

PR #45200 完整报告

vllm-project/vllm

[Models] Fix MiMo v2.x QKV TP sharding + FP4 support

合并时间: 2026-06-15 15:13

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/45200>

执行摘要

- 一句话: 修复 MiMo v2.x 在 $TP \neq 8$ 时的 QKV 分片及 FP4 支持
- 推荐动作: 此 PR 值得精读, 原因如下:
 1. 分片算法设计: `_shard_fp8_qkv_proj` 中处理 fp8 块尺度与 KV 头组交织的约束, 采用反量化 - 重排 - 再量化的方案, 是一个清晰的 trade-off 示例。
 2. 量化路由扩展: 在 `Fp8Config` 中通过 `store_dtype` 分流至 `Mxfp4MoEMethod`, 展示了如何在现有量化框架上以最小侵入支持混合量化 checkpoint。
 3. 评估方法: 作者提供了多个数据集和配置的对比表, 值得参考。

功能与动机

当前 MiMo v2.5 Pro 在 $TP \neq 8$ 时会产生垃圾输出, 因为融合 QKV 投影的权重分片假设每个 rank 恰好一个 KV head, 当非此情况时 Q/K/V 交错排列。同时, FP4 模型 (如 MiMo-V2.5-Pro-FP4-DFlash) 的 MoE 专家权重使用了 MXFP4 格式, 但 `Fp8Config` 未识别 `store_dtype` 字段, 错误地使用了 `Fp8MoEMethod`, 导致推理错误。PR 旨在同时解决这两个问题, 使 MiMo v2.x 系列模型在不同 TP 和量化配置下均能正确推理。

实现拆解

实现分为两部分:

1. 修复 QKV 分片 (`mimo_v2.py`) - 新增 `_shard_fp8_qkv_proj` 函数, 处理融合 QKV 权重的 TP 分片。
 - 当 `kv_heads_per_rank == 1` (即 $TP == \text{num_kv_heads}$) 时, 直接使用 `chunk` 分片, 不重新量化, 保持精度。
 - 当 `kv_heads_per_rank > 1` 时, 先将当前 rank 负责的 KV 组反量化至 float32, 按 [所有 Q | 所有 K | 所有 V] 顺序重排, 再通过 `scaled_quantize` 重新量化为 fp8。重新量化是必要的, 因为原 128 行块尺度在 K/V 边界上交错, 无法简单置换。
 - 修改 `load_weights` 中的 `_try_load_fp8_qkv_proj` 辅助方法, 在加载完权重和尺度后调用上述函数进行分片。
2. 支持 MXFP4 MoE (`fp8.py`) - 在 `Fp8Config.__init__` 中新增 `store_dtype` 参数, 用于记录 checkpoint 中量化配置的 `store_dtype` 字段。
 - 在 `Fp8Config.from_config` 中从配置字典读取 `store_dtype`。
 - 在 `Fp8Config.get_quant_method` 中, 对于 `RoutedExperts` 层且 `store_dtype == 'mxfp4'` 时, 返回 `Mxfp4MoEMethod` 而非 `Fp8MoEMethod`。

此 PR 未包含配套测试文件, 但作者在描述中提供了手动测试和标准基准评测结果, 证明无回归。

关键文件:

- `vllm/model_executor/models/mimo_v2.py` (模块 模型层; 类别 source; 类型 data-contract; 符号 `_shard_fp8_qkv_proj`, `_try_load_fp8_qkv_proj`): 核心修复文件: 新增 `_shard_fp8_qkv_proj` 和 `_try_load_fp8_qkv_proj` 来处理融合 QKV 权重在 $TP \neq 8$ 时的正确分片。
- `vllm/model_executor/layers/quantization/fp8.py` (模块 量化; 类别 source; 类型 data-contract): 扩展 `Fp8Config` 以支持 MXFP4 量化路由, 允许 FP4 checkpoint 的 MoE 层使用正确的量化方法。

关键符号: `_shard_fp8_qkv_proj`, `_try_load_fp8_qkv_proj`, `Fp8Config.init`, `Fp8Config.from_config`, `Fp8Config.get_quant_method`

关键源码片段

`vllm/model_executor/models/mimo_v2.py`

核心修复文件: 新增 `_shard_fp8_qkv_proj` 和 `_try_load_fp8_qkv_proj` 来处理融合 QKV 权重在 $TP \neq 8$ 时的正确分片。

```
def _shard_fp8_qkv_proj(
    w_full: torch.Tensor,
    s_full: torch.Tensor,
    num_heads: int,
    num_kv_heads: int,
    head_dim: int,
    v_head_dim: int,
    tp_rank: int,
    tp_size: int,
    block: int = 128,
) -> tuple[torch.Tensor, torch.Tensor]:
    """Shard the fp8 qkv_proj weights for ``tp_rank``.
```

Checkpoint stores fused QKV as ``num\_kv\_heads`` contiguous groups (one per KV head), each ordered ``[Q | K | V]``.

