

PR #7808 完整报告

verl-project/verl

[vllm] test: cover batch-invariant switch on Ascend

合并时间: 2026-09-10 23:53

原文链接: <http://prhub.com.cn/verl-project/verl/pull/7808>

执行摘要

- 一句话: 新增 Ascend NPU batch-invariant 开关回归测试 (14 例)
- 推荐动作: 值得精读, 但读的是测试脚手架而不是代码改动本身。
- 重点看 `run_on_npu` 的 spawn 子进程隔离设计: 对于会在全局注册 torch dispatch 表或替换 `torch.sum`、`torch_npu.npu_add_rms_norm` 这类入口的开关, 进程级隔离是唯一可靠的隔离手段, 这套写法可直接迁移到其他 NPU 补丁测试。
- 重点看 `_run_check` 如何用真实 vLLMHTTPServer 加最小 mock (模型元数据 + Ray 上下文 + 引擎配置单例) 触发真实的 `init_batch_invariance()`, 避免了「只测 mock」的假阳性, 同时不付出加载权重与拉起推理引擎的代价。
- 关键设计决策: enabled/disabled 双向断言 (特别是 disabled 时要求 `batch_invariant_ops::事件数必须为 0`), 这比只测开启路径更能捕获 dispatch 注册泄漏。
- 若要真正确认能力, 建议后续补一个跨 batch size 的一致性用例: 同一条样本分别放进 `batch=1` 与 `batch=N` 比较 logprob 位级一致, 才覆盖 `full_determinism` 的语义; 并补齐 CI 收集与算子包安装的验证, 避免长期 skipped。
- 维护建议: 把对 `bi_batch_invariant_LIB` 等私有符号的依赖在注释中显式标注为「随 vllm-ascend 版本可能变动」, 并在 PR 描述中把文件名与 body 里的 `test_batch_invariant_switch.py` 对齐。

功能与动机

PR body 明确说明目标是: 验证 `full_determinism` 的 `enabled/disabled` 两种状态能正确传播到环境变量 `VLLM_BATCH_INVARIANT`, 并选择预期的算子注册与执行路径, 覆盖矩阵乘、reduction、AddRMSNorm 与 fused attention 绑定。batch-invariant 算子是昇腾上保证同一 prompt 在不同 batch 组成下产生一致 logprob 的前提, 一旦开关静默失效, RL 训练中的 ratio/advantage 会随 batch 组成抖动且极难定位。作者刻意不修改任何生产代码 ("No public API or configuration changes. The tests exercise the existing configuration"), 说明这是一次针对已有能力的回归保护。此外 PR body 中的《Before Starting》清单坦言重复工作核查未完成: "gh was unavailable and web search did not establish whether an overlapping PR exists"。

