# PR #48538 完整报告

vllm-project/vllm

[Quant] Add `nvfp4_per_token` online MoE quantization

合并时间：2026-07-17 05:25

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48538

## 执行摘要

- 一句话：新增 nvfp4_per_token 在线 MoE 量化方案
- 推荐动作：该 PR 值得精读，因为它展示了如何将一个新的硬件特定量化方案集成到 vLLM 的在线量化框架中，包括配置注册、方法分派、权重处理与内核对接。设计清晰，可作为未来扩展的参考。关注其后续的准确性基准和性能优化。

## 功能与动机

添加 --quantization nvfp4_per_token 快捷名，在加载时量化 MoE 权重并启用 per-token 动态激活缩放，以在 Blackwell GPU 上获得更好的性能和内存效率，同时保持接近 bf16 的精度。

## 实现拆解

### 步骤 1：配置注册

在 vllm/config/quantization.py 中，将 nvfp4_per_token 字符串映射到 QuantSpec(weight=kNvfp4Static)，表示 MoE 权重使用静态 NVFP4 量化键。

### 步骤 2：在线分发表注册

在 vllm/model_executor/layers/quantization/online/base.py 中，向 _ONLINE_MOE_METHODS 字典添加 kNvfp4Static: Nvfp4OnlineMoEMethod，使在线量化配置能够分派到新的方法类。

### 步骤 3：实现 Nvfp4OnlineMoEMethod 类

新增文件 vllm/model_executor/layers/quantization/online/nvfp4.py，包含：

- _quantize_moe_weight_to_nvfp4 函数：将堆叠的 MoE 专家权重从 float16/bf16 量化为 NVFP4 格式。利用 scaled_fp4_quant 自定义操作，折叠每专家的全局缩放因子，一次性对所有权重进行量化，产生打包的 FP4 权重 (E, N, K//2)、每块 (group-16) FP8 缩放 (E, N, K//16) 和每专家全局缩放 (E,)。
- Nvfp4OnlineMoEMethod 类：继承 OnlineMoEMethodBase，提供 __init__（检查 Blackwell GPU 并选择后端）、process_weights_after_loading（调用量化和内核设置）、_quantize_weights（替换权重参数并设置激活缩放为 1.0，因为 per-token 缩放由内核在运行时计算）、_setup_kernel（调用 oracle 模块转换为内核格式并构造 FusedMoEKernel 对象）、get_fused_moe_quant_config（返回量化配置）。

### 步骤 4：TRTLLM 内核支持 per-token 激活

修改 vllm/model_executor/layers/fused_moe/experts/trtllm_nvfp4_moe.py：

- 在 TrtLlmNvFp4ExpertsBase.__init__ 中添加 per_token_activation: bool = False 参数，控制是否启用 per-token 模式。
- 添加 expects_unquantized_inputs 属性，当 per_token_activation=True 时返回 True，指示上层跳过预量化。
- 添加 _quantize_per_token_input 方法，调用 FlashInfer 的 nvfp4_quantize 函数，使用预定义的基础缩放和 per_token_activation=True，返回打包 FP4、块缩放和 per-token 缩放。
- 在 workspace_shapes 中，如果 per_token_activation=True，抛出 NotImplementedError 以排除 EP/modular 路径。
- 在 _invoke_kernel 中，如果 per_token_activation=True，调用 _quantize_per_token_input 对输入进行量化，并将 per-token 缩放传递给内核的 per_token_scale 参数（之前为 None）。

### 步骤 5：测试补充

在 tests/quantization/test_online.py 中添加 test_online_nvfp4_per_token_moe 测试函数，使用小型模型验证 MoE 量化方法类型和 dense 层保持未量化。

