# PR #45703 完整报告

vllm-project/vllm

[Kernel] Extend Marlin thread-tile padding to MoE (WNA16 + FP8/MXFP8)

合并时间：2026-06-24 02:45

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45703

## 执行摘要

- 一句话：Marlin MoE 支持线程块填充，提升 WNA16 和 FP8/MXFP8 性能
- 推荐动作：建议精读，特别是 marlin_moe_padded_intermediate 的设计和各量化路径的集成方式。该 PR 体现了内核约束感知的权重预处理技巧，值得学习。

## 功能与动机

Follow-up to #45295，将线程块填充机制从密集 Marlin 路径和 NVFP4 MoE 扩展到 WNA16、FP8 和 MXFP8 MoE。TP 分片后中间大小可能不满足 Marlin 的 64 对齐要求，导致 WNA16 MoE 拒绝使用 Marlin（回退到慢速 kernel）和 FP8/MXFP8 MoE 在重打包时崩溃。通过零填充到有效线程块，可以安全启用 Marlin kernel 并获得性能提升。

## 实现拆解

1. 引入填充中间大小计算函数 marlin_moe_padded_intermediate，使用 round_up(intermediate, lcm(64, group_size)) 确保满足 Marlin 线程块要求且分组计数不变。
2. 在 WNA16 MoE 权重预处理函数 _process_weights_marlin 中添加填充逻辑：对 w13 的列、w2 的行、scales、qzeros、bias 进行填充，同时断言 act-order 情况下不填充。
3. 在 FP8 MoE 权重预处理函数 prepare_fp8_moe_layer_for_marlin 中添加填充逻辑：调用 _moe_pad_shard_rows 填充 w13 权重 /scales，_moe_pad_last 填充 w2 权重 /scales，并调整 scale 置换逻辑适应填充后尺寸。
4. 修改 check_moe_marlin_supports_layer 增加 allow_tile_padding 参数，当启用时只要求 hidden size 和 group 对齐，不再要求中间大小被 64 整除。
5. 在压缩张量、AWQ、GPTQ 量化方法的 MoE 创建路径中传入 allow_tile_padding=True，使填充机制生效。

关键文件：

- vllm/model_executor/layers/quantization/utils/marlin_utils.py（模块 量化工具；类别 source；类型 data-contract；符号 marlin_moe_padded_intermediate, check_moe_marlin_supports_layer）：核心函数 marlin_moe_padded_intermediate 和 check_moe_marlin_supports_layer 修改，定义填充逻辑和检查入口。
- vllm/model_executor/layers/quantization/utils/marlin_utils_fp8.py（模块 量化工具；类别 source；类型 data-contract；符号 _moe_pad_shard_rows, _moe_pad_last）：

FP8/MXFP8 MoE 填充实现，新增 _moe_pad_shard_rows 和 _moe_pad_last 辅助函数，并在 prepare_fp8_moe_layer_for_marlin 中集成填充调用。

- vllm/model_executor/layers/fused_moe/oracle/int_wna16.py（模块 MoE 路由；类别 source；类型 data-contract；符号 _pad_w13_shard_cols, _pad_rows, _pad_w13_bias）：WNA16 MoE 填充辅助函数和集成，在 _process_weights_marlin 中添加填充调用。

关键符号：marlin_moe_padded_intermediate, check_moe_marlin_supports_layer, _pad_w13_shard_cols, _pad_rows, _pad_w13_bias, _moe_pad_shard_rows, _moe_pad_last

## 关键源码片段

### vllm/model_executor/layers/quantization/utils/marlin_utils.py

核心函数 marlin_moe_padded_intermediate 和 check_moe_marlin_supports_layer 修改，定义填充逻辑和检查入口。

```
def marlin_moe_padded_intermediate(
    intermediate_size: int, group_size: int = -1
) -> int:
    # 计算满足 MoE Marlin 线程块的最小中间大小
    # 约束：gate-up 需要 2 * intermediate % 128 == 0
    # down 需要 intermediate % 64 == 0，即 intermediate % 64 == 0
    # 同时保持 group_size 倍数性以确保分组计数完整
    group = group_size if group_size > 0 else 1
    # round_up 到 lcm(64, group) 的倍数
    padded = round_up(intermediate_size, math.lcm(64, group))
    if padded != intermediate_size:
        logger.warning_once(
            "Marlin requires thread-tile padding for the MoE intermediate size "
            "of some layers in this model. Padded experts pad/slice activations "
            "on every forward; performance may be degraded."
        )
    return padded


def check_moe_marlin_supports_layer(
    layer: RoutedExperts, group_size: int, allow_tile_padding: bool = False
) -> bool:
    # 判断 fused MoE Marlin kernel 是否支持该层
    # 当 allow_tile_padding=True 时，允许在权重预处理时进行 tile 填充
    if current_platform.is_rocm():
        return False
    hidden_size = layer.hidden_size
    intermediate_size_per_partition = (
        layer.moe_config.intermediate_size_per_partition_unpadded
    )
    assert intermediate_size_per_partition is not None
    supports_router_weight = not layer.apply_router_weight_on_input
```

```
    if allow_tile_padding:
        # 启用填充时仅要求 hidden_size 被 128 整除且 group 对齐
        supports_shape = hidden_size % 128 == 0 and (
            group_size <= 0
            or intermediate_size_per_partition % group_size == 0
        )
    else:
        # 严格模式要求中间大小被 max(64, group_size) 整除
        supports_shape = (
            hidden_size % 128 == 0
            and intermediate_size_per_partition % max(64, group_size) == 0
        )
    supports_group_size = group_size in [-1, 32, 64, 128]
    return (
        supports_shape and supports_group_size and supports_router_weight
    )
```