实现拆解

1. 测试入口与跳过策略 (run_on_npu) : 先用 `pytest.importorskip("torch") / pytest.importorskip("torch_npu")` 与 `torch.npu.is_available()` 判定是否为可用 NPU 环境 ; 再用 `importlib.util.find_spec("batch_invariant_ops")` 检查独立 AscendC 算子包, 注释里明确区分「包缺失可 skip / 包存在但加载失败必须报错」。随后每个用例通过 `multiprocessing.get_context("spawn")` 拉起独立子进程执行 `_run_check`, 超时 180 s, `join` 后在 `finally` 中 `kill/join/close`, 目的是让每个算子状态拿到全新的全局 `dispatch` 表与独立退出码, 避免相互污染。
2. 用真实 server 驱动开关传播 (`_run_check`) : 先清空 `VLLM_BATCH_INVARIANT`、`VERL_FULL_DETERMINISM`、`VERL_SEED` 三个环境变量, 再用真实 `RolloutConfig(name="vllm", full_determinism=enabled, tensor_model_parallel_size=1, max_model_len=128)` 加最小 `SimpleNamespace` (`hf_config.max_position_embeddings=128`) 与 `SimpleNamespace(get_job_id=...)` 上下文, `patch` 掉 `ray.get_runtime_context` 与 `ray.util.get_node_ip_address` 后构造真实 `vLLMHttpServer`; 不加载模型、不启动 Ray 集群与推理引擎。构造完成后显式关闭 `_master_sock`、`_dp_rpc_sock`、`_dp_master_sock` 三个 socket, 再断言 `VLLM_BATCH_INVARIANT` 与开关一致、`bi.HAS_ASCENDC_BATCH_INVARIANT` 为真, 最后用 `patch("vllm_ascend.ascend_config.get_ascend_config", return_value=SimpleNamespace(weight_nz_mode=1))` 顶替引擎配置单例并调用真实的 `bi.init_batch_invariance()`。
3. 用 profiler 事件断言算子路由 (`profile_operator`) : 在 `torch.autograd.profiler_legacy.profile(use_cuda=False)` 窗口内执行被测算子并 `torch.npu.synchronize()`, 从 `key_averages()` 中筛出 `batch_invariant_ops::` 前缀事件; `enabled` 时要求目标算子事件数 > 0, `disabled` 时要求该前缀事件数完全为 0 (防止 `dispatch` 注册泄漏到未开启场景)。
4. 七项检查的具体断言: `_check_switch` 校验环境变量与 `bi._batch_invariant_LIB` is not None; `_check_mm / _check_matmul` 分别校验 `torch.mm` → `npu_mm_batch_invariant`、`torch.matmul` → `npu_matmul_batch_invariant`, 并用 CPU fp32 计算再转 bf16 做 `rtol=0, atol=0` 位级比对; `_check_torch_sum` 用 `torch.sum` is `bi.reduce_sum`、`_check_tensor_sum` 用 `torch.Tensor.sum` is `bi.reduce_sum` 做身份断言, 并校验 `npu_reduce_sum_batch_invariant` 事件与数值; `_check_add_rms_norm` 校验 `torch_npu.npu_add_rms_norm` is `bi.add_rms_norm`, `enabled` 时要求出现 `aten::add + npu::npu_rms_norm` 且 `npu::npu_add_rms_norm` 为 0、`rstd` is None, `disabled` 时要求出现 `npu::npu_add_rms_norm`; `_check_fused_infer_attention_score_binding` 只校验 `torch_npu.npu_fused_infer_attention_score` is `torch.ops.batch_invariant_ops.npu_fused_infer_attention_score_batch_invariant` 的绑定关系。
5. 测试与配套: 7 个测试函数统一用 `@pytest.mark.parametrize("enabled", [False, True], ids=["disabled", "enabled"])`, 合计 14 例; 运行命令为 `python -m pytest -v -s tests/workers/rollout/rollout_vllm/test_batch_invariant_ops_on_npu.py`。本 PR 无源码、配置、文档或 submodule 改动; 作者自述需在装有 PyTorch、torch_npu、vLLM Ascend 与独立 AscendC batch-invariant 算子包的昇腾环境运行, 并声明「NPU tests and pre-commit checks have not been run during PR preparation」。

关键文件：

- tests/workers/rollout/rollout_vllm/test_batch_invariant_ops_on_npu.py（模块 昇腾回归测试；类别 test；类型 test-coverage；符号 _run_check, run_on_npu, profile_operator, _check_switch）：本 PR 唯一新增文件，用 14 个用例（7 项检查 × enabled/disabled）为昇腾 NPU 上的 vLLM full_determinism / VLLM_BATCH_INVARIANT 开关建立回归网；包含 spawn 子进程隔离、真实 vLLMHttpServer 构造、profiler 事件断言与 CPU 参考值校验四个可复用的测试模式。

关键符号：_run_check, run_on_npu, profile_operator, _check_switch, _check_mm, _check_matmul, _check_torch_sum, _check_tensor_sum, _check_add_rms_norm, _check_fused_infer_attention_score_binding

