PR #42952 完整报告

vllm-project/vllm

[XPU]feat: enable FP8 block-scaled quantization on XPU

合并时间：2026-05-23 12:33

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42952

执行摘要

- 一句话：XPU 启用 FP8 block-scaled 量化
- 推荐动作：值得精读，因为它展示了如何为 vLLM 添加新的硬件后端支持——尤其是内核注册、平台检测和量化路径的适配。对于打算支持 Intel GPU、AMD 或其他非 NVIDIA 后端的开发者，此 PR 是典型范例。此外，讨论中关于如何绕过 Triton 编译限制的设计思路也值得借鉴。

功能与动机

作为 DeepSeek-V4 XPU 支持系列的一部分，需要在 Intel XPU 上支持 FP8 块缩放量化。现有代码仅支持 CUDA 和 ROCm，XPU 需要类似的自定义算子路径。具体动机参见 PR body：'Enable the Triton-based FP8 block-scaled GEMM kernel on XPU, and add necessary scale handling for the XPU quantization path.' 此外，Intel Triton 无法处理 fp8e4nv 类型，因此需要利用自定义 op 来绕开编译问题。

实现拆解

1. 打开 XPU 平台检测: 在 TritonFp8BlockScaledMMKernel.is_supported() 中增加 current_platform.is_xpu() 条件（文件 vllm/model_executor/kernels/linear/scaled_mm/triton.py），使该内核对 XPU 设备可见。
2. 注册 XPU 内核: 在 vllm/model_executor/kernels/linear/__init__.py 的 _POSSIBLE_FP8_BLOCK_KERNELS 字典中为 PlatformEnum.XPU 添加 TritonFp8BlockScaledMMKernel，使线性层选择器在 XPU 平台上能够选中该内核。
3. 修改 XPU 量化输入路径: 在 vllm/model_executor/layers/quantization/input_quant_fp8.py 的 forward_xpu 方法中，对于动态 group quantization，调用 fp8_utils.per_token_group_quant_fp8（而非 fallback 到 forward_cuda），从而避开 Intel Triton 对 fp8e4nv 类型编译的限制。
4. 扩展量化工具函数: 在 vllm/model_executor/layers/quantization/utils/fp8_utils.py 中，per_token_group_quant_fp8 函数新增 is_xpu() 分支，调用自定义 C++/Triton op torch.ops._C.per_token_group_fp8_quant；同时将 w8a8_triton_block_scaled_mm 中 E8M0 缩放因子上转换条件从 is_rocm() 扩展为 is_rocm() or is_xpu()，因为 XPU 也需要将 E8M0 格式转换为 FP32 后再送入 Triton 内核。

此系列变更均为平台注册和路由调整，不影响 CUDA/ROCM 现有逻辑。由于依赖 vllm-xpu-kernels 中的自定义 op，需要该外部库提供 XPU 上的 per_token_group_fp8_quant 实现。

关键文件：

- vllm/model_executor/layers/quantization/input_quant_fp8.py（模块 量化；类别 source；类型 data-contract；符号 forward_xpu）：核心量化输入处理，修改 forward_xpu 方法，为 XPU 动态 group quantization 添加自定义 op 路由，避免 Intel Triton 编译 fp8e4nv 失败。
- vllm/model_executor/layers/quantization/utils/fp8_utils.py（模块 量化工具；类别 source；类型 data-contract；符号 per_token_group_quant_fp8, w8a8_triton_block_scaled_mm）：提供 XPU 上的 per_token_group_fp8_quant 自定义 op 调用，以及 E8M0 scale 上转换的 XPU 支持。
- vllm/model_executor/kernels/linear/scaled_mm/triton.py（模块 内核；类别 source；类型 data-contract；符号 TritonFp8BlockScaledMMKernel.is_supported）：修改 is_supported 方法，允许 XPU 设备使用该内核。
- vllm/model_executor/kernels/linear/__init__.py（模块 线性层；类别 source；类型 data-contract）：注册 TritonFp8BlockScaledMMKernel 到 XPU 平台的内核选择列表中。

关键符号：forward_xpu, TritonFp8BlockScaledMMKernel.is_supported, per_token_group_quant_fp8, w8a8_triton_block_scaled_mm

