

PR #51968 完整报告

vllm-project/vllm

[XPU][Tests] Make tests device-agnostic

合并时间: 2026-08-20 17:25

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/51968>

执行摘要

- 一句话: 8 套 Triton 内核测试设备无关化, 补齐 XPU 覆盖
- 推荐动作: 值得精读。本 PR 是 vLLM 测试基础设施跨平台化的重要范本, 有三类要点值得学习: 一是设备无关测试的统一门控模式 (如何在保留 CUDA 算力细分能力的同时放开 XPU 执行); 二是 CUDA 专用 API 到 torch.accelerator 通用 API 的迁移路径 (is_current_stream_capturing 的等价改写及讨论中关于 serving 与测试环境差异的解释); 三是共享 fixture 前置依赖的识别方法 (通过实测数据定位 workspace_init 是解锁批量 MoE 测试的前提而非附带改动)。

功能与动机

RFC #48480 明确指出: vLLM 大量 Triton 内核测试被 pinned 到 CUDA, 即使内核本身可移植 (XPU/ROCm 能编译运行), 也会因硬编码 device="cuda"、调用 CUDA-only host API (如 torch.cuda.get_device_capability、torch.cuda.is_current_stream_capturing) 而报错或被直接 skip, 导致这些内核在非 NVIDIA 后端完全无隔离测试覆盖。PR body 描述 8 个套件硬编码 device="cuda" 或门控在 torch.cuda.is_available() / is_cuda() / has_device_capability() 上, 尽管其中 Triton 内核在 XPU 上已能构建运行。

实现拆解

1. 确立统一的设备无关模式: 每个测试文件顶部定义 DEVICE = current_platform.device_type, 模块级 pytestmark 门控从 is_cuda_alike() 放宽为 is_cuda_alike() or is_xpu(); has_device_capability() 类的算力门控只在 is_cuda_alike() 为真时检查, 避免在 XPU 上误触发。该模式在 test_merge_attn_states.py、test_batched_moe.py、test_triton_moe_no_act_mul.py、test_fused_recurrent_packed_decode.py 中一致落地, 引用、容差与 CUDA 行为均不改变。
2. 按 commit 逐个套件改造: commit 1 改造 tests/kernels/test_fused_recurrent_packed_decode.py (gated delta rule packed decode, 去掉 torch.cuda.is_available() 装饰器, device 改用 DEVICE); commit 2 改造 tests/kernels/attention/test_merge_attn_states.py (11 处 device 替换, merge_attn_states_cuda 更名为 merge_attn_states_native, markdown 表格列名同步改名); commit 3 处理 MoE 三件套 test_batched_moe.py、test_count_expert_num_tokens.py、test_triton_moe_no_act_mul.py 及共享工具 tests/kernels/moe/utils.py、quant_utils.py (该 commit 是 #43407 的 rebase);

commit 4 处理 test_silu_mul_per_token_group_quant_fp8_colmajor.py 与 quantization/test_per_token_group_quant.py; commit 5 处理 mamba/test_replayssm_prefill_decode_equivalence_mamba2.py。

3. 生产代码配套改造: vllm/model_executor/layers/fused_moe/experts/fused_batched_moe.py 的 `_is_capturing_or_compiling()` 从 `current_platform.is_cuda_alike()` and `torch.cuda.is_current_stream_capturing()` 改为 `torch.accelerator.current_stream().is_capturing()`。原因是 XPU 的 torch 没有 CUDA 流捕获 API, 不改则 XPU graph capture 期间会在 host 端错误读取 `expert_num_tokens`。该改动对 CUDA 是行为等价的 (review 中由 guangyey 确认 accelerator API 的等价性)。
4. 共享 fixture 前置修复: tests/conftest.py 的 `workspace_init` fixture 把有状态的 `torch.cuda.is_available()` 换成无状态的 `torch.accelerator.is_available()`, 并改用 `torch.device(0)`; 这是 `test_batched_moe.py` 在 XPU 解锁的前提 (否则 288 failed / 104 passed, 全部是 WorkspaceManager not initialized 断言)。
5. 验证与配套: PR body 给出 XPU (Intel Arc Pro B70, torch 2.13.0+xpu) 与 CUDA (L40S sm_89) 的 A/B 实测表, 所有套件从 skip/fail 变为通过, CUDA 前后数字完全一致; 同时运行 `moe/test_moe.py::test_fused_moe` 作为共享 fixture 副作用检查。无 CI 配置、schema 或部署改动。