关键源码片段

tests/workers/rollout/rollout_vllm/test_batch_invariant_ops_on_npu.py

本 PR 唯一新增文件，用 14 个用例（7 项检查 × enabled/disabled）为昇腾 NPU 上的 vLLM full_determinism / VLLM_BATCH_INVARIANT 开关建立回归网；包含 spawn 子进程隔离、真实 vLLMHttpServer 构造、profiler 事件断言与 CPU 参考值校验四个可复用的测试模式。

```
# tests/workers/rollout/rollout_vllm/test_batch_invariant_ops_on_npu.py（整理后的核心支架）
```

```
def _run_check(check, enabled):
    # 每个子进程都从干净环境开始：先清掉 shell 或上次运行遗留的开关变量，
    # 保证 VLLM_BATCH_INVARIANT 只可能由 RolloutConfig.full_determinism 写入。
    for name in ("VLLM_BATCH_INVARIANT", "VERL_FULL_DETERMINISM", "VERL_SEED"):
        os.environ.pop(name, None)

    import torch
    import torch_npu # noqa: F401 - 仅用于注册 NPU backend
    import vllm_ascend.batch_invariant as bi

    from verl.workers.config import RolloutConfig
    from verl.workers.rollout.replica import RolloutMode
    from verl.workers.rollout.vllm_rollout.vllm_async_server import vLLMHttpServer

    # 复用真实 RolloutConfig: full_determinism 就是被测开关，其余字段压到最小可运行规模。
    config = RolloutConfig(name="vllm", full_determinism=enabled, tensor_model_parallel_size=1,
                           max_model_len=128)
    # 只补足模型元数据与 Ray 身份；不加载权重、不拉起 Ray 集群、不启动推理引擎。
    model = SimpleNamespace(hf_config=SimpleNamespace(max_position_embeddings=128))
    context = SimpleNamespace(get_job_id=lambda: "batch-invariant-ut")
    with (
        patch("ray.get_runtime_context", return_value=context),
        patch("ray.util.get_node_ip_address", return_value="127.0.0.1"),
    ):
        pass
```

```

server = vLLMHttpServer(
    config, model, RolloutMode.STANDALONE, [], 0, 0, 1, 1, os.environ.get("ASCEND_RT_VISIBLE_DEVICES", "0")
)
# vLLMHttpServer 构造期会开 ZMQ/RPC socket, 必须显式关闭, 避免子进程退出前占端口。
for sock in (server._master_sock, server._dp_rpc_sock, server._dp_master_sock):
    sock.close()
# 断言点 1: 开关是否真的落到 vLLM Ascend 读取的环境变量入口上。
assert os.environ.get("VLLM_BATCH_INVARIANT", "0") == ("1" if enabled else "0")
# 包「装了但加载失败」必须报错, 只有「完全缺失」才允许在 run_on_npu 里 skip。
assert bi.HAS_ASCENDC_BATCH_INVARIANT, "The installed independent AscendC operator package could not be loaded."
# 用假配置单例顶替引擎配置, 但执行的是 vllm_ascend 真实的算子注册代码。
with patch("vllm_ascend.ascend_config.get_ascend_config", return_value=SimpleNamespace(weight_nz_mode=1)):
    bi.init_batch_invariance()
torch.npu.set_device(0)
check(enabled)

def run_on_npu(check, enabled):
    torch = pytest.importorskip("torch")
    pytest.importorskip("torch_npu")
    if not torch.npu.is_available():
        pytest.skip("Requires an Ascend NPU.")
    # 只把「包不存在」当作可跳过条件; 已安装但加载失败会在 _run_check 里直接断言失败。
    if importlib.util.find_spec("batch_invariant_ops") is None:
        pytest.skip("Requires the independent batch-invariant operator package (batch_invariant_ops).")

    # 核心设计: 每个算子 / 开关组合跑在独立 spawn 子进程中,
    # 从而拿到全新的 torch dispatch 表、monkey patch 状态和独立的退出码。
    process = multiprocessing.get_context("spawn").Process(target=_run_check, args=(check, enabled))
    process.start()
    try:
        process.join(timeout=180) # 超时即判失败, 避免昇腾卡死拖垮整个 CI
        assert process.exitcode == 0, (
            f"{check.__name__} failed or timed out ({enabled=}, exitcode={process.exitcode}); see subprocess output."
        )
    finally:
        # 无论成功、失败还是超时都要回收子进程, 防止残留进程继续占用 NPU。
        if process.is_alive():
            process.kill()
            process.join()
        process.close()

def profile_operator(entry, call, enabled, operator=None):

