PR #44132 完整报告

vllm-project/vllm

[Quantization] add online fp8 ptpc

合并时间：2026-06-08 22:42

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/44132

执行摘要

- 一句话：新增在线 FP8 per-channel 量化方法
- 推荐动作：值得精读，展示了在线量化框架的扩展方式，包括量化键（QuantKey）、调度表注册、MoE 集成以及测试策略。关注 MarlinFP8 兼容性检查和 ROCm 缺失问题。

功能与动机

用户在使用在线量化时缺乏 per-channel 粒度的 weight scale + per-token activation 方案。llmcompressor 的 FP8_DYNAMIC 配方精度良好，但要求预量化检查点。本 PR 提供了同等的量化布局，无需预量化，只需指定 --quantization fp8_per_channel。参考 PR body 和 docstring。

实现拆解

1. 新增量化方法类：在 vllm/model_executor/layers/quantization/online/fp8.py 中添加 Fp8PtpcOnlineLinearMethod 和 Fp8PtpcOnlineMoEMethod，分别继承 _Fp8OnlineLinearBase 和 _Fp8OnlineMoEBase。使用 kFp8StaticChannelSym（权重）和 kFp8DynamicTokenSym（激活）量化键。
2. 配置注册：在 vllm/config/quantization.py 中添加 fp8_per_channel 缩写，映射到 QuantSpec(weight=kFp8StaticChannelSym)，并注册 QUANT_KEY_NAMES。
3. 调度表更新：在 vllm/model_executor/layers/quantization/online/base.py 中将 kFp8StaticChannelSym 映射到新方法类，更新 _ONLINE_LINEAR_METHODS 和 _ONLINE_MOE_METHODS。
4. 入口注册：在 vllm/model_executor/layers/quantization/__init__.py 的 _ONLINE_SHORTHANDS 校验列表中添加 "fp8_per_channel"。
5. 测试配套：新增 tests/quantization/test_fp8_per_channel.py 测试注册表一致性和 kernel 行为；新增 tests/models/quantization/test_fp8_per_channel.py 进行端到端 logprobs 对比（使用 dense 和 MoE 模型）。
6. ROCm 兼容处理：测试中使用 is_quant_method_supported("fp8_per_channel") 跳过不支持平台（如 ROCm）。

关键文件：

- vllm/model_executor/layers/quantization/online/fp8.py（模块 量化方法；类别 source；类型 core-logic；符号 Fp8PtpcOnlineLinearMethod, create_weights,

process_weights_after_loading, apply）：核心实现，新增 Fp8PtpcOnlineLinearMethod 和 Fp8PtpcOnlineMoEMethod，包含权重创建、处理和应用逻辑。

- vllm/model_executor/layers/quantization/online/base.py（模块 量化框架；类别 source；类型 data-contract）：在线量化调度表，注册新方法类的映射。
- vllm/config/quantization.py（模块 配置层；类别 source；类型 core-logic）：配置定义，添加 fp8_per_channel 缩写和 QUANT_KEY_NAMES。
- vllm/model_executor/layers/quantization/__init__.py（模块 量化入口；类别 source；类型 data-contract）：在线量化方法列表注册，确保缩写校验通过。
- tests/quantization/test_fp8_per_channel.py（模块 量化测试；类别 test；类型 test-coverage；符号 test_fp8_per_channel_shorthand_registered, test_scaled_fp8_quant_per_channel_shape, test_fp8_per_channel_online_quantization, check_model）：单元测试：验证注册表一致性、kernel 输出形状、端到端 smoke test。
- tests/models/quantization/test_fp8_per_channel.py（模块 端到端测试；类别 test；类型 test-coverage；符号 test_fp8_per_channel_logprobs）：端到端质量测试：对比 BF16 与 FP8 per-channel 量化的 logprobs。

关键符号：Fp8PtpcOnlineLinearMethod.create_weights, Fp8PtpcOnlineLinearMethod.process_weights_after_loading, Fp8PtpcOnlineLinearMethod.apply, Fp8PtpcOnlineMoEMethod.init, _Fp8OnlineMoEBase.init(重构)