关键源码片段

vllm/model_executor/layers/quantization/input_quant_fp8.py

核心量化输入处理，修改 forward_xpu 方法，为 XPU 动态 group quantization 添加自定义 op 路由，避免 Intel Triton 编译 fp8e4nv 失败。

```
    def forward_xpu(
        self,
        x: torch.Tensor,
        scale: torch.Tensor | None = None,
        scale_ub: torch.Tensor | None = None,
        use_triton: bool = False,
    ) -> tuple[torch.Tensor, torch.Tensor]:
        # Dynamic group quant 在 XPU 上需要特殊处理
        if self.is_group_quant and not self.static:
            from vllm.model_executor.layers.quantization.utils import fp8_utils

            # 调用 fp8_utils.per_token_group_quant_fp8,
            # 在 XPU 上它会使用自定义 op，避免 Triton 编译 fp8e4nv 问题
            return fp8_utils.per_token_group_quant_fp8(
                x,
                group_size=self.group_size,
                column_major_scales=self.column_major_scales,
                dtype=_FP8_DTYPE,
                use_ue8m0=self.use_ue8m0,
```

```
        )
    # 其他情况与 CUDA 路径一致
    return self.forward_cuda(x, scale, scale_ub, use_triton)
```

vllm/model_executor/layers/quantization/utils/fp8_utils.py

提供 XPU 上的 per_token_group_fp8_quant 自定义 op 调用，以及 E8M0 scale 上转换的 XPU 支持。

```
# prefer CUDA/XPU kernel if available
if current_platform.is_cuda() and x.is_contiguous():
    torch.ops._C.per_token_group_fp8_quant(
        x, x_q, x_s, group_size, eps, fp8_min, fp8_max,
        use_ue8m0, column_major_scales, tma_aligned_scales,
    )
    return x_q, x_s

# XPU 自定义 op 路径（暂时不支持 column_major_scales/tma_aligned_scales）
if current_platform.is_xpu() and x.is_contiguous():
    torch.ops._C.per_token_group_fp8_quant(
        x, x_q, x_s, group_size, eps, fp8_min, fp8_max, use_ue8m0
    )
    return x_q, x_s
```

评论区精华

- 正确性问题: gemini-code-assist[bot] 指出最初的 forward_xpu 实现缺少 fp8_utils 导入，且未传递 column_major_scales 和 use_ue8m0 参数，可能导致运行时错误或精度损失。后续修改解决了该问题。
- 设计选择: jikunshang 询问为什么不直接使用 per_token_group_quant_fp8。majian4work 解释因 XPU 内核接口尚未对齐 column_major_scales 和 tm_aligned_scales，但 Yejing-Lai 确认 XPU 支持 group quant 并计划更新 XPU kernel API 对齐，最终采用了 per_token_group_quant_fp8。
- XPU 支持确认: Yejing-Lai 提供了 XPU 上 group quant 的参考实现链接，证实了该功能可行。
  - forward_xpu 动态 group quant 实现问题 (correctness): 修改后使用 fp8_utils.per_token_group_quant_fp8 并正确传入参数
  - 是否使用 per_token_group_quant_fp8 函数 (design): 决定改用 per_token_group_quant_fp8，并依赖 Yejing-Lai 更新 XPU kernel API 对齐
  - XPU group quantization 支持的确认 (question): 确认支持，Yejing-Lai 将更新 XPU kernel API

风险与影响

- 风险:
  - 回归风险: 本 PR 仅改动 XPU 路径，CUDA/ROCM 逻辑保持不变。但 fp8_utils.py 中 E8M0 上转换条件从 is_rocm() 改为 is_rocm() or is_xpu()，可能因意外的平台匹配错误

引入 ROCm 路径的细微行为变化，但可能性较低。

- 性能风险: 在 XPU 上使用 Triton-based 内核，性能可能不如 CUDA 上的专门内核。由于是首次使能，后续需要 benchmark 调优。
- 兼容性风险: 依赖于 vllm-xpu-kernels 提供的自定义 op per_token_group_fp8_quant，如果该 op 未正确安装或版本不匹配，将导致推理失败。
- 测试覆盖: 没有新增测试用例，仅依赖手动测试和 CI 中可能存在的 XPU 测试。test_fused_moe_lora_kernel_fully_sharded 已在本地通过，但 CI 中该测试仍在验证中（见 Issue 评论中 jikunshang 请求检查）。

- 影响:
  - 用户影响: XPU 用户现在可以运行 FP8 量化的模型，如 DeepSeek-V4，无需回退到 CUDA 模拟或不支持状态。对于使用 Intel GPU（如 Ponte Vecchio、Max）的用户是直接收益。
  - 系统影响: 不影响其他后端。线性层 kernel 选择器新增一个平台条目，调度开销可忽略。
  - 团队影响: 需要维护 XPU 特定的量化路径和自定义 op，但本 PR 将逻辑集中到了 fp8_utils 和 input_quant_fp8，保持了一定可维护性。
  - 风险标记：缺少测试覆盖，依赖自定义 XPU op，新硬件路径需性能验证

关联脉络

- PR #42950 [XPU]fix: add XPU platform guards to DeepSeek-V4 ops: 同一 DeepSeek-V4 XPU 支持系列，为本 PR 的 XPU 量化使能提供平台守卫基础。
- PR #42209 Add NVFP4 MOE support for Deepseek V4.: DeepSeek-V4 量化支持系列，本 PR 补充了 FP8 块缩放量化路径，两者共同覆盖了 DeepSeek-V4 的量化推理需求。