```

```

import torch
from torch.autograd.profiler_legacy import profile

# enabled 时 dispatch 会把算子路由到 batch_invariant_ops::*, 因此用 profiler
# 事件反证路由生效。
with profile(use_cuda=False) as captured:
    result = call()
    torch.npu.synchronize()
events = {event.key: int(event.count) for event in captured.key_averages()}
counts = {name: count for name, count in events.items() if name.startswith("batch_invariant_
ops::")}
if enabled and operator is not None:
    assert counts.get(f"batch_invariant_ops::{operator}", 0) > 0, (entry, counts)
else:
    # 关闭开关时必须完全没有 batch_invariant_ops 事件, 防止注册泄漏到后续用例。
    assert not counts, (entry, counts)
print(f"{enabled=}, {entry=}, {counts=}", flush=True)
return result, events

# AddRMSNorm 检查: 断言的是「执行路径选择」, 而不是同一个算子

def _check_add_rms_norm(enabled):
    import torch
    import torch_npu
    import vllm_ascend.batch_invariant as bi

    # 先做身份断言: 开关关闭时 torch_npu.npu_add_rms_norm 不应被替换成 batch-invariant
    # 实现。
    assert (torch_npu.npu_add_rms_norm is bi.add_rms_norm) is enabled
    x_cpu = (torch.arange(512).reshape(16, 32) % 7 - 3).to(torch.bfloat16)
    residual_cpu = torch.full_like(x_cpu, 0.5)
    weight_cpu = (torch.arange(32).float() / 32 + 0.5).to(torch.bfloat16)
    x, residual, weight = x_cpu.npu(), residual_cpu.npu(), weight_cpu.npu()
    eps = 1e-6
    # CPU 参考实现: 先 add 再做 RMSNorm, 与开启 batch invariant 后的算子组合保持一致。
    added = x_cpu.float() + residual_cpu.float()
    normalized = added * torch.rsqrt(added.square().mean(dim=-1, keepdim=True) + eps) *
    weight_cpu.float()
    (actual, rstd, residual_out), events = profile_operator(
        "AddRMSNorm", lambda: torch_npu.npu_add_rms_norm(x, residual, weight, eps), enabled
    )
    # 当前 vLLM Ascend 版本开启 batch invariant 后走 add + RMSNorm
    # 组合, 而不是独立的融合算子,
    # 所以这里断言的是路径选择: 开启时不允许出现 npu::npu_add_rms_norm。
    if enabled:
        assert events.get("aten::add", 0) > 0, events
        assert events.get("npu::npu_rms_norm", 0) > 0, events
        assert events.get("npu::npu_add_rms_norm", 0) == 0, events
        assert rstd is None # 组合路径不返回 rstd, 是区分两条路径的另一个判据

```

```
else:
    assert events.get("npu::npu_add_rms_norm", 0) > 0, events
assert actual.device == residual_out.device == x.device
# residual_out 要求位级一致, normalized 允许 1e-2 误差, 反映两条路径的数值差异预期。
torch.testing.assert_close(residual_out.cpu(), added.to(torch.bfloat16), rtol=0, atol=0)
torch.testing.assert_close(actual.cpu(), normalized.to(torch.bfloat16), rtol=1e-2, atol=1e-2)
```

