PR #41100 完整报告

vllm-project/vllm

[ROCm][CI] Extended Fused MoE and FP8 MoE test support

合并时间：2026-08-19 23:19

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41100

执行摘要

- 一句话：扩展 ROCm 融合 MoE 与 FP8/FP4 测试，修复分布式失败传播。
- 推荐动作：值得精读，尤其是跨后端保留标量缩放 ABI 的设计（0-D 张量在 Triton 启动边界转指针）和分布式测试失败传播模式，对多平台 CI 测试框架设计有借鉴意义。

功能与动机

PR body 指出要让融合 MoE 层测试矩阵在 ROCm/MI355 上可用，必须修复其暴露的真实 ModelOpt FP8/FP4 失败，并让分布式子用例失败对 pytest 可见，避免父测试报告 PASSED 而子 rank 实际失败。

实现拆解

1. 分布式失败传播：修改 tests/kernels/moe/test_moe_layer.py 中的 _parallel_worker，在每个 rank 收集失败详情并写入临时报告文件（tempfile + failure_report_path），父进程启动逻辑在结束后汇总失败，确保任何子 rank 失败都会导致父 pytest 失败，而不是仅打印。同时保留原有 pass/fail 计数。
2. ROCm FP4 测试赋能：新增 on_gfx950() 辅助函数（从 vllm.platforms.rocm 导入），并在 is_valid_config 中允许 modelopt_fp4 在 gfx950 上运行。由于 ops.scaled_fp4_quant 在 ROCm 不可用，在 tests/kernels/moe/utils.py 中新增 _pack_e2m1_fp4 和 _scaled_fp4_quant_emulated，调用 ref_nvfp4_quant 生成 E2M1 值并手工打包，替代原生 ops。
3. FP8 激活缩放形状修复：在 vllm/model_executor/layers/fused_moe/fused_moe.py 的 invoke_fused_moe_triton_kernel 中，若 A_scale 是 0-D 张量，则 reshape(1) 再传给 Triton 内核，因为 Triton 将 0-D 参数当作标量，而内核按指针加载缩放。对应的 tests/quantization/test_fp8.py 新增测试确保 ModelOpt 静态 FP8 路径的输出缩放保持 0-D 标量。
4. MoRI 矩阵门控与 OAI Triton MoE 适配：在 test_moe_layer.py 中添加 MORI_BACKENDS 集合，并检查 VLLM_TEST_ENABLE_MORI_MOE_LAYER=1 且 AITER 融合 MoE 已启用、共享专家融合已禁用时才运行 MoRI 配置。在 test_modular_oai_triton_moe.py 中，将测试从 is_cuda 改为 is_cuda_alike，并对 ROCm 填充权重 / 输入到 CDNA4 缩放布局对齐（如 256/512 边界），输出再切回原始形状。

5. CI 配置扩展：在 .buildkite/test-amd.yaml 和 .buildkite/test_areas/kernels.yaml 中新增 / 调整 MI355 上的 kernels/moe 测试步骤、忽略列表和软失败组。

关键文件：

- tests/kernels/moe/test_moe_layer.py（模块 MoE 测试；类别 test；类型 test-coverage；符号 on_gfx950, _parallel_worker, is_valid_config, MORI_BACKENDS）：核心测试文件，实现分布式失败传播、gfx950 上 modelopt_fp4 支持以及 MoRI 门控，直接影响测试矩阵在 ROCm 上的可用性。
- tests/kernels/moe/utils.py（模块 测试工具；类别 test；类型 test-coverage；符号 _pack_e2m1_fp4, _scaled_fp4_quant_emulated）：新增 FP4 仿真量化路径，解决 ROCm 上无 ops.scaled_fp4_quant 的问题。
- vllm/model_executor/layers/fused_moe/fused_moe.py（模块 融合 MoE；类别 source；类型 data-contract；符号 invoke_fused_moe_triton_kernel）：核心源码修复，在 Triton 启动边界将 0-D 的 A_scale 重塑为 (1,)，保证 Triton 内核能通过指针加载缩放，同时保留后端无关的标量表示。
- tests/kernels/moe/test_moe.py（模块 MoE 测试；类别 test；类型 test-coverage；符号 test_triton_moe_launcher_passes_scalar_scale_as_pointer, FakeKernel）：新增测试验证 Triton 启动器会把 0-D 的 A_scale 转成 (1,) 指针，锁定了数据契约。
- tests/quantization/test_fp8.py（模块 FP8 测试；类别 test；类型 test-coverage；符号 test_static_fp8_moe_input_scales_remain_scalar）：新增测试确保 ModelOpt 静态 FP8 MoE 输入缩放保持 0-D 标量，保护后端无关的数据契约。
- .buildkite/test-amd.yaml（模块 CI 配置；类别 config；类型 configuration）：CI 配置，新增 MI355 上 kernels/moe 测试步骤与忽略列表，是 ROCm 测试矩阵落地的关键。
- .buildkite/test_areas/kernels.yaml（模块 CI 配置；类别 config；类型 configuration）：CI 测试区域配置，新增软失败测试组，用于标记 MI355 上已知不稳定的 MoE 后端组合。

