PR #49591 完整报告

vllm-project/vllm

[Bugfix][CPU] Zero-pad MoE intermediate size for grouped-gemm TP alignment

合并时间：2026-07-27 16:32

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/49591

执行摘要

- 一句话：CPU MoE 中间维零填充，修复 TP 分组 GEMM 静默回退
- 推荐动作：值得精读。PR 展示了完整的性能悬崖定位方法（profiling 定位 74% 时间片与 16 万次小 mm 调用），以及 padding 布局与内核切分契约的联动设计；fail-loud 策略对静默性能回归的取舍也值得参考。但合并 / 参考前必须处理 review 中提出的 ARM NEON 分支回归，并评估 AMX 上非 bf16 场景由回退变报错的影响。

功能与动机

PR body 明确指出：CPUFusedMOE.check_grouped_gemm 只在 per-partition moe_intermediate_size（即 moe_intermediate_size // tp_size）是 32 的倍数时才启用快速内核，否则静默回退到 cpu_fused_moe_torch 的 per-expert Python 循环。对 google/gemma-4-26B-A4B-it（intermediate=704，128 experts），tp=2 分片为 352（对齐走快速路径），tp=4 分片为 176（未对齐，静默回退），于是出现 "tp=4 throughput was lower than tp=2 despite more parallelism, with no indication anything had gone wrong"。Profiling 显示回退路径消耗 74% 的 CPU 时间，其中约 164k 次 aten::mm 调用（每次平均 56 微秒）主要是调度开销而非计算。

实现拆解

1. 定位性能悬崖：csrc/cpu/cpu_fused_moe.cpp 的 AMX/vector 分组 GEMM 内核将 expert 中间维按固定 32 宽块切分且无尾部处理，check_grouped_gemm 只允许 32 对齐的分片走快速路径，其余静默回退 forward_torch。
2. 引入填充入口：在 CPUFusedMOE.__init__ 首行新增调用 _pad_moe_intermediate_for_grouped_gemm(layer)，先于 check_grouped_gemm 执行。该方法做四层守卫：无 prepack_moe_weight 算子、ARM 架构、余数为 0、SWIGLUOAI 激活均直接返回；然后按 _MOE_GROUPED_GEMM_N_TILE = 32 计算 pad，重建 w13_weight、w2_weight 与可选 w13_bias。
3. 填充布局的关键设计：w13 输出维是 [gate 半块 | up 半块] 连续拼接，内核在固定偏移 output_size_13/2 处切分并施加 silu/gelu 等激活，因此零行必须插入两个半块之间（new_w13[:, :intermediate_size] 放 gate，new_w13[:, padded_size:padded_size+intermediate_size] 放 up），不能追加在末尾，否则会推后偏移破坏 up 半块；w2 只需在输入列尾部补零列。填充值精确为零，数学上不改变激活结果。

4. 失败模式改为响亮：check_grouped_gemm 重构后，在 AMX 支持的机器上把 bf16、四个维度的 32 对齐合并为一个条件，不满足则 raise RuntimeError(_grouped_gemm_alignment_error(layer))，替代原先的 return False, "none" 静默回退；非 AMX（NEON/vec）保持原回退行为不变。
5. 测试与验证配套：tests/kernels/moe/test_cpu_fused_moe.py 新增 _StubMoELayer 桩和 13 个用例：对 intermediate_size=176 覆盖 SILU/GELU/GELU_TANH × bias/no-bias，断言填充后命中 forward_grouped_gemm 且输出与未填充参考实现一致；另有一个 swigluoai 用例断言 AMX 上抛错、其他平台回退 forward_torch。无配置、schema 或部署配套改动。

关键文件：

- vllm/model_executor/layers/fused_moe/cpu_fused_moe.py（模块 MoE 内核；类别 source；类型 data-contract；符号 _pad_moe_intermediate_for_grouped_gemm, _grouped_gemm_alignment_error, check_grouped_gemm, init）：核心修复文件：新增零填充逻辑与报错策略，重构 check_grouped_gemm 的 AMX 分支，是性能悬崖与正确性契约的关键位置。
- tests/kernels/moe/test_cpu_fused_moe.py（模块 MoE 测试；类别 test；类型 test-coverage；符号 _StubMoELayer, init, test_cpu_fused_moe_unaligned_intermediate_size, test_cpu_fused_moe_unaligned_intermediate_size_swigluoai）：新增 13 个用例验证填充正确性与架构分支行为，是发现 ARM 回归的关键测试文件。

