PR #42595 完整报告

vllm-project/vllm

[Bugfix] [ROCm] [DSV4] Fix AITER MXFP4 MoE weight loading and shuffle…

合并时间：2026-05-29 19:08

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42595

执行摘要

- 一句话：修复 DSv4 AITER MoE 权重加载与 shuffle 三大 Bug
- 推荐动作：建议精读。该 PR 修复了 DeepSeek-V4 在 ROCm 上的关键障碍，并通过一次性能显著提升验证了 AITER FlyDSL MoE 的实用性。尤其值得关注 TP 分片偏移的修正逻辑，以及 AITER 原生 shuffle 函数的对接方式，这为后续其他 MXFP4 后端的同类问题提供了参考。

功能与动机

使用 --moe-backend aiter 运行 DeepSeek-V4 时，由于三个 Bug 导致输出乱码：TP 加载偏移使用了填充后的 shard_size，使最后几个 rank 读取零权重；AITER 分支错误地对已分离的 w1/w3 进行解交织；使用了错误的 shuffle 函数（shuffle_weight_a16w4），与 AITER FlyDSL 内核期望的 shuffle 组合不匹配。修复后才能利用 AITER 的 FlyDSL MoE 加速。

实现拆解

1. layer.py — TP 分片偏移修正：_load_w13 和 _load_w2 中，偏移计算从 shard_size * tp_rank 改为使用 loaded_per_rank * tp_rank，其中 loaded_per_rank = loaded_weight.shape[shard_dim] // tp_size，确保每个 rank 以未经填充的每 rank 大小切片 checkpoint 权重，修复最后几个 rank 加载零值的问题。
2. mxfp4.py — 删除错误解交织：移除 AITER_MXFP4_BF16 分支中对 w13 的 view(e, n//2, 2, k).permute(0,2,1,3) 操作，因为标准 _load_w13 已输出 [gate_all, up_all] 布局，不应再解交织。
3. mxfp4.py — 改用原生 shuffle 函数：将 rocm_aiter_ops.shuffle_weight_a16w4/shuffle_scale_a16w4 替换为匹配 AITER 测试脚本的 aiter.ops.shuffle.shuffle_weight 和 aiter.utility.fp4_utils.e8m0_shuffle，两阶段均使用 shuffle_weight(w, (16,16)) + e8m0_shuffle 组合。
4. 后端优先级调整：select_deepseek_v4_mxfp4_moe_backend 中将 DeepSeek-V4 在 ROCm 上的优先后端从 [TRITON_UNFUSED, AITER_MXFP4_BF16] 翻转为 [AITER_MXFP4_BF16, TRITON_UNFUSED]，使 AITER 成为默认选择。
5. 保留必要导入：应 reviewer 要求保留 from vllm._aiter_ops import rocm_aiter_ops（加 # noqa: F401），避免 rocm_aiter_ops 未定义错误。

关键文件：

- vllm/model_executor/layers/fused_moe/oracle/mxfp4.py（模块 MoE 层；类别 source；类型 core-logic；符号 select_deepseek_v4_mxfp4_moe_backend, convert_weight_to_mxfp4_moe_kernel_format）：核心变更文件。修正了 AITER MXFP4 MoE 权重转换中的解交织和 shuffle 逻辑，并翻转了后端优先级。
- vllm/model_executor/layers/fused_moe/layer.py（模块 MoE 层；类别 source；类型 data-contract；符号 _load_w13, _load_w2）：修正 TP 分片偏移计算，确保所有 rank 从 checkpoint 中加载正确的权重切片。影响所有使用 FusedMoE 的模型。

关键符号：select_deepseek_v4_mxfp4_moe_backend, convert_weight_to_mxfp4_moe_kernel_format, _load_w13, _load_w2

关键源码片段

vllm/model_executor/layers/fused_moe/oracle/mxfp4.py

核心变更文件。修正了 AITER MXFP4 MoE 权重转换中的解交织和 shuffle 逻辑，并翻转了后端优先级。

```
# vllm/model_executor/layers/fused_moe/oracle/mxfp4.py
# AITER_MXFP4_BF16 分支的权重转换（已修复版本）
elif mxfp4_backend == Mxfp4MoeBackend.AITER_MXFP4_BF16:
    from vllm._aiter_ops import rocm_aiter_ops # noqa: F401 # 保留导入避免未定义

    if w13_bias is not None:
        w13_bias = w13_bias.data.to(torch.float32)
    if w2_bias is not None:
        w2_bias = w2_bias.data.to(torch.float32)

    e, n, k = w13_weight.shape

    # 不再执行解交织：标准 _load_w13 已经输出 [gate_all, up_all] 布局
    # 直接使用 aiter 原生 shuffle 函数（匹配 aiter 测试脚本模式）
    from aiter.ops.shuffle import shuffle_weight as _shuf_w
    from aiter.utility.fp4_utils import e8m0_shuffle as _e8m0_shuf

    # w13（gate+up，stage1）：shuffle_weight 与 layout (16,16)
    w13_weight = torch.nn.Parameter(
        _shuf_w(w13_weight.data.view(torch.float4_e2m1fn_x2), (16, 16)),
        requires_grad=False,
    )
    shuffled_w13_scale = _e8m0_shuf(
        w13_weight_scale.view(-1, w13_weight_scale.shape[-1])
    )

    # w2（down-proj，stage2）：同样使用 shuffle_weight((16,16)) + e8m0_shuffle
    w2_weight = torch.nn.Parameter(
        _shuf_w(w2_weight.data.view(torch.float4_e2m1fn_x2), (16, 16)),
        requires_grad=False,
    )
```

```
    shuffled_w2_scale = _e8m0_shuf(
        w2_weight_scale.view(-1, w2_weight_scale.shape[-1])
    )
```

