PR #51877 完整报告

vllm-project/vllm

[ROCm][CI] Speed Up ROCm Skinny GEMM Tests (reduced parameterizations,

合并时间：2026-08-12 09:40

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/51877

执行摘要

- 一句话：加速 ROCm Skinny GEMM 测试，参数化从 11040 降到 2644
- 推荐动作：值得精读。该 PR 展示了两个可迁移的技巧：一是通过分析参数之间的独立性，用配对采样替代完整笛卡尔积，在不显著损失覆盖的前提下将用例数量级压缩；二是用 skip_global_cleanup + 模块级 autouse fixture 把高频的元成本从“每用例”降为“每模块”。对 kernel 测试维护者和 CI 效率优化者均有直接参考价值。

功能与动机

PR body 明确指出：'This PR reduces the test_rocm_skinny_gemm suite from a ~2 hour runtime to about 10 seconds by reducing the number of parameterizations from 11040 to 2644, and removing the unnecessary environment cleanup between tests. We now just cleanup the environment one time at the end of the entire module rather than needlessly eating the 0.3s cost to cleanup after each parameterization.' 即原始测试因多因子笛卡尔积导致用例爆炸，且每个参数化组合都会触发一次环境清理，叠加后成为主要耗时来源。

实现拆解

实现分为四个步骤：

1. 参数化策略重构：在 tests/kernels/quantization/test_rocm_skinny_gemms.py 中新增 OPTIONS_WVSPLITKRC、OPTIONS_WVSPLITK、OPTIONS_WVSPLITK_FP8 三个配对选项列表，将 dtype、padded_a、padded_b、bias_mode、biased、xnorm 等原先各自独立 parametrize 的因子组合为 6 组针对性覆盖。代码注释说明这些选项在 kernel 内相互独立，因此只需覆盖每对组合而非完整笛卡尔积，这是从 11040 降到 2644 的关键。
2. 抽取 make_bias 辅助函数：三个测试函数中重复的 bias 生成逻辑被统一到 make_bias(bias_mode, n, m, dtype)，按 bias_mode 返回 None、(m,) 向量或 (n, m) 矩阵，同时移除了原代码中从未被 BIAS_MODES 引用的 bias_mode == 3 分支，减少冗余。
3. 环境清理后置：设置模块级 pytestmark = pytest.mark.skip_global_cleanup 跳过 pytest 默认的每用例全局清理，并新增 scope="module", autouse=True 的 fixture cleanup_after_all_tests，在模块内全部用例结束后统一调用 cleanup_dist_env_and_memory()。这样把约 0.3s/ 次的清理成本从 2644 次叠加中消除，只付出一次。

4. CI 配置配套：.buildkite/test-amd.yaml 中 mi300 与 mi355 两处 Kernels Quantization Test 任务移除 parallelism: 2，命令从带 --shard-id/--num-shards 的并行调用改为单条 pytest -v -s kernels/quantization。由于测试时长已降至单任务可承受范围，无需再为它维护并行分片逻辑。

关键文件：

- tests/kernels/quantization/test_rocm_skinny_gemms.py（模块 内核测试；类别 test；类型 test-coverage；符号 cleanup_after_all_tests, make_bias, test_rocm_wvsplitkrc_kernel, test_rocm_wvsplitk_kernel）：核心变更文件。通过配对选项列表将参数化组合从 11040 缩减到 2644，并将每用例环境清理改为模块末尾一次性执行，是提速的关键。
- .buildkite/test-amd.yaml（模块 CI 配置；类别 config；类型 configuration）：配合测试提速移除 ROCm Quantization 测试任务的并行分片，避免为已降速到 10 秒级的短任务维护并行逻辑，属于配套 CI 配置调整。

关键符号：cleanup_after_all_tests, make_bias, test_rocm_wvsplitkrc_kernel, test_rocm_wvsplitk_kernel, test_rocm_wvsplitk_fp8_kernel

