PR #32505 完整报告

sgl-project/sglang

[UT][NPU] add unit tests for ascend_torch_native_backend and mla_preprocess

合并时间：2026-08-08 16:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32505

执行摘要

- 一句话：为 NPU 注意力后端与 MLA 预处理新增单元测试
- 推荐动作：值得快速阅读，不需要精读。重点看两点：_ref_extend 参考实现如何用 torch 原生 SDPA 模拟 paged KV 的 extend 语义，这是验证注意力后端正确性的通用范式；test_npu_mla_preprocess.py 中环境变量 + lru_cache 的测试隔离手法（patch.dict + cache_clear）。若后续 NPU 后端有 kernel 级重构，此 PR 提供的测试可直接作为回归基线。

功能与动机

PR body 明确指出：mla_preprocess 和 ascend_torch_native_backend 模块当前缺乏专门单元测试，本 PR 的目标是“pin down their behavior contracts and catch regressions during future refactors”。NPU（Ascend）注意力后端是硬件差异化代码路径，此前缺少行为契约的回归保障，一旦重构容易静默破坏。

实现拆解

1. 测试注册入口：两个新测试文件均在顶部调用 register_npu_ci(est_time=4, suite="stage-a-unit-test-npu")，将测试注册进 NPU CI 流水线的 stage-a 单测套件，预估每个文件耗时 4 分钟。
2. test_npu_ascend_torch_native_backend.py（+831 行）：围绕 AscendTorchNativeAttnBackend 分四组测试。TestInit/TestSupportTriton 验证构造成功且 support_triton() 返回 False；TestScaledDotProductAttentionWithSoftcapping 覆盖 SDPA 的各种模式——基础注意力、causal mask、attn_mask 与 is_causal 互斥断言、显式 scale、GQA 广播、tanh softcapping、logit_cap=0 关闭 softcapping、boolean 与 additive mask；TestRunSdpaForwardExtend 提供手写参考实现 _ref_extend，逐序列按 req_to_token 查表拼装 KV、构造 padded query，覆盖多序列、带 prefix 和 causal 三种 extend 场景；另有 decode 前向路径测试（PR body 提及）。
3. test_npu_mla_preprocess.py（+207 行）：覆盖 mla_preprocess 工具函数。TestRoundUp 固定 round_up 边界行为，包括精确倍数、向上取整、0 值、align=0 返回 0，以及负 align 的行为 pin（round_up(10, -4) == 8）；TestTransdata 逐元素验证 NZ 布局映射公式 nz[c // bs1, r, c % bs1]，覆盖 padding、非方阵、自定义 block 和 3D 输入报错；TestTransRopeWeight 验证 RoPE 权重区域偶先奇后重排、非 RoPE 区域不变、3D 权重（多 expert）逐 expert 重排；TestIsMlaPreprocessEnabled/TestIsFiaNz 用

patch.dict(os.environ) 配合 cache_clear() 验证环境变量开关及其缓存行为，并固定“FIA NZ 依赖 MLA preprocess，未开启时断言失败”的契约。

4. 配套改动：无源码、配置、文档或部署改动，纯新增测试文件。

关键文件：

- test/registered/unit/npu/attention/test_npu_ascend_torch_native_backend.py（模块 注意力后端；类别 test；类型 test-coverage；符号 TestInit, TestSupportTriton, TestScaledDotProductAttentionWithSoftcapping, TestRunSdpaForwardExtend）：覆盖 NPU 注意力后端核心行为：构造、Triton 支持标记、SDPA+softcapping 全模式，以及用 torch 原生 SDPA 手写参考实现验证 extend/decode 前向路径，是本次测试的主体（+831 行）。
- test/registered/unit/npu/attention/test_npu_mla_preprocess.py（模块 MLA 预处理；类别 test；类型 test-coverage；符号 TestRoundUp, TestTransdata, TestTransRopeWeight, TestIsMlaPreprocessEnabled）：覆盖 MLA 预处理工具链：round_up 边界、NZ 布局逐元素映射、RoPE 权重重排，以及两个环境变量开关的缓存与依赖契约，是 NPU MLA 路径重构的回归防线（+207 行）。