关键符号：on_gfx950, _pack_e2m1_fp4, _scaled_fp4_quant_emulated, test_triton_moe_launcher_passes_scalar_scale_as_pointer, test_static_fp8_moe_input_scales_remain_scalar, invoke_fused_moe_triton_kernel, _parallel_worker

关键源码片段

tests/kernels/moe/utils.py

新增 FP4 仿真量化路径，解决 ROCm 上无 ops.scaled_fp4_quant 的问题。

```
def _pack_e2m1_fp4(fp4_values: torch.Tensor) -> torch.Tensor:
    # 将 E2M1 格式的 FP4 值按两个一组打包进一个 uint8 字节。
    # 假设最后一个维度是偶数长度，交替取低位和高位 4 bit。
    assert fp4_values.shape[-1] % 2 == 0

    abs_values = fp4_values.abs()
    codes = torch.empty_like(abs_values, dtype=torch.uint8)
    # E2M1 只有 8 个正数值，逐个映射为 3 bit 编码
    for code, value in enumerate((0.0, 0.5, 1.0, 1.5, 2.0, 3.0, 4.0, 6.0)):
```

```
        codes[abs_values == value] = code
    # 符号位放入第 4 bit
    codes = codes | ((fp4_values < 0).to(torch.uint8) << 3)
    # 奇数位置的值移到高 4 bit，与低 4 bit 合并
    return codes[..., 0::2] | (codes[..., 1::2] << 4)


def _scaled_fp4_quant_emulated(
    w: torch.Tensor, w_gs: torch.Tensor
) -> tuple[torch.Tensor, torch.Tensor]:
    # ROCm 上没有 ops.scaled_fp4_quant，因此用 NVFP4 参考量化路径
    # 生成 FP4 值，再手工打包成硬件需要的布局。
    fp4_values, w_s = ref_nvfp4_quant(w, w_gs, block_size=16)
    return _pack_e2m1_fp4(fp4_values), w_s.to(torch.float8_e4m3fn)
```

vllm/model_executor/layers/fused_moe/fused_moe.py

核心源码修复，在 Triton 启动边界将 0-D 的 A_scale 重塑为 (1,)，保证 Triton 内核能通过指针加载缩放，同时保留后端无关的标量表示。

```
    # Triton 将 0-D 张量参数当作标量值，但内核通过指针加载张量级缩放。
    # 因此，在启动前将 0-D 的 A_scale 重塑为形状 (1,) 的张量，
    # 这样 Triton 内核拿到的是一个可解引用的指针，而不是标量。
    if A_scale is not None and A_scale.ndim == 0:
        A_scale = A_scale.reshape(1)

    fused_moe_kernel[grid](
        A,
        B,
        C,
        B_bias,
        A_scale,
        B_scale,
        topk_weights,
        sorted_token_ids,
        expert_ids,
        num_tokens_post_padded,
        B.size(1),
        B.size(2),
        EM,
        num_tokens,
        A.stride(0),
        A.stride(1),
        B.stride(0),
        B.stride(2),
        B.stride(1),
        C.stride(1),
        C.stride(2),
        A_scale.stride(0) if A_scale is not None and A_scale.ndim == 2 else 0,
        A_scale.stride(1) if A_scale is not None and A_scale.ndim == 2 else 0,
```

```
        B_scale.stride(0) if B_scale is not None and B_scale.ndim >= 2 else 0,
        B_scale.stride(2) if B_scale is not None and B_scale.ndim == 3 else 0,
        B_scale.stride(1) if B_scale is not None and B_scale.ndim >= 2 else 0,
        B_bias.stride(0) if B_bias is not None else 0,
        B_bias.stride(1) if B_bias is not None else 0,
        0 if block_shape is None else block_shape[0],
        0 if block_shape is None else block_shape[1],
    )
```

