PR #39968 完整报告

vllm-project/vllm

[XPU] Add XPU block-scaled W8A8 fp8 path

合并时间：2026-06-03 20:16

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/39968

执行摘要

- 一句话：XPU 新增块缩放 W8A8 FP8 内核路径
- 推荐动作：该 PR 是理解 vllm 内核选择体系如何扩展的典型例子，展示了添加新平台特定内核并设计 fallback 策略的完整流程。建议内核开发者和平台移植人员精读，尤其是 xpu.py 中类实现和 linear/__init__.py 中注册模式。

功能与动机

在 XPU 上支持 W8A8 块缩放 FP8 量化推理，利用底层 oneDNN kernel（vllm-xpu-kernels 提供）加速矩阵乘，并在原生内核不可用时回退到 Triton 实现。PR body 明确说明 'adds the XPU block-scaled W8A8 FP8 path and updates FP8 block kernel selection so XPU can fall back to Triton when the native XPU FP8 block kernel is unavailable.'

实现拆解

实现分为四步：

1. 新增原生内核类：在 vllm/model_executor/kernels/linear/scaled_mm/xpu.py 中定义 XPUFp8BlockScaledMMKernel，继承 Fp8BlockScaledMMLinearKernel。is_supported 检查平台是否为 XPU；process_weights_after_loading 调用父类后额外对 weight_scale 进行转置并设为 contiguous；apply_block_scaled_mm 调用 torch.ops._xpu_C.fp8_gemm（入口来自 vllm-xpu-kernels），并对输入 B 转置以满足 [K, N] 布局。
2. 导出与注册：在 scaled_mm/__init__.py 中导入 XPUFp8BlockScaledMMKernel 并加入 __all__；在 linear/__init__.py 的 _POSSIBLE_FP8_BLOCK_KERNELS[PlatformEnum.XPU] 列表首位添加该内核，确保优先使用。
3. 保留 Triton fallback：在同一个 XPU 候选列表中添加 TritonFp8BlockScaledMMKernel（位于 scaled_mm/triton.py），并修改其 is_supported 方法的报错信息以明确支持 XPU，当原生内核不可用时自动选择 Triton。
4. 测试与验证：初始包含 mock 测试和精度测试，但 review 后决定移除（测试归属到 vllm-xpu-kernels 仓库）。作者在 B60 上通过 GSM8K 验证了推理精度（accuracy 0.857）。最终合并版本不包含测试文件。

关键文件：

- vllm/model_executor/kernels/linear/scaled_mm/xpu.py（模块 FP8 内核；类别 source；类型 core-logic；符号 XPUFp8BlockScaledMMKernel, is_supported, process_weights_after_loading, apply_block_scaled_mm）：新增 XPUFp8BlockScaledMMKernel 类，实现 XPU 原生块缩放 FP8 矩阵乘，是 PR 的核心变更。
- vllm/model_executor/kernels/linear/scaled_mm/__init__.py（模块 FP8 内核；类别 source；类型 data-contract；符号 XPUFp8BlockScaledMMKernel）：导出 XPUFp8BlockScaledMMKernel 并加入 all，使其对上层模块可见。
- vllm/model_executor/kernels/linear/__init__.py（模块 内核选择；类别 source；类型 data-contract；符号 XPUFp8BlockScaledMMKernel）：将 XPUFp8BlockScaledMMKernel 注册到 XPU 块缩放内核候选列表，并预留 Triton 作为 fallback。
- vllm/model_executor/kernels/linear/scaled_mm/triton.py（模块 FP8 内核；类别 source；类型 core-logic；符号 TritonFp8BlockScaledMMKernel）：修改 TritonFp8BlockScaledMMKernel.is_supported 明确允许 XPU 平台，作为原生内核不可用时的 fallback。

关键符号：XPUFp8BlockScaledMMKernel.is_supported, XPUFp8BlockScaledMMKernel.process_weights_after_loading, XPUFp8BlockScaledMMKernel.apply_block_scaled_mm, TritonFp8BlockScaledMMKernel.is_supported

关键源码片段

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

新增 XPUFp8BlockScaledMMKernel 类，实现 XPU 原生块缩放 FP8 矩阵乘，是 PR 的核心变更。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project

from collections.abc import Sequence
import torch

from vllm.model_executor.layers.quantization.utils.quant_utils import (
    kFp8StaticChannelSym,
    kFp8StaticTensorSym,
)
from vllm.model_executor.utils import replace_parameter
from vllm.platforms import current_platform

from .BlockScaledMMLinearKernel import Fp8BlockScaledMMLinearKernel
from .ScaledMMLinearKernel import FP8ScaledMMLinearKernel,
FP8ScaledMMLinearLayerConfig


class XPUFP8ScaledMMLinearKernel(FP8ScaledMMLinearKernel):
```

```
    # 原有 XPU 非块缩放内核，保持不变
    ...