关键符号：_pad_moe_intermediate_for_grouped_gemm, _grouped_gemm_alignment_error, check_grouped_gemm, test_cpu_fused_moe_unaligned_intermediate_size, test_cpu_fused_moe_unaligned_intermediate_size_swigluoai

关键源码片段

tests/kernels/moe/test_cpu_fused_moe.py

新增 13 个用例验证填充正确性与架构分支行为，是发现 ARM 回归的关键测试文件。

```
# 模拟 gemma-4-26B-A4B-it 在 tp=4 下的分片结果：704 // 4 == 176（非 32 倍数）
UNALIGNED_INTERMEDIATE_DIM = 176

class _StubMoELayer(torch.nn.Module):
    """最小化 MoE 层桩，只暴露 CPUFusedMOE 读取/替换的成员。"""

    def __init__(self, w13_weight, w2_weight, activation, w13_bias=None, w2_bias=None):
        super().__init__()
        self.w13_weight = torch.nn.Parameter(w13_weight, requires_grad=False)
        self.w2_weight = torch.nn.Parameter(w2_weight, requires_grad=False)
        self.activation = activation
        if w13_bias is not None:
            self.w13_bias = torch.nn.Parameter(w13_bias, requires_grad=False)
        if w2_bias is not None:
            self.w2_bias = torch.nn.Parameter(w2_bias, requires_grad=False)
```

```
@pytest.mark.parametrize("expert_num", EXPERT_NUM)
@pytest.mark.parametrize("hidden_size", HIDDEN_DIM)
@pytest.mark.parametrize("use_bias", USE_BIAS)
@pytest.mark.parametrize("dtype", DTYPE)
@pytest.mark.parametrize("act", [MoEActivation.SILU, MoEActivation.GELU, MoEActivation.
GELU_TANH])
def test_cpu_fused_moe_unaligned_intermediate_size(
    default_vllm_config, expert_num, hidden_size, use_bias, dtype, act):
    """未对齐的 per-partition 中间维必须零填充走上 grouped-gemm 快速路径,
    且输出与未填充权重计算的参考实现一致。"""
    if current_platform.get_cpu_architecture() == CpuArchEnum.ARM:
        # 填充只作用于 x86 AMX / vector 内核，ARM 上跳过本用例
        pytest.skip("padding is only applied on the x86 AMX/vector kernels")

    set_random_seed(0)
    batch_size = 64
    intermediate_size = UNALIGNED_INTERMEDIATE_DIM
    topk_num = max(expert_num // 2, 1)
    up_dim = 2 * intermediate_size

    input = torch.randn((batch_size, hidden_size), dtype=dtype) / (0.5 * hidden_size**0.5)
    w13 = torch.randn((expert_num, up_dim, hidden_size), dtype=dtype) / (0.5 * hidden_size**0.
5)
    w2 = torch.randn((expert_num, hidden_size, intermediate_size), dtype=dtype) / (0.5 *
intermediate_size**0.5)
    router_logits = torch.randn((batch_size, expert_num), dtype=dtype)
    # ... 构造可选 bias 与 router 结果 ...

    # 先基于未 padding 的原始权重计算参考输出
    ref_output = ref_fused_moe(input, w13, w2, w13_bias, w2_bias, topk_weight, topk_ids, act)

    layer = _StubMoELayer(w13.clone(), w2.clone(), act, w13_bias, w2_bias)
    cpu_moe = CPUFusedMOE(layer)
    # 关键断言：填充后必须命中 grouped-gemm 快速路径，而非静默回退
    assert cpu_moe.forward_method == cpu_moe.forward_grouped_gemm

    output = cpu_moe.forward_method(layer, input, topk_weight, topk_ids, act, expert_num,
    False)
    torch.testing.assert_close(output, ref_output, atol=get_default_atol(output), rtol=get_default_
    rtol(output))
```