关键文件：

- vllm/model_executor/layers/quantization/online/nvfp4.py（模块 量化层；类别 source；类型 core-logic；符号 _quantize_moe_weight_to_nvfp4, Nvfp4OnlineMoEMethod, init, process_weights_after_loading）：核心新增文件，定义 Nvfp4OnlineMoEMethod 类和权重量化函数 _quantize_moe_weight_to_nvfp4，是整个变更的主体。
- vllm/model_executor/layers/fused_moe/experts/trtllm_nvfp4_moe.py（模块 MoE 内核；类别 source；类型 data-contract；符号 expects_unquantized_inputs, _quantize_per_token_input）：修改 TRTLLM 内核封装以支持 per-token 激活路径，包括 expects_unquantized_inputs、_quantize_per_token_input 和 _invoke_kernel 中的条件量化。
- vllm/config/quantization.py（模块 配置；类别 source；类型 configuration）：添加 nvfp4_per_token 配置键，将用户友好的字符串映射到量化配置。
- vllm/model_executor/layers/quantization/online/base.py（模块 分发表；类别 source；类型 data-contract）：将 Nvfp4OnlineMoEMethod 注册到在线 MoE 方法分发表中，使其可被调度。
- tests/quantization/test_online.py（模块 测试；类别 test；类型 test-coverage；符号 test_online_nvfp4_per_token_moe, check_model）：添加 smoke test 验证 nvfp4_per_token 量化在 Blackwell GPU 上的基本功能。

关键符号：_quantize_moe_weight_to_nvfp4, Nvfp4OnlineMoEMethod.init, Nvfp4OnlineMoEMethod.process_weights_after_loading, Nvfp4OnlineMoEMethod._quantize_weights, Nvfp4OnlineMoEMethod._setup_kernel, Nvfp4OnlineMoEMethod.get_fused_moe_quant_config, expects_unquantized_inputs, _quantize_per_token_input, test_online_nvfp4_per_token_moe

## 关键源码片段

### vllm/model_executor/layers/fused_moe/experts/trtllm_nvfp4_moe.py

修改 TRTLLM 内核封装以支持 per-token 激活路径，包括 expects_unquantized_inputs、_quantize_per_token_input 和 _invoke_kernel 中的条件量化。

```
# 基础缩放常数：1.0 / (448.0 * 6.0)，由 FlashInfer 参考实现给定
_PER_TOKEN_BASE_GLOBAL_SCALE = 1.0 / (448.0 * 6.0)

class TrtLlmNvFp4ExpertsBase:
    def __init__(
        self,
        moe_config: FusedMoEConfig,
        quant_config: FusedMoEQuantConfig,
        per_token_activation: bool = False, # 新增参数
    ):
        self.moe_config = moe_config
        self.quant_config = quant_config
        self.per_token_activation = per_token_activation # 存储标志
        ...

    @property
    def expects_unquantized_inputs(self) -> bool:
        """如果启用 per-token 激活，则指示上层输入未量化."""
        return self.per_token_activation

    def _quantize_per_token_input(
        self, hidden_states: torch.Tensor
    ) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
        """使用 per-token 全局缩放对激活进行 NVFP4 量化.

        返回 (packed_fp4, block_scale, per_token_scale).
        """
        from flashinfer import SfLayout, nvfp4_quantize

        hs_fp4, hs_block_scale, per_token_scale = nvfp4_quantize(
            hidden_states,
            _PER_TOKEN_BASE_GLOBAL_SCALE,
            sfLayout=SfLayout.layout_linear,
            per_token_activation=True, # 启用 per-token 模式
        )
        return hs_fp4, hs_block_scale, per_token_scale

    def workspace_shapes(...) -> tuple:
        """对于 per-token 激活路径，排除 EP/modular 支持."""
        if self.per_token_activation:
            raise NotImplementedError(
                "NVFP4 per-token activation is only supported on the monolithic "
                "(non-EP) FlashInfer TRTLLM MoE path."
```

```
            )
        ...

    def _invoke_kernel(self, ...):
        """内核入口: per-token 模式下先量化输入再调用内核."""
        if self.per_token_activation:
            hidden_states, block_scale, per_token_scale = (
                self._quantize_per_token_input(hidden_states)
            )
        else:
            block_scale, per_token_scale = a1q_scale, None
        ...
```

