

PR #39162 完整报告

sgl-project/sglang

[HiCache][LoRA] Simplify decode offload hash inputs

合并时间: 2026-09-14 11:23

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39162>

执行摘要

- 一句话: 简化 decode 卸载哈希入参, 改由 req 读取命名空间
- 推荐动作: 值得快速阅读, 不必精读。价值在两点: 一是 'stacked PR + review 收口 + 尽量不动已测 head' 的协作模式; 二是 `_compute_prefix_hash` 的接口收敛——命名空间读取集中到 `storage_namespace_seed(req.extra_key, req.cache_salt)`, 避免调用方漏传参数。若要理解 HiCache + LoRA 的存储哈希命名空间隔离, 建议结合 #38577 与 #37058 一起看。

功能与动机

PR body 说明: 'Address the two nonblocking review comments in #38577 without changing its tested head. This PR is stacked on #38577 and should merge after it; only ff2c6c6ee5 is new.' 即 #38577 引入 LoRA / cache namespace 哈希链后, `_compute_prefix_hash` 要求调用方重复传 `extra_key` 与 `cache_salt`, review 意见希望简化。本 PR 把命名空间读取下沉到 seed helper, 并保留 '#38577 已测 head 不变' 的约束。commit b8ba4131 还提到该功能线承接 #37058 的 cache namespace 需求, 目标是让 LoRA 适配器与基础请求在存储端互不共享 KV 页。

实现拆解

1. 接口收敛: python/sglang/srt/disaggregation/decode_kvcache_offload_manager.py 中 `_compute_prefix_hash` 签名由 (tokens, prior_hash="", extra_key=None, cache_salt=None) 改为 (req: Req, tokens, prior_hash=""); 内部用 `storage_namespace_seed(req.extra_key, req.cache_salt)` 作为新哈希链的种子。
2. 调用点同步: offload_kv_cache 中 prefill 段哈希改为 `_compute_prefix_hash(req, req.origin_input_ids[:prefill_offloaded_len])`; `_trigger_backup` 中增量段哈希改为 `_compute_prefix_hash(req, incremental_tokens, prior_hash)`。哈希 continuation (prior_hash 链) 逻辑原样保留。
3. 测试适配: test/registered/unit/disaggregation/test_specv2_kvcache_offloading.py 的 `test_decode_offload_hash_chain_matches_prefill` 改为用 `SimpleNamespace(**namespace)` 构造 req 传入, 仍覆盖“无、仅 extra_key、仅 cache_salt、两者兼有”四组组合, 并与 `RadixKey(tokens,**namespace)` 的参考哈希对照。
4. 验证配套: pre-commit 全部通过; targeted 用例因本机无 Triton 未能本地收集, 依赖 CI 验证 (PR Test 与 Extra 均通过, AMD ROCm 10 运行失败需另查)。无配置、schema、

部署或文档改动，也未做性能测试——PR body 注明 “argument refactor only”。

关键文件：

- python/sglang/srt/disaggregation/decode_kvcache_offload_manager.py（模块 解码卸载；类别 source；类型 core-logic；符号 _compute_prefix_hash）：核心源码文件：_compute_prefix_hash 签名从显式命名空间参数收敛为直接接收 req，并在内部读取 req.extra_key / req.cache_salt，两处调用点（offload_kv_cache、_trigger_backup）同步更新。
- test/registered/unit/disaggregation/test_specv2_kvcache_offloading.py（模块 卸载测试；类别 test；类型 test-coverage）：测试配套：test_decode_offload_hash_chain_matches_prefill 改为用 SimpleNamespace 构造 req 传参，保留四组命名空间组合的链式哈希对照，验证 decode 链与 prefill 命名空间一致。

关键符号：_compute_prefix_hash, offload_kv_cache, _trigger_backup

关键源码片段

python/sglang/srt/disaggregation/decode_kvcache_offload_manager.py

核心源码文件：_compute_prefix_hash 签名从显式命名空间参数收敛为直接接收 req，并在内部读取 req.extra_key / req.cache_salt，两处调用点（offload_kv_cache、_trigger_backup）同步更新。

```
# python/sglang/srt/disaggregation/decode_kvcache_offload_manager.py
def _compute_prefix_hash(self, req: Req, tokens, prior_hash=""):
    """Match prefill storage hashes."""
    page_hashes = []
    # 新链首次播种（无 prior_hash）时，从 req 读取 extra_key 与 cache_salt。
    # 这样 LoRA 适配器与基础请求在存储侧的哈希链天然隔离，不会共享 KV 页。
    last_hash = prior_hash or storage_namespace_seed(req.extra_key, req.cache_salt)
    for offset in range(0, len(tokens), self.page_size):
        page_tokens = tokens[offset : offset + self.page_size]
        last_hash = self.cache_controller.get_hash_str(page_tokens, last_hash)
        page_hashes.append(last_hash)
    return page_hashes

# 调用点 1: offload_kv_cache 中对 prefill 段求哈希
prefill_hashes = self._compute_prefix_hash(
    req, req.origin_input_ids[:prefill_offloaded_len]
)

# 调用点 2: _trigger_backup 中对增量段求哈希，继续已有链
page_hashes = self._compute_prefix_hash(req, incremental_tokens, prior_hash)
```