关键符号：AscendTorchNativeAttnBackend.scaled_dot_product_attention_with_softcapping, AscendTorchNativeAttnBackend.run_sdpa_forward_extend, AscendTorchNativeAttnBackend.run_sdpa_forward_decode, AscendTorchNativeAttnBackend.support_triton, round_up, transdata, trans_rope_weight, is_mla_preprocess_enabled, is_fia_nz

关键源码片段

test/registered/unit/npu/attention/test_npu_ascend_torch_native_backend.py

覆盖 NPU 注意力后端核心行为：构造、Triton 支持标记、SDPA+softcapping 全模式，以及用 torch 原生 SDPA 手写参考实现验证 extend/decode 前向路径，是本次测试的主体（+831 行）。

```
# 关键参考实现：用 torch 原生 SDPA 模拟 paged KV + 多序列 batching 语义。
# query 是 [total_query, H, D] 的扁平布局，需要按序列切分并 0-padding 到
# [1, H, seq_len_kv, D]，再与 k_cache/v_cache 按 req_to_token 查表得到的
# 整段 KV 做注意力，最后切回 [ext_len, H, D] 与后端输出布局对齐。
def _ref_extend(self, query, k_cache, v_cache, req_to_token, req_pool_indices,
                seq_lens, extend_prefix_lens, extend_seq_lens,
                scaling=None, enable_gqa=False, causal=False):
    H, D = query.shape[1], query.shape[2]
    outputs = []
    start_q = 0
    for seq_idx in range(seq_lens.shape[0]):
        ext_len = int(extend_seq_lens[seq_idx].item())
        pre_len = int(extend_prefix_lens[seq_idx].item())
        seq_len_kv = int(seq_lens[seq_idx].item())
        end_q = start_q + ext_len
```

```
        # 通过 req_to_token 拿到该序列的全部 KV token 位置。
        req_pool_idx = req_pool_indices[seq_idx]
        tokens = req_to_token[req_pool_idx, :seq_len_kv]

        # 构造 padded query，extend 部分放在 pre_len 之后，前缀部分置 0。
        q_pad = torch.zeros(1, H, seq_len_kv, D, dtype=query.dtype)
        q_pad[0, :, pre_len:pre_len + ext_len, :] = query[start_q:end_q].movedim(0, 1)
        k_u = k_cache[tokens].movedim(0, 1).unsqueeze(0)
        v_u = v_cache[tokens].movedim(0, 1).unsqueeze(0)

        # GQA 场景下把 KV head 重复扩展到与 query head 数一致。
        if enable_gqa and k_u.size(-3) != q_pad.size(-3):
            rep = q_pad.size(-3) // k_u.size(-3)
            k_u = k_u.repeat_interleave(rep, -3)
            v_u = v_u.repeat_interleave(rep, -3)

        out = scaled_dot_product_attention(q_pad, k_u, v_u, scale=scaling, is_causal=causal)
        # 切出 extend 段的输出并恢复为 [ext_len, H, D] 布局。
        out = out[0, :, pre_len:pre_len + ext_len, :].movedim(1, 0)
        outputs.append(out)
        start_q = end_q
    return torch.cat(outputs, dim=0)
```

