PR #33613 完整报告

sgl-project/sglang

Remove revoke queue after hit-then-alloc refactoring

合并时间：2026-08-06 18:10

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33613

执行摘要

- 一句话：删除 prefetch revoke 队列，撤销判断并入调度线程
- 推荐动作：值得快速浏览，作为 #19320（hit-then-alloc）重构的收尾，展示了「当决策所需信息到达时再决策」如何简化跨线程通信。如果想深入 HiCache 异步管线，建议按 #19320 → 本 PR 的顺序阅读。本 PR 本身无讨论、无测试配套，不需要精读。

功能与动机

PR body 明确说明这是 #19320 的 follow-up code cleaning：#19320 将主机内存分配推迟到 storage hit 之后（hit-then-alloc），使得调度线程在 drain 时已经拿到 operation.storage_hit_count，可以就地决定 revoke 或分配内存，原来用于跨线程通知撤销的 prefetch_revoke_queue 不再是必需。移除它可以消除一条队列、一次跨线程投递和一组参数传递，让异步控制流更简洁。

实现拆解

1. 删除队列生命周期管理（python/sglang/srt/managers/cache_controller.py）：_start_storage_threads() 不再创建 prefetch_revoke_queue，reset() 不再对该队列做 clear()，避免 attach/reset 路径残留无用队列。
2. 简化 prefetch 线程（python/sglang/srt/managers/cache_controller.py）：prefetch_thread_func() 移除 storage_hit_count < self.prefetch_threshold 的分流分支，不再向 revoke 队列投递 request_id；所有结果统一截断 hash_value 并写入 storage_hit_count 后投入 prefetch_hit_queue，把决策权完整交给调度线程。
3. 收敛 drain 决策（三个 radix cache 实现）：python/sglang/srt/mem_cache/hiradix_cache.py、python/sglang/srt/mem_cache/unified_radix_cache.py、python/sglang/srt/mem_cache/hi_mamba_radix_cache.py 中删除 _drain_revoke() 嵌套函数、_drain_storage_control_queues_impl() 的 n_revoke 形参及末尾的 _drain_revoke() 调用；在 _drain_and_alloc_storage_hit() 头部新增 operation.storage_hit_count < self.prefetch_threshold 分支，触发时调用 _revoke_pending_prefetch(req_id) 并 continue。
4. 调整队列统计与 TP 同步：drain_storage_control_queues()、check_hicache_events() 及 unified 的 _sync_hicache_ready_counts() 中移除 prefetch_revoke_queue.qsize()，all-reduce 向量从 4 元缩为 3 元；unified 中 extra_release_counts 的切片索引相应从 [4:] 改为 [3:]。

5. 配套与测试：没有新增或修改测试文件，依赖现有 HiCache 测试回归；由于是行为等价迁移，对模型输出无影响。

关键文件：

- python/sglang/srt/mem_cache/hiradix_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 _drain_revoke, _drain_storage_control_queues_impl, _drain_and_alloc_storage_hit, drain_storage_control_queues）：HiCache 非 Mamba 路径的调度线程 drain 逻辑所在文件：删除 _drain_revoke 与 n_revoke 参数，threshold 撤销判断迁入 _drain_and_alloc_storage_hit。
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 _drain_revoke, _drain_storage_control_queues_impl, _drain_and_alloc_storage_hit, drain_storage_control_queues）：unified radix cache 版本的同类改动：除删除 _drain_revoke 外，还涉及 drain_storage_control_queues、check_hicache_events 与 _sync_hicache_ready_counts 的队列统计调整，以及 extra_release_counts 索引从 [4:] 改为 [3:]。
- python/sglang/srt/mem_cache/hi_mamba_radix_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 _drain_revoke, _drain_storage_control_queues_impl, _drain_and_alloc_storage_hit, drain_storage_control_queues）：Mamba 场景的 HiCache 实现，改动同 hiradix（删除 _drain_revoke 与 n_revoke，threshold 判断前移），保证三种 radix cache 实现行为一致。
- python/sglang/srt/managers/cache_controller.py（模块 缓存层；类别 source；类型 entrypoint；符号 prefetch_revoke_queue, prefetch_thread_func, reset, _start_storage_threads）：队列的「所有者」：prefetch_revoke_queue 在这里创建、清理并在 prefetch_thread_func 中投递；本 PR 移除其创建和 reset 清理，并让 prefetch 线程不再做阈值分流。

关键符号：prefetch_thread_func, _drain_storage_control_queues_impl, _drain_and_alloc_storage_hit, drain_storage_control_queues, check_hicache_events, _sync_hicache_ready_counts, _drain_revoke

