PR #33864 完整报告

sgl-project/sglang

[diffusion] fix: MiniMax-H3 text encoder device mismatch under --text-encoder-cpu-offload

合并时间：2026-08-07 14:04

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33864

执行摘要

- 一句话：修复 MiniMax-H3 文本编码器在 CPU offload 下的设备错配
- 推荐动作：值得精读。该 PR 用很小的改动解决了一个隐蔽的分布式推理陷阱——把“参数存储设备”当作“计算设备”。device 属性语义的定义、docstring 对 FSDP offload 的解释，以及测试中通过 __new__ 构造轻量实例的技巧，都值得后续实现 offload 组件时参考。

功能与动机

PR body 明确说明 --text-encoder-cpu-offload 在 MiniMax-H3 上不可用：每个请求在文本编码阶段都会失败，报错 RuntimeError: Expected all tensors to be on the same device, but got mat2 is on cpu, different from other tensors on cuda:0，在 1x RTX PRO 5000 上复现 3/3。根因是 MiniMaxH3Qwen3VLEncoder.device 返回 next(self.parameters()).device，在 FSDP CPU offload 策略下参数停留在 CPU，而 forward 在加速器上执行，encode_ids 因此把输入张量放在 CPU，最终在 Qwen3VLRotaryEmbedding.forward 的 bmm 处崩溃。该问题同时阻塞了模型级 offload 与 DiT 层级 offload 混用两种组合。

实现拆解

1. 定位根因：MiniMaxH3Qwen3VLEncoder.device 属性原先返回 next(self.parameters()).device，在 --text-encoder-cpu-offload 下参数存储于 CPU 而 forward 在 GPU 上执行，导致 encode_ids 用 .to(self.device) 把 input_ids、attention_mask、position_ids 都放到 CPU，向 Rope 的 bmm 传入跨设备张量而使每个请求崩溃。
2. 修改实现：在 python/sglang/multimodal_gen/runtime/models/encoders/minimax_h3_qwen3vl.py 中引入 get_local_torch_device()（来自 sglang.multimodal_gen.runtime.distributed），将 device 属性改为返回该函数结果，并补充 docstring 说明为何不能使用参数设备。该属性被用于 4 处输入构建，因此一处修改即可修复所有输入张量的放置。
3. 新增单元测试：新增 test_minimax_h3_encoder_device.py，用 __new__ 构造未初始化的 encoder 实例并注册一个 CPU 参数，mock get_local_torch_device 为指定设备，断言 device 属性返回计算设备而非参数设备；同时覆盖纯 CPU 平台两者一致的场景。
4. 验证与配套：PR body 提供精度与速度数据——offload 路径从崩溃变为通过（峰值 43.4 GB），--dit-cpu-offload 组合从崩溃变为预期 OOM；非 offload 路径峰值内存字节级一致，e2e 差异为噪声。CI 由维护者触发 rerun 并通过。

关键文件：

- python/sglang/multimodal_gen/runtime/models/encoders/minimax_h3_qwen3vl.py（模块 文本编码器；类别 source；类型 data-contract）：bug 根因所在：device 属性原先返回参数存储设备，offload 下导致输入张量放错设备；修改后返回 get_local_torch_device()，并新增 docstring 说明语义。
- python/sglang/multimodal_gen/test/unit/test_minimax_h3_encoder_device.py（模块 文本编码器；类别 test；类型 test-coverage；符号 TestMiniMaxH3EncoderDevice, _encoder_with_param_on, test_device_ignores_cpu_offloaded_parameters, test_device_follows_local_device_on_cpu_only_platforms）：新增单元测试，锁定 device 属性必须反映计算设备的不变式，覆盖 CPU offload 与纯 CPU 两种场景。

关键符号：MiniMaxH3Qwen3VLEncoder.device, TestMiniMaxH3EncoderDevice._encoder_with_param_on, TestMiniMaxH3EncoderDevice.test_device_ignores_cpu_offloaded_parameters, TestMiniMaxH3EncoderDevice.test_device_follows_local_device_on_cpu_only_platforms