## tests/quantization/test_online.py

添加 smoke test 验证 nvfp4_per_token 量化在 Blackwell GPU 上的基本功能。

```
@pytest.mark.skipif(
    not (
        current_platform.is_cuda()
        and current_platform.is_device_capability_family(100)
        and has_flashinfer_trtllm_fused_moe()
    ),
    reason="nvfp4_per_token needs a Blackwell (SM100) GPU + FlashInfer TRTLLM MoE.",
)
def test_online_nvfp4_per_token_moe(vllm_runner, monkeypatch) -> None:
    """验证在线 NVFP4 仅量化 MoE 层, dense 层保持未量化."""
    monkeypatch.setenv("VLLM_ALLOW_INSECURE_SERIALIZATION", "1")

    with vllm_runner(
        "ibm-granite/granite-3.0-1b-a400m-base",
        quantization="nvfp4_per_token",
        enforce_eager=True,
    ) as llm:

        def check_model(model):
            layer = model.model.layers[0]
            # 验证 MoE 层使用了 Nvfp4OnlineMoEMethod
            assert isinstance(
                layer.block_sparse_moe.experts._quant_method, Nvfp4OnlineMoEMethod
            )
            # 验证 attention 输出投影保持未量化
            assert isinstance(
                layer.self_attn.o_proj.quant_method, UnquantizedLinearMethod
            )

        llm.apply_model(check_model)
        outputs = llm.generate_greedy(["Hello my name is"], max_tokens=4)
        print(outputs[0][1])
```

## 评论区精华

### 讨论 1：权重量化优化

审核者 pavanimajety 提议使用 scaled_fp4_experts_quant_sm1xxa 内核进行更高效的权重量化，mgoin 接受作为后续改进。

### 讨论 2：测试覆盖

审核者 LironKesem 询问是否需要在 test_online.py 中添加测试（已添加 smoke test），并进一步建议添加准确性测试（如 tests/kernels/quantization/*）和基准测试（如 benchmarks/kernels/benchmark_nvfp4_gemm.py）。mgoin 回应已做手动 B200 评估，并计划后续添加 nightly 在线量化评估作业。

### 讨论 3：使用警告

审核者 pavanimajety 建议对所有 checkpoint 抛警告声明准确性 / 性能未经验证；mgoin 未直接回应但已在 PR body 中声明了限制。

- 权重量化性能优化 (performance): mgoin 同意作为后续改进（follow-up），当前使用通用 scaled_fp4_quant 实现。
- 测试覆盖：准确性测试和基准测试 (testing): mgoin 回应已手动在 B200 上评估，并计划后续添加 nightly 在线量化评估作业。
- 使用警告 (design): 未直接处理，但 PR body 已声明限制。

## 风险与影响

- 风险：
  1. 硬件限制：仅 Blackwell（SM100）GPU 支持，其他 GPU 会抛出 ValueError。
  2. 内核路径限制：仅支持 FlashInfer TRTLLM 内核路径；Humming 等其他后端尚未支持。
  3. 模型层覆盖不全：仅量化 MoE 权重，dense 层保持 bf16/fp16，可能误导用户认为所有层均已量化。
  4. 部署灵活性受限：EP/modular 路径（workspace_shapes）抛出 NotImplementedError，限制了分布式设置。
  5. 测试覆盖不足：现有测试仅为 smoke test，缺乏全面准确性和性能基准测试。 - 影响：用户：Blackwell GPU 用户可使用 --quantization nvfp4_per_token 快速获得紧凑的 MoE 表示，无需预量化 checkpoint。系统：新增模块依赖 FlashInfer TRTLLM 内核，但集成点最小，不破坏已有功能。团队：需要维护新量化路径，但代码组织清晰，遵循现有在线量化模式。 - 风险标记：仅 Blackwell GPU，仅 FlashInfer TRTLLM 内核，MoE-only 未量化 dense 层，EP/modular 路径抛 NotImplementedError，缺乏准确性测试覆盖

## 关联脉络

- 暂无明显关联 PR