Each TP rank owns ``g = num\_kv\_heads / tp\_size`` of these groups, and the forward expects them de-interleaved into a single Q, K, and V block (``[all Q | all K | all V]``).

When ``g == 1`` a plain chunk suffices (bit-exact, no re-quantization).

When ``g > 1`` we cannot re-permute fp8 block scales because each scale covers 128 rows and K (192 rows) straddles the K/V boundary. Instead we dequantize the rank's groups to float32, reorder rows into the expected layout, and re-quantize to fp8.

```
"""
```

```
assert tp_size <= num_kv_heads and num_kv_heads % tp_size == 0
```

```
kv_heads_per_rank = num_kv_heads // tp_size
```

```
if kv_heads_per_rank == 1:
```

```
    # Bit-exact path: preserve original scales
```

```

w = w_full.chunk(tp_size, dim=0)[tp_rank]
s = s_full.chunk(tp_size, dim=0)[tp_rank]
return w, s

q_rows_per_group = (num_heads // num_kv_heads) * head_dim
k_rows_per_group = head_dim
v_rows_per_group = v_head_dim
rows_per_group = q_rows_per_group + k_rows_per_group + v_rows_per_group
scale_rows_per_group = s_full.shape[0] // num_kv_heads

qs, ks, vs = [], [], []
for g_idx in range(tp_rank * kv_heads_per_rank,
                    (tp_rank + 1) * kv_heads_per_rank):
    row_start = g_idx * rows_per_group
    scale_row_start = g_idx * scale_rows_per_group
    # Dequantize this group's weights to fp32 for reordering
    w_g = w_full[row_start:row_start + rows_per_group].to(torch.float32)
    s_g = s_full[scale_row_start:scale_row_start + scale_rows_per_group]
    # Extract Q, K, V slices from the dequantized group
    qs.append(w_g[:q_rows_per_group])
    ks.append(w_g[q_rows_per_group:q_rows_per_group + k_rows_per_group])
    vs.append(w_g[q_rows_per_group + k_rows_per_group:])

# Concatenate to form [all Q | all K | all V] and re-quantize
w_float = torch.cat(qs + ks + vs, dim=0)
# Use scaled_quantize to produce fp8 weights and block scales
w, s = scaled_quantize(w_float, block_size=block)
return w, s

```

评论区精华

Review 评论主要关注精度影响：

- jeejeelee询问 `_shard_fp8_qkv_proj` 是否影响精确度。作者回复只在 $TP \neq 8$ 时产生变化， $TP=8$ 时跳过重新量化，且之前 $TP \neq 8$ 完全不可用，因此是严格改进。后续提供了 $TP=8$ 的对比评测，表明无回归。
- benchislett追问是否保留了原始 `scale` 还是重新计算。作者解释了在 `kv_heads_per_rank > 1` 时必须重新计算 `scale`，因为 `block` 尺度无法跨 KV 头边界置换，只能通过反量化 - 重排 - 再量化途径。该 trade-off 被认为合理。

最终 Isotr0py 批准了该 PR。

- $TP \neq 8$ 时重新量化对精度的影响 (correctness): $TP=8$ 无精度影响； $TP \neq 8$ 重新量化是必要权衡，评估无回归。
- 是否保留原始 `scale` 还是重新计算 (correctness): $TP=8$ 时 `scale` 保留； $TP \neq 8$ 时重新计算，但精度经评测仍在噪声水平。接受此 trade-off。

风险与影响

- 风险:

1. 精度风险: 在 $TP \neq 8$ 时重新量化 QKV 权重可能引入微小精度损失。作者提供了 GSM8K、AIME25、GPQA Diamond 上的评测结果并声称在噪声范围内无回归, 但更细粒度的任务 (如逻辑推理、数学证明) 未覆盖。
2. 性能风险: 反量化再量化涉及额外的 fp32 运算, 但仅在权重加载时发生 (一次成本), 推理路径不增加开销。
3. 兼容性风险: `store_dtype` 字段在其他 FP8 模型中的含义可能不同于 MiMo, 但当前仅当 `store_dtype == 'mxfp4'` 时影响路由, 现有模型无此字段则行为不变。
4. 测试覆盖: 没有单元测试覆盖 `_shard_fp8_qkv_proj` 或新的分片路径, 回归依赖手动验证。
 - 影响: 对用户: MiMo v2.5 Pro 和 MiMo v2.5 Pro FP4 现在可以在任意 TP 下正常运行 (此前只支持 $TP=8$, 且 FP4 完全不可用)。对现有 $TP=8$ 用户无任何影响 (评测确认)。对系统: 新增约 160 行源码, 无额外依赖。对团队: 需要维护针对特定模型的分片策略, 但代码封装在一个函数内, 易于迭代。
 - 风险标记: 精度风险, 缺少测试覆盖, $TP \neq 8$ 路径未广泛验证

关联脉络

- 暂无明显关联 PR