关键源码片段

python/sglang/multimodal_gen/test/unit/test_minimax_h3_encoder_device.py

新增单元测试，锁定 device 属性必须反映计算设备的不变式，覆盖 CPU offload 与纯 CPU 两种场景。

```
import unittest
from unittest import mock

import torch

from sglang.multimodal_gen.runtime.models.encoders import minimax_h3_qwen3vl
from sglang.multimodal_gen.runtime.models.encoders.minimax_h3_qwen3vl import (
    MiniMaxH3Qwen3VLEncoder,
)


class TestMiniMaxH3EncoderDevice(unittest.TestCase):
    """`device` 必须命名计算侧，而不是参数存储侧。

    `--text-encoder-cpu-offload` 以 FSDP CPU offload 策略加载本编码器：
    分片参数在 CPU 上，前向时才 all-gather 到加速器。若按参数设备报告，
    `encode_ids` 会把 `input_ids`、`attention_mask`、`position_ids` 建在
    CPU 上，而前向在加速器上执行，Rope 的 matmul 就会以
    "Expected all tensors to be on the same device ... mat2 is on cpu" 失败。
    """

    def _encoder_with_param_on(self, device: torch.device) -> MiniMaxH3Qwen3VLEncoder:
        # 用 __new__ 绕过重量级 __init__，只注册一个 CPU 参数即可模拟 offload 状态。
        encoder = MiniMaxH3Qwen3VLEncoder.__new__(MiniMaxH3Qwen3VLEncoder)
```

```
        torch.nn.Module.__init__(encoder)
        encoder.register_parameter(
            "offloaded", torch.nn.Parameter(torch.zeros(1, device=device))
        )
        return encoder

    def test_device_ignores_cpu_offloaded_parameters(self):
        # 参数真实在 CPU，但本地计算设备是 CUDA:3，device 应报告后者。
        encoder = self._encoder_with_param_on(torch.device("cpu"))
        compute_device = torch.device("cuda", 3)

        with mock.patch.object(
            minimax_h3_qwen3vl, "get_local_torch_device", return_value=compute_device
        ):
            self.assertEqual(encoder.device, compute_device)

        # 确认参数确实在 CPU：property 不是简单地把参数设备回显出来。
        self.assertEqual(next(encoder.parameters()).device.type, "cpu")

    def test_device_follows_local_device_on_cpu_only_platforms(self):
        # 纯 CPU 平台上，本地计算设备就是 CPU，行为应与原来一致。
        encoder = self._encoder_with_param_on(torch.device("cpu"))
        cpu = torch.device("cpu")

        with mock.patch.object(
            minimax_h3_qwen3vl, "get_local_torch_device", return_value=cpu
        ):
            self.assertEqual(encoder.device, cpu)


if __name__ == "__main__":
    unittest.main()
```

评论区精华

该 PR 没有实质 review 讨论：BBuf 直接批准（APPROVED，无评论 body），维护者 mickqian 仅执行 /tag-and-rerun-ci 触发 CI 重跑，未产生设计争议或未解决疑虑。

- 暂无高价值评论线程

风险与影响

- 风险：风险点集中在 *device* 属性的语义变更：修复前该属性被用于四处输入放置，修复后统一以 *get_local_torch_device()* 为准；若未来该函数在特定部署（如多流、多设备上下文）中不表示实际计算设备，可能引入新的错配。当前正常 GPU 驻留路径下两者等价，回归风险较低。测试全部基于 mock，未覆盖真实 FSDP 多卡集成环境，真实环境行为仍需 CI 验证。该修复只触及 MiniMax-H3 编码器，不影响其他模型。

- 影响：用户侧：解锁了 --text-encoder-cpu-offload 在 MiniMax-H3 上的使用，使其可在 72 GB 单卡（RTX PRO 5000）上运行 1344x768、124 帧的 t2va 任务；同时 device 属性的语义被修正为“计算侧设备”，对后续其他 offload 组件有示范作用，也消除了“参数在 CPU 上就认为编码器在 CPU”这类常见误解。团队侧：新增的单元测试将设备语义契约固化为可回归的测试资产，便于后续维护。
- 风险标记：设备语义变更，依赖分布式工具函数，缺少真实 FSDP 集成测试

关联脉络

- PR #33875 [diffusion] Fix 4/8-step distilled MiniMax-H3 Turbo LoRA merge: 同属 MiniMax-H3 功能线，修复该模型的另一个缺陷（LoRA 2D 权重合并崩溃），与本次编码器 device 修复无直接依赖，但共同完善 H3 支持。
- PR #33707 Derive H3 attention admission from backend capabilities: 同为 H3 相关改造，调整注意力后端准入与 backends 契约，与本 PR 都在 MiniMax-H3 组件上变更 device/backend 语义。