test/registered/unit/disaggregation/test_specv2_kvcache_offloading.py

测试配套：test_decode_offload_hash_chain_matches_prefill 改为用 SimpleNamespace 构造 req 传参，保留四组命名空间组合的链式哈希对照，验证 decode 链与 prefill 命名空间一致。

```
# test/registered/unit/disaggregation/test_specv2_kvcache_offloading.py
# 四组命名空间组合：无、仅 extra_key、仅 cache_salt、两者兼有
for extra_key, cache_salt in [
    (None, None), ("lora-a", None), (None, "tenant-a"), ("lora-a", "tenant-a"),
]:
    with self.subTest(extra_key=extra_key, cache_salt=cache_salt):
        namespace = dict(extra_key=extra_key, cache_salt=cache_salt)
        req = SimpleNamespace(**namespace) # 用 req 对象携带命名空间字段
        prefix = manager._compute_prefix_hash(req, tokens[:4])
        tail = manager._compute_prefix_hash(req, tokens[4:], prefix[-1])
        self.assertEqual(
            prefix + tail,
            get_storage_hash_str(RadixKey(tokens, **namespace), page_size=2),
        )
```

评论区精华

- 本 PR 没有任何 review 评论，两位 reviewer (alphabtc1、ShangmingCai) 直接 APPROVED。
- alphabtc1 在 Issue 侧执行 /tag-and-rerun-ci 触发 CI 重跑，对应 labels 中的 run-ci、run-ci-extra、bypass-fastfail。
- 合入前 alphabtc1 提交了两个 commit: [8a6e3f34](#) (解决 main 冲突) 与 [47de2b47](#) ([refactor: pass explicit fields to storage namespace seed](#))，说明 main 冲突解决后又对字段传递做了二次收口，最终形态与 ff2c6c6ee5 一致。
- 无 review 讨论，仅 CI 重跑与 approve (other): 两位 reviewer 直接通过；最终形态为 [_compute_prefix_hash\(req, tokens, prior_hash\)](#) 并从 req 读取 [extra_key / cache_salt](#)。

风险与影响

- 风险：
 - 行为等价性：哈希编码与链式 continuation 完全不变，唯一变化是命名空间字段的读取位置从调用方移入 helper。
 - 字段缺失风险：[_compute_prefix_hash](#) 现在直接读 `req.extra_key / req.cache_salt`，若将来出现缺少这两个属性的 mock 或特殊 Req 构造路径，会抛 `AttributeError`；测试中已用 `SimpleNamespace` 覆盖四种组合。
 - 调用点遗漏风险：[_compute_prefix_hash](#) 是私有方法，当前两处调用点均已更新；但若外部模块以旧签名调用会 `TypeError`，这是有意为之的接口收口。
 - 依赖顺序：本 PR 依赖 #38577 已合入 main，否则 `storage_namespace_seed` 与 `req.extra_key` 会缺失。
 - CI 噪音：AMD ROCm 10 workflow 本次失败 (Run #34704433460)，与本次 Python 层重构大概率无关，但合入前建议确认为已知平台问题。
- 影响：
 - 用户与服务：运行时行为不变，无新配置项，KV offload 的哈希结果与存储命中行为完全一致。

- 系统层面：decode 侧 offload 管理器的接口更简洁，命名空间读取集中到 `storage_namespace_seed` 一处，与 `prefill` / `Rust` 侧语义对齐。
- 团队层面：清理 #38577 遗留的 review 意见，降低后续维护成本；改动面仅 2 个文件、+7/-15，回归面很小。
- 风险标记：行为等价重构，依赖 #38577 先合入，`req` 字段缺失有 `AttributeError` 风险，AMD ROCm CI 未通过待观察

关联脉络

- PR #38577 [HiCache][LoRA] Isolate storage pages by extra key (父 PR, stack 基础) : PR body 明确本 PR stacked on #38577 并应在其后合入；5ff49e84 与 b8ba4131 两个 commit 均来自 #38577，只有 ff2c6c6ee5 是本 PR 的新提交。
- PR #38486 [HiCache] Publish a host store event for storage-prefetch refills: 同属 HiCache + unified radix cache 存储哈希功能线，改动 `rust/sglang-radix-tree` 与 `unified cache` 测试，与本 PR 的 `storage hash / namespace` 语义一致，可作演进上下文。