## vllm/model_executor/layers/quantization/utils/marlin_utils_fp8.py

FP8/MXFP8 MoE 填充实现，新增 _moe_pad_shard_rows 和 _moe_pad_last 辅助函数，并在 prepare_fp8_moe_layer_for_marlin 中集成填充调用。

```
def _moe_pad_shard_rows(
    x: torch.Tensor, n: int, padded_n: int
) -> torch.Tensor:
    # 将 (E, 2*n, ...) 中每个 gate/up shard 的行从 n 填充到 padded_n
    # FP8 零解码为 0.0，因此填充行对输出无贡献
    if padded_n == n:
        return x
    e = x.size(0)
    rest = x.shape[2:]
    x = x.view(e, 2, n, *rest)
    padding = (0, 0) * len(rest) + (0, padded_n - n)
    x = torch.nn.functional.pad(x, padding)
    return x.reshape(e, 2 * padded_n, *rest)


def _moe_pad_last(
    x: torch.Tensor, n: int, padded_n: int
) -> torch.Tensor:
    # 将 (E, ..., n) 的最后一维从 n 填充到 padded_n
    if padded_n == n:
        return x
    return torch.nn.functional.pad(x, (0, padded_n - n))
```

## vllm/model_executor/layers/fused_moe/oracle/int_wna16.py

WNA16 MoE 填充辅助函数和集成，在 _process_weights_marlin 中添加填充调用。

```
def _pad_w13_shard_cols(
    x: torch.Tensor, unit: int, padded_unit: int
```

```
) -> torch.Tensor:
    # 将 (E, rows, 2*unit) 中每个 gate/up shard 的列从 unit 填充到 padded_unit
    if padded_unit == unit:
        return x
    e, rows, _ = x.shape
    x = x.view(e, rows, 2, unit)
    x = torch.nn.functional.pad(x, (0, padded_unit - unit))
    return x.reshape(e, rows, 2 * padded_unit).contiguous()


def _pad_rows(x: torch.Tensor, padded_rows: int) -> torch.Tensor:
    # 将 (E, rows, cols) 的行从 rows 填充到 padded_rows
    if padded_rows == x.size(1):
        return x
    return torch.nn.functional.pad(x, (0, 0, 0, padded_rows - x.size(1)))


def _pad_w13_bias(
    bias: torch.Tensor, n: int, padded_n: int
) -> torch.Tensor:
    # 将 (E, 2*n) 中每个 shard 的 bias 从 n 填充到 padded_n
    if padded_n == n:
        return bias
    e = bias.size(0)
    bias = bias.view(e, 2, n)
    bias = torch.nn.functional.pad(bias, (0, padded_n - n))
    return bias.reshape(e, 2 * padded_n).contiguous()
```

# 评论区精华

本 PR 无审核评论，但用户评论指出该 PR 是多个未决 PR（如 #36807）的最完整修复，已取代它们。

- 用户评论 #36807 被取代 (question): 确认该 PR 是统一完整的解决方案。

# 风险与影响

- 风险：
  - 兼容性：填充后权重形状变化，需确保其他路径不受影响（如 act-order 显式禁止）。
  - 性能：填充本身增加极少量预处理开销，但避免 kernel 回退带来正向收益；填充区域的零乘荷不影响输出。
  - 内存：填充后权重增加少量内存占用，但通常很小。
  - 正确性：测试验证了 WNA16 和 FP8/MXFP8 的 round-trip 与反量化参考一致。
- 影响：
  - 用户：分段 MoE 模型自动启用 Marlin kernel，提升性能，不再手动 fallback。
  - 系统：统一了 Marlin MoE 的填充机制，简化支持。

- 团队：需要维护填充函数和检查逻辑。
- 风险标记：权重预处理变更，act-order 不支持填充，内存占用少量增加，性能退化警告

## 关联脉络

- PR #45295 [Kernel] Add thread-tile padding to dense Marlin paths and NVFP4 MoE: 前序 PR，将 tile padding 添加到密集 Marlin 路径和 NVFP4 MoE；本 PR 将相同机制扩展到其余 MoE 变体。
- PR #36807 Fix Marlin FP8 MoE tile alignment for TP-sharded experts: 用户指出该 PR 完全取代了 #36807，解决了相同的 tile 对齐问题。