vllm/model_executor/layers/fused_moe/layer.py

修正 TP 分片偏移计算，确保所有 rank 从 checkpoint 中加载正确的权重切片。影响所有使用 FusedMoE 的模型。

```
# vllm/model_executor/layers/fused_moe/layer.py
# _load_w13 中修正的 TP 分片偏移部分
def _load_w13(
    self,
    expert_data: torch.Tensor,
    shard_dim: int,
    shard_id: str,
    loaded_weight: torch.Tensor,
    tp_rank: int,
    load_full: bool = False,
):
    # ... 计算 shard_size ...
    if not load_full and loaded_weight.ndim > 0:
        # 当参数被填充时（例如 MXFP4 round up intermediate_size），
        # shard_size 是填充后的大小。使用 * 未填充的 * 每 rank 大小
        # 来计算 checkpoint 偏移，确保每个 rank 落在正确的切片上。
        tp_size = self.moe_config.moe_parallel_config.tp_size
        loaded_per_rank = loaded_weight.shape[shard_dim] // tp_size
        start_offset = loaded_per_rank * tp_rank
        available = loaded_weight.shape[shard_dim] - start_offset
        if available <= 0:
            return
        narrow_size = min(loaded_per_rank, available)
        loaded_weight = loaded_weight.narrow(shard_dim, start_offset, narrow_size)
    # ... 继续加载到 expert_data ...
```

评论区精华

- CK MoE 兼容性疑虑：tjtanaa 询问新 shuffle 逻辑是否会破坏 CK MoE。BowenBao 分析认为旧逻辑处理的是 GPT-OSS 交织格式，新逻辑对非 GPT-OSS 模型应该有效，但建议测试更多模型，并提及曾提交相关 issue 但未被关注。
- 恢复被删注释和导入：tjtanaa 要求恢复 _load_w13/_load_w2 中被误删的注释，以及 AITER 分支中的 rocm_aiter_ops 导入，避免未定义错误。作者均予以采纳。
- 偏移逻辑影响评估：tjtanaa 表示需要对 TP 分片偏移计算逻辑变更的影响做进一步评估，最终批准仅表明基础功能验证通过。
  - 新 shuffle 逻辑对 CK MoE 的兼容性 (correctness): 新逻辑仅影响 AITER_MXFP4_BF16 分支，不影响 CK MoE；但整体 MXFP4 路径对不同检查点格式的兼容性仍需验证。
  - 恢复被删除的注释和导入 (style): 作者全部接受并修复。

- TP 分片偏移逻辑变更的潜在影响 (correctness): 最终 tjtanaa 批准，表明在 DeepSeek-V4 场景下验证通过，但后续仍需持续关注其他模型。

风险与影响

- 风险：
  1. 其他 MoE 后端兼容性：shuffle 逻辑变更目前只作用于 AITER_MXFP4_BF16 分支，不会直接影响 CK MoE 等方式，但若未来有混合后端路径需要考虑交互。
  2. TP 偏移抽象影响：_load_w13/_load_w2 的基础变化可能影响所有使用 padded 参数的 MoE 层。当前测试覆盖有限（仅手动验证了 DeepSeek-V4 一个模型），其他模型可能隐藏问题。
  3. 缺少单元测试：本次仅依赖手动启动验证，无新增自动化测试，回归依靠 CI。
  4. AITER 版本依赖：PR 需要 AITER 特定版本（main 分支 5104dca），若下游用户使用旧版可能出现符号缺失。 - 影响：用户端：ROCm 上 DeepSeek-V4 用户若未指定后端，默认从 triton_unfused 变为 aiter，可获得正确结果和 30%+ 的性能提升；若已指定 --moe-backend aiter，之前输出乱码的问题被修复。系统端：layer.py 中 TP 分片偏移计算的变化对所有使用 FusedMoE 的模型都有潜在影响（其他模型需验证）。团队协作：PR 作者与 reviewer 经过多轮迭代，确认了关键设计决策（优先级翻转、注释恢复），并引出了深层依赖（AITER 版本、CK MoE 兼容性），为后续持续优化奠定基础。 - 风险标记：核心路径变更，缺少自动化测试，AITER 版本依赖，其他后端兼容性

关联脉络

- PR #42982 [ROCm][Perf] DSv3.2 MI355X TP4 decode-step orchestration cleanup (3 micro-opts): 均为 ROCm + DeepSeek 性能优化相关，本 PR 修复后可使 AITER 后端发挥性能优势，与该 PR 的微优化协同。
- PR #43898 [ROCm][DSv4] Remove device pipeline stall in sparse attention: 同样针对 ROCm DSv4 的优化，本 PR 修复 MoE 权重加载后，稀疏注意力等模块可搭配使用。
- PR #43905 [DSv4] Move mHC tilelang kernels & Don't use CustomOP in dsv4/nvidia: 同为 DeepSeek-V4 的内核重构，本 PR 修复了权重路径，该 PR 重构了内核路径，共同完善 DSv4 在 ROCm 上的运行。