关键源码片段

vllm/model_executor/layers/quantization/online/fp8.py

核心实现，新增Fp8PtpcOnlineLinearMethod和Fp8PtpcOnlineMoEMethod，包含权重创建、处理和应用逻辑。

```
class Fp8PtpcOnlineLinearMethod(_Fp8OnlineLinearBase):
    """Online PTPC FP8 linear quantization.

    Per-output-channel weight scale + dynamic per-token activation scale. The
    layout matches the llmcompressor's FP8_DYNAMIC recipe, so accuracy
    is comparable but no pre-quantized checkpoint is required.
    """

    weight_quant_key = kFp8StaticChannelSym
    activation_quant_key = kFp8DynamicTokenSym

    def create_weights(
        self,
        layer: torch.nn.Module,
        input_size_per_partition: int,
        output_partition_sizes: list[int],
        input_size: int,
        output_size: int,
        params_dtype: torch.dtype,
```

```
    **extra_weight_attrs,
):
    # 调用父类创建原始权重（meta device 上的 fp16/bf16）
    super().create_weights(
        layer,
        input_size_per_partition,
        output_partition_sizes,
        input_size,
        output_size,
        params_dtype,
        **extra_weight_attrs,
    )

    # 根据量化键初始化 FP8 线性 kernel（自动选择 Cutlass / Rocm / Triton）
    self.fp8_linear = init_fp8_linear_kernel(
        activation_quant_key=self.activation_quant_key,
        weight_quant_key=self.weight_quant_key,
        weight_shape=layer.weight.shape,
        input_dtype=self.input_dtype,
        out_dtype=self.out_dtype,
        module_name=self.__class__.__name__,
    )
    # PTPC 需要 per-token activation 量化，MarlinFP8 是 W8A16 仅权重量化，不允许
    if isinstance(self.fp8_linear, MarlinFP8ScaledMMLinearKernel):
        raise ValueError(
            "FP8 PTPC online quant requires a kernel that honors "
            "per-token activation quantization; MarlinFP8 is W8A16 "
            "weight-only. Requires SM89+ for Cutlass FP8 or ROCm MI3xx "
            "for rowwise scaled_mm."
        )

def process_weights_after_loading(self, layer: Module) -> None:
    # 防止重复处理（如权重重载时）
    if getattr(layer, "_already_called_process_weights_after_loading", False):
        return

    layer.input_scale = None
    # 对权重进行 per-channel 量化：scale 维度为 [out_channels, 1]
    qweight, weight_scale = ops.scaled_fp8_quant(
        layer.weight, scale=None, use_per_token_if_dynamic=True
    )

    replace_parameter(layer, "weight", qweight.t())
    replace_parameter(layer, "weight_scale", weight_scale)

    self.fp8_linear.process_weights_after_loading(layer)

    layer._already_called_process_weights_after_loading = True
```

```
    def apply(
        self,
        layer: torch.nn.Module,
        x: torch.Tensor,
        bias: torch.Tensor | None = None,
    ) -> torch.Tensor:
        # 批处理不变性已在 apply_weights 中处理
        return self.fp8_linear.apply_weights(layer, x, bias)
```

评论区精华

- 测试位置：AndreasKaratzas 建议将精度测试合入已有文件，作者解释 tests/models/quantization/ 是质量测试的规范位置。
- 精度验证：AndreasKaratzas 质疑 BF16 与 FP8 的 allclose，作者确认仅在 B200 上通过，reviewer 建议添加小容差避免脆性，但最终未修改。
- ROCm 支持：divakar-amd 指出 fp8_per_channel 未在 ROCm 支持列表中，要求跳过测试。作者采纳并修改了 skip 条件。
- 后端兼容性：AndreasKaratzas 询问 AITER 后端是否兼容，作者未回复。
- 测试文件位置 (style): AndreasKaratzas 收回建议，同意保持新文件。
- 精度验证方法 (testing): 保持现有方式，未添加额外容差。
- ROCm 兼容性 (design): 作者采纳建议，将测试 skipif 条件从 is_quant_method_supported("fp8") 改为 is_quant_method_supported("fp8_per_channel")。
- AITER 后端兼容性 (question): 未得到明确回复，可能需后续跟进。

风险与影响

- 风险：
  - ROCm 兼容性：fp8_per_channel 未在 rocm.py 注册，测试已跳过，但若用户强行使用会导致运行时错误。
  - MarlinFP8 检测：在 create_weights 中显式检查 MarlinFP8ScaledMMLinearKernel 并抛出 ValueError，防止误用。
  - 精度风险：仅使用 logprobs top-k 对比（非实际数值 allclose），可能掩盖细微精度损失。
  - 硬件覆盖：精度测试仅在 B200 (NVIDIA) 上通过，其他 GPU 上未验证。
- 影响：
  - 用户：可通过 --quantization fp8_per_channel 使用新的在线量化方案，提升 FP8 权重量化粒度。
  - 系统：新增两个量化方法类，扩展在线量化框架，对现有功能无影响（未修改任何已有方法）。
  - 团队：为后续 per-channel 量化方案提供了可复用的基类和注册模式。
  - 风险标记：ROCm 兼容性待验证，MarlinFP8 不兼容运行时检测，精度测试仅比较 top-k 而非数值，AITER 后端兼容性未确认

关联脉络

- PR #41184 [MoE Refactor] FusedMoE/MoERunner inversion refactor: 重构了 MoE 量化接口，该 PR 扩展的 Fp8PtpcOnlineMoEMethod 依赖重构后的基类。