关键文件:

- vllm/model_executor/layers/fused_moe/experts/fused_batched_moe.py (模块 专家内核; 类别 source; 类型 core-logic; 符号 `_is_capturing_or_compiling`): 唯一的生产代码改动。`_is_capturing_or_compiling()` 从 CUDA 专用 API 迁移到 `torch.accelerator` 通用 API, 修复 XPU 上 CUDA graph capture 期间 host 端错误读取 `expert_num_tokens` 的问题, 且是 review 讨论的焦点。
- tests/conftest.py (模块 测试夹具; 类别 test; 类型 infrastructure; 符号 `workspace_init`): 共享 fixture `workspace_init` 改用无状态 `accelerator` 检查, 是 `test_batched_moe.py` 在 XPU 从 288 failed 变为通过的前置条件, 影响所有依赖该 fixture 的测试。
- tests/kernels/attention/test_merge_attn_states.py (模块 注意力合并; 类别 test; 类型 test-coverage; 符号 `merge_attn_states_torch`, `test_merge_attn_states`, `test_merge_attn_states_both_empty`, `generate_markdown_table`): 改造量最大的套件: 2592 skip + 6 fail 变为 2598 passed; `merge_attn_states_cuda` 更名为 `merge_attn_states_native`, 是设备无关化模式的完整示范。
- tests/kernels/moe/test_batched_moe.py (模块 批量 MoE; 类别 test; 类型 test-coverage; 符号 `BatchedMMTensors.make_tensors`, `test_batched_mm`, `test_fused_moe_batched_experts`): MoE 主力套件, 96 failed + 672 skipped 变为 392 passed + 384 skipped; float8 用例的算力门控被条件化到仅 CUDA-alike 平台。
- tests/kernels/moe/test_triton_moe_no_act_mul.py (模块 MoE 激活; 类别 test; 类型 test-coverage; 符号 `make_test_tensors`, `test_triton_experts_no_mul_activation`, `test_workspace_shapes_no_mul_vs_gated`, `test_adjust_n_for_activation`): 门控重构的典型代表: 三个逐用例 `has_device_capability(80)` 装饰器合并为模块级 `pytestmark` 且仅在 CUDA-alike 上检查, 85 skipped 变为 87 passed + 11 skipped。

- tests/kernels/test_fused_recurrent_packed_decode.py (模块 门控增量解码; 类别 test; 类型 test-coverage; 符号 test_fused_recurrent_packed_decode_matches_reference, test_packed_decode_supports_large_batch_head_grid) : 首个 commit, 去掉 torch.cuda.is_available() 门控, gated delta rule packed decode 内核与参照实现获得 XPU 覆盖 (6 skipped 变为 6 passed) 。
- tests/kernels/moe/utils.py (模块 测试工具; 类别 test; 类型 test-coverage; 符号 batched_moe, make_quantized_test_activations, make_test_weights, naive_batched_moe) : MoE 多套件共享工具, device 参数默认值从 "cuda" 改为 DEVICE, 是第 3 个 commit 的配套基础。

关键符号: _is_capturing_or_compiling, workspace_init, BatchedMMTensors.make_tensors, test_batched_mm, merge_attn_states_torch, test_merge_attn_states_both_empty, make_test_tensors, test_fused_recurrent_packed_decode_matches_reference, batched_moe

关键源码片段

vllm/model_executor/layers/fused_moe/experts/fused_batched_moe.py

唯一的生产代码改动。_is_capturing_or_compiling() 从 CUDA 专用 API 迁移到 torch.accelerator 通用 API, 修复 XPU 上 CUDA graph capture 期间 host 端错误读取 expert_num_tokens 的问题, 且是 review 讨论的焦点。

```
def _is_capturing_or_compiling() -> bool:
    # 旧实现只在 CUDA-alike 平台上调用 torch.cuda.is_current_stream_capturing(),
    # XPU 的 torch 没有该 API, 且 XPU graph capture 路径上该分支永远为 False,
    # 导致 capture 期间在 host 端直接读取 expert_num_tokens, 行为错误。
    # torch.accelerator.current_stream() 会分发到当前激活的 accelerator 后端,
    # 因此在 CUDA、ROCm、XPU 上都能正确反映 stream capture 状态。
    return (
        torch.compiler.is_compiling()
        or torch.accelerator.current_stream().is_capturing()
    )
```