关键源码片段

tests/kernels/quantization/test_rocm_skinny_gemms.py

核心变更文件。通过配对选项列表将参数化组合从 11040 缩减到 2644，并将每用例环境清理改为模块末尾一次性执行，是提速的关键。

```
# 将 kernel 的独立选项按配对组合，避免全量笛卡尔积。
# bias_mode: 0 = 无 bias, 1 = (m,) 向量 , 2 = (n, m) 矩阵。
OPTIONS_WVSPLITKRC = [
    # dtype, padded_a, bias_mode, xnorm
    (torch.float16, False, 0, False),
    (torch.float16, True, 1, True),
    (torch.float16, True, 2, False),
    (torch.bfloat16, True, 0, True),
    (torch.bfloat16, False, 1, False),
    (torch.bfloat16, False, 2, True),
]

# 整个模块只清理一次：pytest 默认的每用例清理约 0.3s,
# 在 2644 个用例下会累积成不可接受的额外开销。
pytestmark = pytest.mark.skip_global_cleanup


@pytest.fixture(scope="module", autouse=True)
def cleanup_after_all_tests():
    yield
    cleanup_dist_env_and_memory()

def make_bias(bias_mode, n, m, dtype):
    # 统一三种 bias 形态的生成逻辑，供多个测试复用。
```

```
    if bias_mode == 0:
        return None
    shape = (m,) if bias_mode == 1 else (n, m)
    return torch.rand(shape, dtype=dtype, device="cuda") * 2 - 1


@pytest.mark.parametrize("n", N_FACTORS_WVSPLITKRC)
@pytest.mark.parametrize("k", K_FACTORS_WVSPLITKRC)
@pytest.mark.parametrize("m", M_FACTORS_WVSPLITKRC)
@pytest.mark.parametrize("dtype,padded_a,bias_mode,xnorm", OPTIONS_WVSPLITKRC)
@pytest.mark.parametrize("seed", SEEDS)
@pytest.mark.skipif(not current_platform.is_rocm(), reason="only test for rocm")
@pytest.mark.skipif(not on_gfx950(), reason="only meant for gfx950")
def test_rocm_wvsplitkrc_kernel(n, k, m, dtype, padded_a, bias_mode, xnorm, seed):
    # 根据 CU 数量与 K 分片宽度判断该 (N, K, M) 是否适用 wvSplitKrc 内核，
    # 否则跳过，避免对不适用尺寸做无效校验。
    torch.manual_seed(seed)
    cu_count = num_compute_units()
    N_p2 = 1 << (n - 1).bit_length()
    rndup_cus = ((m + 64 - 1) // 64) * ((k + 512 - 1) // 512)
    GrpsShrB = min(N_p2 // 16, 4)
    CuNeeded = rndup_cus * GrpsShrB
    fits_wvsplitkrc = (N_p2 * m * ((k + 512 - 1) // 512)) <= 128 * 1024 * 12
    fits_wvsplitkrc &= CuNeeded <= cu_count
    if not fits_wvsplitkrc:
        pytest.skip("Too large for wvSplitKrc")

    xavier = math.sqrt(2 / k) if xnorm else 1
    A = torch.randn(n, k, dtype=dtype, device="cuda") * xavier
    B = torch.randn(m, k, dtype=dtype, device="cuda") * xavier
    if padded_a:
        A = pad_fp8(A)
    BIAS = make_bias(bias_mode, n, m, dtype)

    ref_out = torch.nn.functional.linear(A, B, BIAS)
    out = ops.wvSplitKrc(A, B, cu_count, BIAS)

    if xnorm:
        # bf16 的 1 ULP 约 3.9e-3，带 bias 时放宽 atol 到 5e-3。
        atol = 5e-3 if (dtype == torch.bfloat16 and BIAS is not None) else 1e-3
        torch.testing.assert_close(out, ref_out, atol=atol, rtol=1e-8)
    else:
        torch.testing.assert_close(out, ref_out, atol=1e-3, rtol=1e-2)
```

评论区精华

该 PR 来自 fork，claude[bot] 提示自动 review 被禁用，维护者 AndreasKaratzas 直接给出 LGTM 并批准，没有公开的技术争议。核心设计决策体现在 PR body 与代码注释中：一是“选项在 kernel 内独立，配对覆盖已足够”，二是“全局清理比每用例清理更划算”。提交历史中的

reinstate parallelism for gating tg 说明作者在合并前对 gating 类任务做了一次并行性回收调整，但最终 diff 中该配置未再引入并行。

- 减少参数化组合是否影响测试覆盖 (design): 接受独立配对假设，将组合数降至 2644，测试时间从约 2 小时降至约 10 秒。
- 环境清理从每用例后置改为模块末尾一次 (testing): 模块级清理显著降低开销，且测试用例相互独立，风险可控。

风险与影响

- 风险:
  1. 参数化覆盖减少：从全笛卡尔积 11040 降到配对采样 2644，依赖“选项相互独立”的假设。例如 OPTIONS_WVSPLITK_FP8 只覆盖 6 组 (padded_a, padded_b, biased) 配对而非全部 8 种，若某对组合存在隐藏的边界交互（如 padded_a=True 与 padded_b=True 同时出现时的 FP8 路径），可能漏测。
  2. 全局清理引入状态泄漏风险：skip_global_cleanup 使测试间不再清理分布式环境与显存，若单个用例异常残留张量或通信状态，可能导致后续用例级联失败或 OOM；虽然测试用例相互独立可降低概率，但失败定位会更困难。
  3. CI 并行策略调整：.buildkite/test-amd.yaml 移除 parallelism: 2 后，单任务需在一个 agent 上跑完整个 kernels/quantization 目录，若未来新增参数化规模扩大，180 分钟超时可能重新成为瓶颈。 - 影响：对开发者：ROCm CI 的 Quantization 内核验证从约 2 小时缩短到约 10 秒，MI300/MI355 相关工作流的反馈周期显著缩短，CI 资源占用大幅下降。对用户：无任何运行时行为影响，纯测试与 CI 配置变更。对团队：建立了“基于独立性配对参数化 + 模块级清理”的测试加速模式，可复制到其他长尾 kernel 测试套件，同时提供了参数化精简时如何评估覆盖风险的参考案例。 - 风险标记：参数化覆盖减少，全局清理状态泄漏风险，CI 并行分片移除

关联脉络

- PR #50654 [ROCm][Perf] Kimi-K3 Fused kernel for KDA decode: 同为 ROCm 内核层性能与测试迭代，依赖类似的 kernel 参数化验证方式，代表 ROCm 内核测试套件持续演进方向。
- PR #50268 [Hardware][AMD] Enable fused bf16→fp32 router GEMM on ROCm: 同为 ROCm 内核测试与实现配套调整，展示了 ROCm 内核测试覆盖策略的演进。
- PR #51652 [CI/Build] Use file rendezvous for local distributed tests: 同属 CI 基础设施优化，都在提升测试执行的效率与稳定性，方向一致。