评论区精华

oops-oom 指出本 PR 让 Arm CPU Test 变红：新增的 swigluoai 用例在 ARM 上确定性失败。根因是 check_grouped_gemm 中 pre-existing 的 NEON 分支不检查 activation——supports_neon 为真且 bf16、输入维 %4==0 时直接返回 (True, "neon")，于是 SWIGLUOAI（intermediate=176 满足 %4==0 但不满足 %32==0）也被选中走 forward_grouped_gemm，而测试期望非 AMX 平台回退 forward_torch。该问题在本 PR 内

没有修复（仅 1 个 commit，HEAD 未包含 NEON 分支补丁）。jikunshang 对 PR 给出 APPROVED（无正文）。

- ARM NEON 分支未同步导致新增测试确定性失败 (testing): 未解决。PR 内只有 1 个 commit，NEON 分支未同步修改，也没有调整测试的 ARM 跳过条件；合入 main 后 Arm CPU Test 会持续变红，需要后续修复。

风险与影响

- 风险：
  1. ARM CI 回归（确定性）：check_grouped_gemm 的 NEON 分支未检查 activation，新增的 test_cpu_fused_moe_unaligned_intermediate_size_swigluoai 在 ARM（NEON）上必然失败，需要后续补丁同步修改 NEON 分支或调整测试的跳过条件。
  2. AMX 上的行为变更：原先 AMX 机器上任何不满足对齐 /bf16 条件的情况都静默回退 torch 循环，现在统一 raise RuntimeError。若用户在 AMX CPU 上用非 bf16（如 fp32）跑 MoE，或在 swigluoai + 未对齐分片下启动服务，会从“能跑但慢”变成“启动失败”，虽然有意的（阻止 3-4x 性能悬崖），但对既有用户是可见破坏。
  3. 填充开销：对 176 的 shard，最多补 31 列，浪费约 17.6% 的 GEMM 计算；对更大的 intermediate 占比迅速降低，相对 3.8x 收益可忽略，但内存占用会随 pad 行 / 列增加。
  4. 内核契约耦合：gate/up 之间插入零行的正确性依赖内核“固定偏移 output_size_13/2 切分半块”的约定，未来若 cpu_fused_moe.cpp 调整布局需同步此逻辑。
- 影响：影响范围限于 CPU 后端的 MoE 推理路径：
  - 用户侧：在 CPU + 多卡 TP 下跑 MoE 模型（如 gemma-4-26B-A4B-it）时，原先 tp=4 静默降速的现象被消除，吞吐提升约 3.8x；AMX 上无法填充的场景改为启动报错，用户需调整 --tensor-parallel-size。
  - 系统侧：显著降低 tp=4 时 CPU 占用（原 74% 时间消耗在 per-expert aten::mm 调度上）。
  - 团队侧：需要跟进 ARM 分支回归与 #43653（新增 swiglustep/relu2 CPU MoE 激活）的兼容性验证。整体影响程度中等，不涉及 GPU 路径。
  - 风险标记：ARM CI 回归未修复，核心路径变更（CPU MoE 路径选择），AMX 上回退改为抛错，非 bf16 dtype 在 AMX 上可能被误判报错

关联脉络

- PR #38833 [Model] Pad the MoE intermediate size for gemma-4-26B-A4B-it on ROCm: 同一模型同一参数（moe_intermediate_size=704）的同类对齐修复，但作用于 ROCm AITER CK GEMM 路径，不涉及 CPU 内核。
- PR #48624 [Bugfix] Zero-pad unaligned MoE intermediate for FlashInfer/CUTLASS NVFP4 and FP8: 同为 TP 切分导致 MoE 中间维未对齐的崩溃 / 回退修复，面向 GPU FlashInfer/CUTLASS 内核，与本 PR 的模式一致。
- PR #43653 [Feature][CPU] Add swiglustep/relu2 CPU MoE activations: 为 CPU MoE 新增激活（当前 open），其新激活使用与 silu/gelu 相同的连续半块布局，本 PR 的 != SWIGLUOAI 检查可继续兼容，需跟进验证。