评论区精华

本次 PR 没有产生实质性的技术讨论 (`review_comments_count=0`)，唯一的 review 结论是 [wucong25](#) 的空 body APPROVED，因此最有价值的信息来自 PR body 自述与 review 结果之间的落差：

- 作者在《Test》小节写明 "Validation status: NPU tests and pre-commit checks have not been run during PR preparation. Hardware/software versions and actual results remain to be recorded."，即测试未在真实昇腾环境跑通、版本与结果未记录，但 reviewer 未在评审批次中提出异议即放行。
- 《Before Submitting》清单中「Verify CI coverage」也被标注为待办: "The test file matches the existing NPU unit-test collection rules; compatibility of the CI image and availability of the independent AscendC operator package remain to be verified."，说明 CI 是否真正收集并执行该文件仍是开放问题。
- CLAassistant 机器人评论提示提交者身份未关联: "wangdongle seems not to be a GitHub user. You need a GitHub account to be able to sign the CLA..."，属流程问题，与代码无关但会影响署名与合规状态。
- 作者对 AI 参与的披露较为克制: "Codex assisted with source inspection and preparation of the commit message and PR description. Human review of every changed line and execution of relevant tests remain pending."
- 测试未在昇腾实机验证即被批准合入 (testing): reviewer [wucong25](#) 以空 body 的 APPROVED 直接放行 (`review_comments_count=0`)，没有任何评论追问实机结果或 CI 收集情况；测试能否真正在昇腾 CI 上执行、是否长期 skipped，仍为未闭合的开放问题。
- 重复工作核查未完成 (question): 未确认是否已有其他 batch-invariant / Ascend full_determinism 相关 PR 覆盖同类验证，存在重复投入的可能；合入后仍需人工核对，且 PR body 中写的目标文件名与实际文件不一致，进一步增加了检索难度。
- CLA 签名与提交者身份未关联 (other): 属流程与合规问题，不影响代码内容；需作者把提交邮箱加入 GitHub 账号后重新触发校验才能清除该告警。

风险与影响

- 风险：
 1. 静默 skip 造成覆盖假象（最高风险）：run_on_npu 在缺少 batch_invariant_ops 包或无 NPU 时直接 pytest.skip。若昇腾 CI 镜像未安装独立 AscendC 算子包，14 个用例会全部 skipped 但仍显示绿色，full_determinism 实际上没有任何回归保护——这正是 PR body 未闭环的验证项。

2. 对 vllm-ascend 私有符号的强耦合: `bi._batch_invariant_LIB`、`bi.reduce_sum`、`bi.add_rms_norm`、`bi.HAS_ASCENDC_BATCH_INVARIANT` 以及 `vllm_ascend.ascend_config.get_ascend_config` 都不是稳定公共 API。上游重构或改名会导致测试假红；更危险的是若上游把开关语义改掉而符号仍在，测试可能通过却已失去意义。
3. 位级精度断言脆弱: `_check_mm` / `_check_matmul` / `_check_torch_sum` / `_check_tensor_sum` 使用 `rtol=0`, `atol=0` 与 CPU fp32 参考值做 bf16 精确比对，只有算子确实以确定性累加顺序实现时才成立，换算子版本、换硬件形态都可能 flaky。相比之下 `AddRMSNorm` 的 `normalized` 断言已放宽到 `rtol=1e-2`, `atol=1e-2`，说明作者对精度差异有意识，但矩阵与 reduction 路径仍偏紧。
4. profiler 依赖可能干扰被测对象: `torch.autograd.profiler_legacy` 已被 PyTorch 标记为 legacy，且断言完全依赖 `batch_invariant_ops::`、`npu::npu_rms_norm`、`aten::add` 等事件名字符串，事件名或 profiler 行为变化会直接破坏判定；profiler 本身也可能影响融合算子的选择。
5. Ray patch 脆弱: `patch("ray.get_runtime_context")`、`patch("ray.util.get_node_ip_address")` 依赖 Ray 的调用路径与时机，Ray 版本升级会让 `vLLMHttpServer` 构造阶段抛出与业务无关的错误。
6. CI 时长与端口占用: 14 个用例各 spawn 独立子进程并构造 `vLLMHttpServer`（含 socket 绑定与 AscendC 库加载），单进程 180 s 上限，若接入夜间 CI 需评估时长预算；socket 虽已显式 close，但异常路径下的清理依赖 `run_on_npu` 的 finally。
7. 描述与产物不一致: PR body 写的是 `tests/workers/rollout/rollout_vllm/test_batch_invariant_switch.py`，实际合入 `test_batch_invariant_ops_on_npu.py`（`test_batch_invariant_switch` 只是其中一个用例名），会降低后续按名检索与重复工作排查的效率。