tests/conftest.py

共享 fixture workspace_init 改用无状态 accelerator 检查, 是 test_batched_moe.py 在 XPU 从 288 failed 变为通过的前置条件, 影响所有依赖该 fixture 的测试。

```
@pytest.fixture
def workspace_init():
    """Initialize the workspace manager for tests that need it.
    用当前平台的 accelerator 设备初始化 workspace manager, 测试结束后复位。
    创建完整 vLLM engine 的测试不应使用此 fixture,
    因为 engine 会自行初始化 workspace manager。
    """
    from vllm.v1.worker.workspace import (
        init_workspace_manager,
        reset_workspace_manager,
    )
```

```

# torch.cuda.is_available() 是有状态调用，且只认识 CUDA；
# torch.accelerator.is_available() 是其文档规定的无状态等价写法，
# 在 XPU / ROCm 上同样正确。这是 test_batched_moe.py 在 XPU
# 从 288 failed 变为通过的前置条件。
if torch.accelerator.is_available():
    init_workspace_manager(torch.device(0))

yield

reset_workspace_manager()

```

tests/kernels/attention/test_merge_attn_states.py

改造量最大的套件：2592 skip + 6 fail 变为 2598 passed；`merge_attn_states_cuda` 更名为 `merge_attn_states_native`，是设备无关化模式的完整示范。

```

import pytest
import torch

from vllm._custom_ops import (
    merge_attn_states as merge_attn_states_native,
)
from vllm.platforms import current_platform
from vllm.v1.attention.ops.triton_merge_attn_states import (
    merge_attn_states as merge_attn_states_triton,
)

# 设备从 current_platform 解析，而不是硬编码 "cuda"；
# 这样同一份测试可以在 CUDA、ROCm、XPU 上运行。
DEVICE = current_platform.device_type

# 门控从 is_cuda_alike() 放宽为 "CUDA-alike 或 XPU"，
# 因为 native 与 Triton 实现都是纯 Triton，在 XPU 上可运行。
pytestmark = [
    pytest.mark.skip_global_cleanup,
    pytest.mark.skipif(
        not (current_platform.is_cuda_alike() or current_platform.is_xpu()),
        reason="merge_attn_states kernels require CUDA, ROCm or XPU.",
    ),
]

@pytest.mark.parametrize(
    "merge_fn", [merge_attn_states_native, merge_attn_states_triton]
)
@pytest.mark.parametrize("output_dtype", [torch.float32, torch.half, torch.bfloat16])
def test_merge_attn_states_both_empty(merge_fn, output_dtype) -> None:
    # 覆盖两侧 LSE 都为 -inf 的空 token 场景，确保 0/0 softmax
    # 归一化不会在合并结果中产生 NaN。
    num_tokens, num_heads, head_size = 6, 8, 128

```

```

prefix_output = torch.zeros(
    num_tokens, num_heads, head_size, device=DEVICE, dtype=output_dtype
)
prefix_lse = torch.randn(num_heads, num_tokens, device=DEVICE)
suffix_output = torch.zeros(
    num_tokens, num_heads, head_size, device=DEVICE, dtype=output_dtype
)
suffix_lse = torch.randn(num_heads, num_tokens, device=DEVICE)

# Tokens 2 and 3 are empty on both sides.
empty = slice(2, 4)
prefix_lse[:, empty] = float("-inf")
suffix_lse[:, empty] = float("-inf")

output = torch.empty_like(prefix_output)
merge_fn(output, prefix_output, prefix_lse, suffix_output, suffix_lse)

assert not output.isnan().any()

```

评论区精华

fused_batched_moe.py 的改动是讨论焦点。jikunshang 提议该改动是否可以等待 PR #50526 落地；pmanczak 解释：裸调用 `torch.cuda.is_current_stream_capturing()` 在 serving 场景可行（model runner 安装了别名），但在 XPU kernel 测试中没有任何安装别名，会直接崩溃。guangyey 随后给出官方等价写法 `torch.accelerator.current_stream().is_capturing()`，jikunshang cc guangyey 将该能力补进 accelerator API 层。tests/confstest.py 上 guangyey 建议用 `torch.accelerator.is_available() + torch.device(0)`，pmanczak 接受。test_silu_mul 文件被 jikunshang 指出缺少统一 `DEVICE` 常量，pmanczak 补齐。