tests/kernels/moe/test_moe.py

新增测试验证 Triton 启动器会把 0-D 的 A_scale 转成 (1,) 指针，锁定了数据契约。

```
def test_triton_moe_launcher_passes_scalar_scale_as_pointer(monkeypatch) -> None:
    # 用 FakeKernel 拦截 fused_moe_kernel 的调用，
    # 捕获第 5 个位置参数（即实际传给 Triton 的 A_scale）。
    captured: dict[str, torch.Tensor] = {}

    class FakeKernel:
        def __getitem__(self, grid):
            def launch(*args, **kwargs) -> None:
                captured["a_scale"] = args[4]

            return launch

    monkeypatch.setattr(fused_moe_module, "fused_moe_kernel", FakeKernel())

    a_scale = torch.tensor(0.5)
    fused_moe_module.invoke_fused_moe_triton_kernel(
        A=torch.ones((1, 1)),
        B=torch.ones((1, 1, 1)),
        C=torch.empty((1, 1, 1)),
        A_scale=a_scale,
        B_scale=torch.ones(1),
        topk_weights=torch.ones((1, 1)),
        sorted_token_ids=None,
        expert_ids=torch.zeros(1, dtype=torch.int32),
        num_tokens_post_padded=torch.ones(1, dtype=torch.int32),
        mul_routed_weight=True,
        top_k=1,
        config={"BLOCK_SIZE_M": 1, "BLOCK_SIZE_N": 1, "BLOCK_SIZE_K": 1},
        compute_type=tl.float32,
        use_fp8_w8a8=True,
        use_int8_w8a8=False,
        use_int8_w8a16=False,
        use_int4_w4a16=False,
        per_channel_quant=False,
    )

    captured_scale = captured["a_scale"]
```

```
# 验证 0-D 的 A_scale 被 reshape 为 (1,) 且共享同一数据指针，
# 确保 Triton 能按指针加载缩放。
assert a_scale.ndim == 0
assert captured_scale.shape == (1,)
assert captured_scale.data_ptr() == a_scale.data_ptr()
```

评论区精华

审查中 tjtanaa 指出对非 ROCm 平台的行为变化（参数化组合缩减），建议仅在 ROCm 上跳过某些组合以保留 CUDA 覆盖，AndreasKaratzas 回复“已回退”并确认。另有关于 CI 标签命名的 NITS，建议将 (2xB2002xMI355) 改为 (2xB200-2xMI355)，已修改。

- 非 ROCm 平台测试组合变化 (testing): AndreasKaratzas 回复已回退，修改为仅 ROCm 平台跳过特定组合。
- CI 标签命名 NITS (style): AndreasKaratzas 回复 Done，已修改。

风险与影响

- 风险：分布式失败传播依赖临时报告文件，若文件清理或命名冲突可能导致误报；FP4 仿真使用参考实现，可能与真实硬件量化存在数值偏差；A_scale 重塑为 (1,) 后，如果其他后端（如 FlashInfer）依赖 0-D 标量 ABI，需确认后续路径不会误用形状；MoRI 矩阵门控依赖环境变量，若未设置则测试静默跳过，可能遗漏问题；CI 配置新增测试步骤可能增加运行时长，但已放入软失败。
- 影响：对用户无直接功能影响，主要影响 ROCm 平台的 CI 测试覆盖与可靠性；对系统而言，分布式 MoE 层测试失败现在能真实反映到 pytest 结果，避免误判；对团队，需要维护新增的环境变量约定和 CI 配置，并关注 MI355 上检测到的真实 ModelOpt 量化问题。
- 风险标记：分布式失败传播依赖临时文件，FP4 仿真与硬件实现存在偏差，A_scale 形状重塑影响多后端 ABI，MoRI 矩阵依赖环境变量门控

关联脉络

- PR #46434 [ROCm][CI] Enable modular OAI Triton MoE tests: 同一测试文件 test_modular_oai_triton_moe.py，本 PR 进一步将其扩展到 CUDA-alike 平台并处理 ROCm 布局对齐。
- PR #51632 [ROCm] [Bugfix] Fix Triton fused shared expert alignment: 同属 ROCm MoE 测试稳定性修复，且本 PR 的 MoRI 门控也涉及 shared expert 融合禁用逻辑。
- PR #52966 [Bugfix][Quantization] Support CT block FP8 with Marlin: 同属 FP8 量化测试覆盖扩展，与本 PR 的 FP8 缩放形状测试存在关联。