• 影响:

- 用户影响: 零。不改动任何生产代码、公共 API、配置键或示例脚本，现有训练流程行为完全不变。
- 系统影响: 为昇腾平台 `full_determinism=True` 路径补上第一道回归网。若 CI 真正接入并有算子包，可在 `torch_npu` / `vllm-ascend` 升级时提前发现「开关未落到 `VLLM_BATCH_INVARIANT`」或「算子未注册」这类静默退化。局限是只验证单算子的绑定与数值，不验证不同 batch size 下的数值一致性，也不覆盖端到端生成确定性——`full_determinism` 的真正语义尚未被端到端证明。
- 团队影响: 确立了一种可复用的昇腾测试范式——真实 `vLLMHttpServer` 触发配置传播、spawn 子进程隔离全局 dispatch 与 monkey patch、profiler 事件断言路由、CPU 参考值断言数值。该模式可被 IPC 开关、量化 patch、其他 NPU 适配补丁的测试沿用。
- 流程影响: PR 在 NPU 实机验证与 pre-commit 均未完成的情况下合入，且 CLA 未关联提交者身份，后续需补齐验证证据，否则测试的可信度仅停留在「写得对」而非「跑得通」。
- 风险标记: 仅测试改动，未在 NPU 实机验证，环境缺失时静默 skip，耦合 `vllm-ascend` 私有符号，位级精度断言可能 flaky，profiler/Ray patch 脆弱，文件名与 PR 描述不一致。

关联脉络

- PR #7779 [vllm, hardware] fix: patch vllm-ascend for GLM-5.2 RL training on Ascend NPU: 同属 vllm-ascend 在昇腾上的适配补丁线 (verl/Utils/vllm/npu_vllm_patch.py) , 本 PR 校验的 vllm_ascend.batch_invariant 正是同一适配链上的开关, 两者共享对 vllm-ascend 内部实现的耦合风险。
- PR #7728 [rollout] fix: default IPC weight transfer on Ascend A5 hardware: 与本 PR 高度同构: 都是昇腾平台「行为开关 + NPU 单测」模式 (verl/plugin/platform/platform_npu.py + tests/Utils/test_check_ipc_version_support_on_npu.py) , 可复用其 CI 收集规则与 skip 策略经验。
- PR #7827 [megatron, ci] fix: configure Megatron integration in Ascend images and tests: 决定了昇腾镜像与 NPU 单测默认加载哪套依赖; 本测试依赖镜像内 torch_npu、vllm-ascend 与独立 AscendC 算子包, 镜像 tag 或依赖变更直接决定该测试能否真正执行还是被 skip。
- PR #7799 [ci] fix: nightlyCI_grpo_qwen3_5_2b_fsdp2_vllm_ascend: 昇腾夜间 CI workflow 的维护线; 本测试若要在 CI 中生效, 需要挂载到类似的 NPU 工作流并确认算子包安装步骤, 两者在 CI 覆盖面上一脉相承。
- PR #7797 [hardware] test: include AMD in platform auto-detection expectations: 同为平台相关的测试补强 (tests/plugin/test_platform_abstraction.py) , 可对比其在断言粒度与 CI 集成上的取舍, 判断本 PR 的 skip 策略是否合理。