- fused_batched_moe 的 graph capture 守卫是否等待 #50526 或使用 accelerator API (design): 采用 `torch.compiler.is_compiling()` or `torch.accelerator.current_stream().is_capturing()` 写法，CUDA 行为不变，同时解锁 XPU 对 graph capture 的正确感知。
- workspace_init 的设备解析写法 (design): 采用 `torch.accelerator.is_available() + torch.device(0)` 的无状态写法，覆盖 CUDA、ROCm、XPU。
- test_silu_mul_per_token_group_quant_fp8_colmajor 缺少 `DEVICE` 常量 (style): 文件统一为 `DEVICE` 常量模式，与其它套件保持一致。

风险与影响

- 风险：
 1. 生产守卫路径 API 迁移： `_is_capturing_or_compiling()` 位于 CUDA graph capture 关键路径，`torch.accelerator.current_stream().is_capturing()` 是较新的 torch API，若用户使用旧版 torch（accelerator API 引入前）可能导致 `AttributeError`；需要确认 vLLM 最低 torch 版本覆盖。CUDA 上语义等价性已由维护者确认，但建议在 CI 中覆盖多 torch 版本。

2. 共享 fixture 全局影响: tests/conftest.py 的 workspace_init 影响所有依赖它的测试; torch.accelerator.is_available() 与 torch.cuda.is_available() 在纯 CPU 环境及多后端共存时的语义差异未被显式测试覆盖, torch.device(0) 指向的 accelerator 需与后续 kernel 调用保持一致。
3. XPU 覆盖仍不完整: test_batched_moe.py 在 XPU 为 392 passed / 384 skipped, 而 CUDA 为 388 / 388, skip 差异来自 sm_89 tensor-descriptor 门控; test_moe.py::test_fused_moe 在 XPU 仍有 56 个用例失败 (两个最大 FUSED_MOE_MNK_FACTORS 形状因 bf16 累加顺序超出 atol=2e-2), PR 明示不声称启用, 后续需要数值对齐工作。
4. 残留 CUDA 硬编码: 如 test_triton_moe_no_act_mul.py 的 test_supported_apply_moe_activation_executes 仍保留 device="cuda" 和 is_cuda() 门控, 说明套件内仍有部分用例未完全设备无关, 可能被后续平台遗漏。- 影响: 对 XPU 用户与开发者: 8 个 Triton 内核套件从 skip/fail 变为实际运行, 显著缩小 XPU 与 CUDA 的测试覆盖差距, MoE、attention 合并、Mamba2 重放、量化等内核在 Intel GPU 上获得隔离回归保护, 属于 RFC #48480 的首个落地批次。对系统运行时: CUDA/ROCm 上 A/B 测试数字完全一致, 生产代码仅有 graph capture 守卫的 API 迁移, 行为等价, 无性能或功能回归。对团队: 确立了可复制的设备无关测试模板 (DEVICE = current_platform.device_type + is_cuda_alike() or is_xpu() 门控 + 条件化算力检查), 后续批次 PR 可直接套用, 降低各平台测试维护分叉成本。- 风险标记: 生产守卫路径 API 迁移, 共享 fixture 全局生效, XPU 仍有部分用例跳过或数值超差, 依赖较新 torch accelerator API

关联脉络

- PR #43407 [XPU][Tests] Enable MoE kernel tests on XPU (rebased from here): PR body 明确说明 commit 3 是 #43407 的 rebase (原作者 Agata Dobrzyniewicz, 已关闭), 本 PR 是其直接后继; 标题为基于内容的推断, 原始标题未在上下文中提供。
- PR #50526 Related to torch.cuda.is_current_stream_capturing on XPU: review 讨论中 jikunshang 提议用该 PR 替代 fused_batched_moe.py 的改动; 作者说明在 XPU kernel 测试中裸调用不可行, 最终改用 torch.accelerator API。标题为基于讨论的推断。