PR #34993 完整报告

sgl-project/sglang

[diffusion] fix: make MiniMax-H3 AdaLN cache rebuild transactional

合并时间：2026-08-19 10:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34993

执行摘要

- 一句话：H3 AdaLN 缓存重建改事务性发布，修复重置与失败两条路径
- 推荐动作：值得精读。核心价值不在代码量而在设计决策：缓存元数据的发布时机决定了失败后的可重试性，事务性发布把"部分完成"状态对后续请求隐藏。建议重点关注两点：一是容量重置时按 wanted 而非 missing 重建的语义选择；二是 plan_lengths 与 _slots 的提交顺序这一不变式。对想为 diffusion 缓存补状态类测试的开发者，测试文件中的 _online_cache、_write_online_weights 辅助函数是可直接复用的模式。

功能与动机

PR body 明确列出两条会让缓存不一致的记账路径：

1) 当新请求超过 max_plans 时，缓存清空所有驻留计划但只重建清空前的缺失计划；若请求同时复用了已有计划，该计划会从 slab 消失，并在 denoising 时查找失败。 2) 计划在 checkpoint 读取与投影完成前就被写入 host 端 _slots 索引；若重建中途抛错，重试会把不完整计划当作缓存命中而跳过重建，随后因 plan_lengths 仍为零导致查找失败。此外缓存接受非正计划容量，会构造出退化 slab，直到更晚才以不易操作的错误暴露。triple-mu 在 review 中补充确认仓库内尚无调用 build() 的代码，该路径此前完全零测试覆盖。

实现拆解

1. 构造期校验前置：MiniMaxH3AdalnCache.__init__（minimax_h3.py）新增两项校验，max_plans < 1 与 max_plan_width < 1 直接抛 ValueError；后者按 review 建议在消息中点名 --minimax-h3-adaln-plan-width。这样非正容量在缓存构造时即被拒绝，而不是拖到第一个请求才以难读的错误暴露。
2. 预检从 missing 改为 wanted：build() 中容量与宽度校验改为对当前请求的完整计划集合 wanted 执行，且全部放在清空驻留元数据之前。非法请求直接抛错退出，不再破坏仍可复用的缓存（修复前宽度校验位于清空逻辑之后，超宽请求会连带驱逐两个驻留计划）。
3. 重置语义修正：新增 reset = len(self._slots) + len(missing) > self.max_plans 判定，并令 plans_to_build = wanted if reset else missing。修复前重置后只重建 missing 集合，导致重置前命中、重置时被清出的计划（如 plan_a）丢失；现在重置后重建整个 wanted 集合。
4. 事务性元数据发布：新增局部字典 pending_slots 暂存新 slot 分配（slot 编号基于清理后的 _slots 长度线性分配），所有 adaln_proj 层与 final_layer 投影全部写入

block_params/final_params、且 plan_lengths 激活之后，才执行 self._slots.update(pending_slots)。中途抛出异常时，未完成 slot 对 lookup() 保持不可见，后续请求可安全重试。

5. 测试与基础设施：测试文件抽取 _ARCH、_BLOCK_WIDTH、_FINAL_WIDTH 常量，新增 _ensure_single_process_parallel_runtime()（初始化 TP1/SP1 并行环境）、_write_online_weights()（生成支持按名省略键的 checkpoint）、_online_cache()、_embed() 辅助函数；新增三个场景测试分别锁定容量重置、失败重试、超宽拒绝。

关键文件：

- python/sglang/multimodal_gen/runtime/models/dits/minimax_h3.py（模块 缓存层；类别 source；类型 core-logic；符号 MiniMaxH3AdaInCache.init, MiniMaxH3AdaInCache.build, MiniMaxH3AdaInCache.lookup）：核心修复文件。MiniMaxH3AdaInCache.build() 的 slot 发布改为事务性：先校验、可重置、最后提交 _slots；构造期拒绝非正容量。
- python/sglang/multimodal_gen/test/unit/test_minimax_h3_adaln_cache.py（模块 缓存测试；类别 test；类型 test-coverage；符号 test_online_cache_reset_rebuilds_previously_resident_request_plans, test_online_cache_failed_rebuild_can_be_retried, test_online_cache_width_rejection_preserves_resident_plans, _online_cache）：新增 3 个缓存状态机测试（容量重置、失败重试、超宽拒绝），并重构测试基础设施以支持 online 缓存路径。

关键符号：MiniMaxH3AdaInCache.init, MiniMaxH3AdaInCache.build, MiniMaxH3AdaInCache.lookup, test_online_cache_reset_rebuilds_previously_resident_request_plans, test_online_cache_failed_rebuild_can_be_retried, test_online_cache_width_rejection_preserves_resident_plans, _online_cache, _write_online_weights, _embed, _ensure_single_process_parallel_runtime