class XPUFp8BlockScaledMMKernel(Fp8BlockScaledMMLinearKernel):
    """
    XPU 原生块缩放 W8A8 FP8 矩阵乘内核。
    依赖 `torch.ops._xpu_C.fp8_gemm`（来自 vllm-xpu-kernels）。
    当该 op 不可用时，系统自动 fallback 到 Triton 实现。
    """

    @classmethod
    def is_supported(
        cls, compute_capability: int | None = None
    ) -> tuple[bool, str | None]:
        # 仅支持 XPU 平台
        if not current_platform.is_xpu():
            return False, "XPUFp8BlockScaledMM only support on XPU"
        return True, None

    def process_weights_after_loading(self, layer: torch.nn.Module):
        # 先让父类处理框架级别的权重重排
        super().process_weights_after_loading(layer)
        # 确定 scale 属性的名称（兼容 weight_scale_inv 命名）
        scale_attr = (
            "weight_scale_inv" if hasattr(layer, "weight_scale_inv") else "weight_scale"
        )
        scale = getattr(layer, scale_attr)
        # 转置并保证 contiguous，满足 oneDNN kernel 的内存布局要求
        replace_parameter(layer, scale_attr, scale.data.t().contiguous())

    def apply_block_scaled_mm(
        self,
        A: torch.Tensor,
        B: torch.Tensor,
        As: torch.Tensor,
        Bs: torch.Tensor,
    ) -> torch.Tensor:
        # B 的原始 shape 为 [N, K]，通过 .t() 转为 [K, N] 视图避免数据拷贝
        return torch.ops._xpu_C.fp8_gemm(
            A,
            B.t(),
            self.config.out_dtype,
            As,
            Bs,
            torch.Tensor(), # bias，此处不传
        )
```

评论区精华

核心讨论点包括：

- Fallback 策略：审阅者 jikunshang 要求将 TritonFp8BlockScaledMMKernel 也加入 XPU 候选列表作为回退。作者添加后询问条件判断，jikunshang 建议在 is_supported 中处理。最终列表包含两个内核，原生优先。
- 测试有效性：AndreasKaratzas 质疑 mock 测试无法真正验证正确性，作者表示会重做。后续由 jikunshang 决定将详细测试移至 vllm-xpu-kernels 并由其维护。
- 测试归属：jikunshang 评论 we should have such kind of test in vllm-xpu-kernel side. no need to add it here. 最终删除 vllm 侧的测试文件。
- Fallback 策略：是否应在 XPU 候选列表中同时包含原生内核和 Triton 内核 (design): 最终在 _POSSIBLE_FP8_BLOCK_KERNELS[PlatformEnum.XPU] 中同时包含 XPUFp8BlockScaledMMKernel 和 TritonFp8BlockScaledMMKernel，通过 is_supported 中的条件判断决定是否可用，实现优先原生、降级 Triton 的 fallback 策略。
- 测试有效性：mock/ patch 测试是否能真正验证内核行为 (testing): 后续 jikunshang 决定这些测试应归属 vllm-xpu-kernels 仓库，PR 最终移除了 vllm 侧的测试文件。
- 测试归属：详细内核测试应在 vllm-xpu-kernels 仓库而非 vllm 仓库 (testing): 作者删除了该测试文件，最终 PR 不包含 vllm 侧的详细测试。

风险与影响

- 风险：
  1. 缺少 vllm 侧测试覆盖：最终合并 PR 不包含针对 XPUFp8BlockScaledMMKernel 的单元测试，回归风险由外部仓库承担。
  2. 外部依赖：依赖 torch.ops._xpu_C.fp8_gemm 来自 vllm-xpu-kernels 库（PR #173），版本不匹配或接口变更可能导致运行时崩溃。
  3. 权重布局变更：在 process_weights_after_loading 中对 weight_scale 进行了转置和 contiguous 操作，可能与其他量化路径的权重预热逻辑产生冲突。
  4. 回退路径可靠性：Triton kernel 在 XPU 上的性能和正确性尚未充分验证，可能降低推理质量。 - 影响：用户：XPU 用户使用 FP8 量化模型时，自动受益于块缩放内核加速；若原生内核缺失，透明降级到 Triton，功能不中断。系统：仅修改 XPU 平台的 kernel 选择逻辑，CUDA/ROCm/CPU 不受影响。团队：Intel XPU 团队需维护新类并与 vllm-xpu-kernels 对接，其他团队无需关注。影响程度：中等，限于 XPU FP8 推理路径。
- 风险标记：缺少 vllm 侧单元测试，依赖外部 vllm-xpu-kernels 库版本兼容性，权重 scale 转置可能影响加载后状态，Triton 内核在 XPU 上的性能未充分验证

关联脉络

- PR #43759 [XPU]fallback to TRITON_ATTN for vit attn on xpu when use float32 dtype: 同为 XPU 平台上的 fallback 设计模式：当原生实现条件不满足时回退到 Triton。本 PR 的 fallback 思路可参考该 PR 的实现和讨论。