test/registered/unit/npu/attention/test_npu_mla_preprocess.py

覆盖 MLA 预处理工具链：round_up 边界、NZ 布局逐元素映射、RoPE 权重重排，以及两个环境变量开关的缓存与依赖契约，是 NPU MLA 路径重构的回归防线（+207 行）。

```
# 环境变量开关类测试：被测函数带 lru_cache，测试之间必须成对 cache_clear()，
# 否则前一个用例的缓存会污染后一个用例的结论。
class TestIsMlaPreprocessEnabled(unittest.TestCase):
    def setUp(self):
        is_mla_preprocess_enabled.cache_clear()

    def tearDown(self):
        is_mla_preprocess_enabled.cache_clear()

    def test_not_set_returns_false(self):
        with patch.dict(os.environ):
            os.environ.pop("SGLANG_NPU_USE_MLAPO", None)
            self.assertFalse(is_mla_preprocess_enabled())

    def test_set_to_one_returns_true(self):
        with patch.dict(os.environ, {"SGLANG_NPU_USE_MLAPO": "1"}):
            self.assertTrue(is_mla_preprocess_enabled())

    def test_set_to_zero_returns_false(self):
        with patch.dict(os.environ, {"SGLANG_NPU_USE_MLAPO": "0"}):
            self.assertFalse(is_mla_preprocess_enabled())
```

```
class TestIsFiaNz(unittest.TestCase):
    def setUp(self):
        is_mla_preprocess_enabled.cache_clear()
        is_fia_nz.cache_clear()

    def tearDown(self):
        is_mla_preprocess_enabled.cache_clear()
        is_fia_nz.cache_clear()

    # 关键契约：FIA NZ 布局依赖 MLA preprocess 开关，只开前者会触发断言。
    def test_fia_nz_without_mlapo_raises(self):
        with patch.dict(os.environ):
            os.environ.pop("SGLANG_NPU_USE_MLAPO", None)
            os.environ["SGLANG_USE_FIA_NZ"] = "1"
            with self.assertRaises(AssertionError):
                is_fia_nz()
```

评论区精华

该 PR 无任何人工 review 评论或讨论线程。Issue 区仅两条 bot 留言：gemini-code-assist[bot] 声明其代码审查已停止；sglang-npu-bot 通过 /tag-and-rerun-ci 触发 CI 重跑。合入流程完全由 CI gating 与 bot 驱动，未出现设计权衡或争议。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. CI 稳定性风险：新增两个 NPU 单测注册进 stage-a-unit-test-npu，若 NPU runner 资源紧张或环境不稳定，可能引入 CI 抖动；est_time=4 的预估需验证实际耗时。
  2. 测试与实现强耦合：round_up(10, -4) == 8 等负 align 行为被显式 pin 住，若未来实现修正该语义，测试会先红灯——这既是契约固定，也可能阻碍合理的行为修正，需要维护者知晓。
  3. 环境变量缓存隔离：is_mla_preprocess_enabled 与 is_fia_nz 带 lru_cache，测试依赖 setUp/tearDown 中成对 cache_clear()；若生产代码后续增加新参数导致缓存键变化，测试可能漏覆盖，但当前隔离是完整的。
  4. 无推理路径风险：纯新增测试文件，对线上推理、性能、二进制产物零影响。 - 影响：对用户与推理服务无直接影响（纯测试变更）。对团队的影响主要在 CI 侧：NPU CI 流水线新增 stage-a 单测负载约 8 分钟（两个文件各 4 分钟预估）。对 NPU/Ascend 后端开发者而言，ascend_torch_native_backend 与 mla_preprocess 从此有了行为契约的可执行记录，后续重构（如 kernel 替换、布局调整）可获得即时回归反馈。测试中手写参考实现与 NZ 布局逐元素验证的模式，也可作为其他硬件后端单测的范本。 - 风险标记：CI 依赖 NPU 环境，测试与实现强耦合，环境变量缓存隔离

关联脉络

- PR #33981 [AMD] Add K3 verified mla kernel for DSpark on triton backend: 同为不同硬件后端（AMD vs NPU）上的 MLA 注意力路径验证工作，与本 PR 的 NPU MLA 预处理测试形成平行脉络，可对比不同硬件后端对 MLA 计算图的验证策略；非直接代码依赖。