_memory，该模块仍处于实验阶段，可能在不同通信后端或拓扑下行为未定义。
  4. Blackwell 阈值保守：SP_MIN_PER_GPU_SIZE_MB=32 可能使某些中等规模 batch 无法受益，需持续调优。 - 影响：用户影响：使用 FlashInfer FP8 且启用 torch.compile（VLLM_COMPILE 模式）的 B200 用户，输出吞吐提升约 2.8%，TTFT 降低约 5-10%（依据 PR 中 benchmark 数据）。H100 用户无直接影响，但源码路径一致，后续可能受益。

系统影响：新增两个自定义 op 注册到 vllm.ops 命名空间，增大二进制体积。AsyncTPPass 改为 VllmFusionPatternMatcherPass，引入 VllmPatternReplacement 基类，便于未来扩展更多融合模式。

团队影响：需维护额外模式匹配规则和自定义 op，理解 SymmetricMemory API 的开发者可快速上手。

- 风险标记：特定 GPU 架构依赖（Blackwell），新 op 未覆盖所有 scale 类型（仅 per-tensor），SymmetricMemory 仍为实验性 API，小 batch 场景可能退化，测试覆盖不够全面（未覆盖 Hopper 端到端）

关联脉络

- PR #27893 [Bug]: AsyncTP pass has poor perf on B200: 本 PR 旨在解决该 issue 中报告的 FlashInfer FP8 GEMM 融合缺失导致的性能问题。