关键源码片段

python/sglang/multimodal_gen/test/unit/test_minimax_h3_adaln_cache.py

新增 3 个缓存状态机测试（容量重置、失败重试、超宽拒绝），并重构测试基础设施以支持 online 缓存路径。

```
def test_online_cache_reset_rebuilds_previously_resident_request_plans(tmp_path):
    """容量重置不得丢弃当前请求复用的计划。"""
    cache = _online_cache(tmp_path, max_plan_width=1)
    plan_a = torch.tensor([1.0])
    plan_b = torch.tensor([2.0])
    plan_c = torch.tensor([3.0])

    # 第一次 build 填满两个槽位（max_plans 默认 2）
    cache.build([plan_a, plan_b], embed=_embed)

    # 第二次 build 的 wanted 为 {plan_a, plan_c}，其中 plan_a 原已驻留。
    # len(_slots) + len(missing) = 2 + 1 > 2 触发重置；修复前只重建
    # missing 里的 plan_c，plan_a 被清出后 lookup 失败——本测试即
```

```
# 用于锁定该回归。
cache.build([plan_a, plan_c], embed=_embed)

cache.lookup(plan_a)
cache.lookup(plan_c)
```

评论区精华

triple-mu 在首个 inline 评论中给出关键证据：他在 main 与本分支上分别运行三个场景，均为“修改前失败、修改后通过”；同时指出 repo 中还没有调用 build() 的代码，--minimax-h3-adaln-online 完全没有测试覆盖，并将“缺失回归测试”列为唯一阻塞项。第二个评论关注 max_plan_width 的错误消息——argparse 以裸 int 接收该 flag，用户确实可能传 0，建议新校验消息像旧路径一样点名 --minimax-h3-adaln-plan-width。作者采纳后 triple-mu 批准，并提示 CI 因缺少 run-ci 标签未运行，由 mickqian 补充 /tag-and-rerun-ci 后完成合并。

- 三个失败场景复现与回归测试缺口 (testing): 作者补齐 3 个单元测试并重构测试基础设施，triple-mu 在 head 7b4fb15a6d 上无遗留问题后批准。
- max_plan_width 错误消息应指出 CLI flag (design): 作者按建议在消息中加入 "set --minimax-h3-adaln-plan-width to at least 1"，triple-mu 标注 non-blocking 后采纳。

风险与影响

- 风险：
  1. 提交顺序依赖：事务性发布要求 plan_lengths 先激活、_slots 后提交，两者顺序颠倒会重新引入“假命中”问题；未来维护需保持该不变式。
  2. 多 rank 场景未覆盖：新增测试全部运行在 TP1 单进程环境，TP>1 时重建依赖 all-gather 与 checkpoint 分片读取，collective 中途失败的异常路径没有被测试覆盖。
  3. 错误消息契约变化：容量与宽度校验从基于 missing 改为基于 wanted，措辞与触发时机均变化，依赖字符串匹配的外部监控或脚本可能受影响。
  4. 影响范围受功能启用状态约束：triple-mu 指出仓库内尚无 build() 调用者，因此本修复的实际线上影响取决于 --minimax-h3-adaln-online 是否被启用，当前更多是消除潜伏的正确性隐患。 - 影响：对用户：启用 --minimax-h3-adaln-online 的 MiniMax-H3 服务不再因容量重置后缓存查找失败而在 denoising 中途整请求失败，重建失败后也可安全重试。对系统：build() 的 checkpoint 读取、GEMM 投影与通信路径保持不变，正常成功路径仅增加一次 dict 的 update 拷贝，无性能开销。对团队：确立了“先校验、后清空、最后发布”的缓存事务范式，可作为 Cache-DiT 等其他 diffusion 缓存的参考；同时补齐了该模块的测试基础设施（单进程并行环境初始化、checkpoint 生成辅助），降低后续改动门槛。 - 风险标记：缓存状态机一致性，此前零测试覆盖路径，TP>1 失败重试未覆盖，错误消息契约变化

关联脉络

- PR #34581 [Diffusion] Optimizing MiniMax-H3 for consumer-level GPUs: INT8 Linear + pluggable DiT attention backends: 同一 MiniMax-H3 功能线：AdaLN online 缓存的按需重建是该项目引入的吞吐优化路径，本 PR 修复该缓存路径的正确性。

- PR #34680 [diffusion][Minimax H3]support subblock sparse attention on SM90: 同一 MiniMax-H3 模型在注意力后端与硬件层面的扩展，共享 minimax_h3.py 中的缓存与层实现。
- PR #35339 [diffusion] Per-request lossy accelerations: Cache-DiT, CFG gating, attention backend override: diffusion 管线把有损加速改为按请求开关，与 AdaLN 缓存的精确性 / 吞吐权衡管理同属一个演进方向。