关键源码片段

python/sglang/srt/mem_cache/hiradix_cache.py

HiCache 非 Mamba 路径的调度线程 drain 逻辑所在文件：删除 _drain_revoke 与 n_revoke 参数，threshold 撤销判断迁入 _drain_and_alloc_storage_hit。

```
# 该函数是 _drain_storage_control_queues_impl 的嵌套函数，由调度线程调用；
# cc 即 self.cache_controller。drain 时 storage hit 数已经由 prefetch 线程
# 经过 TP all-reduce 后写入 operation.storage_hit_count。
def _drain_and_alloc_storage_hit():
    # hit-then-alloc 的核心：storage hit 数此刻已确定，只需预留恰好那么多的 host 内存，
    # 不再提前过度分配。alloc/evict 是 rank 局部的，但对 TP 各 rank 是确定性的：
    # host 池变更只发生在调度线程的 lockstep 点，按 TP-min 计数 drain，
    # 因此每个 rank 都会得到相同的成功 / fallback / revoke 结论。
    for operation in _drain_queue(cc.prefetch_hit_queue, n_storage_hit):
```

```
        req_id = operation.request_id
        info = self.ongoing_prefetch.get(req_id)
        if info is None:
            # 请求已被 abort/ 清理，直接跳过
            continue
        if operation.is_terminated():
            # 请求在 storage 查询期间被 abort，撤销其 pending prefetch
            self._revoke_pending_prefetch(req_id)
            continue
        if operation.storage_hit_count < self.prefetch_threshold:
            # 新增的阈值判断：收益不足不预取，就地撤销。
            # 原先该判断在 prefetch 线程中通过独立 revoke 队列异步通知，
            # 现在随 prefetch_revoke_queue 一起被删除，决策统一收拢到这里。
            self._revoke_pending_prefetch(req_id)
            logger.debug(
                f"Revoking prefetch for request {req_id} due to insufficient hits ({operation.storage_
                hit_count})."
            )
            continue

        alloc_len = operation.storage_hit_count
        host_indices = cc.mem_pool_host.alloc(alloc_len)
        if host_indices is None:
            self.evict_host(alloc_len)
            host_indices = cc.mem_pool_host.alloc(alloc_len)
        if host_indices is None:
            # 内存压力 fallback：退而求其次，分配页对齐的较短前缀
            available_size = cc.mem_pool_host.available_size()
            alloc_len = min(
                operation.storage_hit_count,
                available_size - (available_size % self.page_size),
            )
            if alloc_len >= self.prefetch_threshold:
                host_indices = cc.mem_pool_host.alloc(alloc_len)
        if host_indices is None:
            # 仍分配失败则撤销本次 prefetch，不阻塞请求
            self._revoke_pending_prefetch(req_id)
            logger.debug(
                f"Revoking prefetch for request {req_id} due to host memory allocation failure."
            )
            continue

        operation.storage_hit_count = alloc_len
        operation.hash_value = operation.hash_value[: alloc_len // self.page_size]
        operation.host_indices = host_indices
        cc.prefetch_buffer.put(operation)
```

python/sglang/srt/managers/cache_controller.py

队列的「所有者」：prefetch_revoke_queue 在这里创建、清理并在 prefetch_thread_func 中投递；本 PR 移除其创建和 reset 清理，并让 prefetch 线程不再做阈值分流。

```
def prefetch_thread_func(self):
    """管理从 storage 后端到 host 内存的 prefetch 操作。"""
    self.prefetch_buffer = Queue()
    self.prefetch_io_aux_thread = threading.Thread(
        target=self.prefetch_io_aux_func, daemon=True
    )
    self.prefetch_io_aux_thread.start()
    while (not self.storage_stop_event.is_set()) or not self.prefetch_queue.empty():
        try:
            operation = self.prefetch_queue.get(block=True, timeout=1)
            if operation is None:
                continue
            if operation.is_terminated():
                hash_value, storage_hit_count = [], 0
            else:
                hash_value, storage_hit_count = self._storage_hit_query(operation)
            storage_hit_count_tensor = torch.tensor(storage_hit_count, dtype=torch.int)
            self._all_reduce_prefetch_groups(
                storage_hit_count_tensor, torch.distributed.ReduceOp.MIN
            )
            storage_hit_count = storage_hit_count_tensor.item()

            # 这里不再判断 storage_hit_count 是否低于 prefetch_threshold：
            # 阈值判断被下沉到调度线程的 drain 阶段（_drain_and_alloc_storage_hit），
            # 因此本线程只需把结果统一投递到 prefetch_hit_queue 即可。
            operation.hash_value = hash_value[: (storage_hit_count // self.page_size)]
            operation.storage_hit_count = storage_hit_count
            self.prefetch_hit_queue.put(operation)
        except Empty:
            continue
```

评论区精华

本 PR 无任何 review 评论或 issue 讨论，唯一审核记录为 hzh0425 的 APPROVED（同意，无附加说明）。设计的核心取舍由 PR body 引用的 #19320 支撑：内存分配推迟后，撤销决策所需的信息在 drain 时已齐备，独立 revoke 队列不再有意义。

- 暂无高价值评论线程

风险与影响

- 风险：行为等价性：_drain_revoke() 原先在 _drain_and_alloc_storage_hit() 之前执行，现在同一 operation 的撤销要在它排到 prefetch_hit_queue 头部时触发；由于分配同样依赖 hit 队列顺序，二者对同一请求不存在竞争窗口，语义保持一致。遗漏引用风险：prefetch_revoke_queue 在创建与 reset 路径的引用均已删除，_stop_storage_threads() 从可见代码看未引用该队列；仍需确认 detach/re-attach 组合路径不残留对该属性名的访

问。回归面：改动仅影响 enable_storage 开启时的 HiCache storage 路径，普通推理不触碰；缺少针对该队列删除的直接单测，依赖既有 HiCache 测试覆盖。性能影响：减少一个队列、一次跨线程投递与一个 TP 同步向量元素，调度线程仅增加一次阈值比较，整体开销可忽略甚至略优。

- 影响：对用户：HiCache storage 后端（prefetch/backup 到远端存储）路径行为不变，模型输出不受影响；普通推理路径完全不触碰这些代码。对系统：调度线程 drain 逻辑成为 revoke 与分配的唯一决策点，异步控制流更简单，减少一个潜在的跨线程不一致来源（例如队列长度在 TP 之间不同步时需要特殊处理）。对团队：后续改动 HiCache 存储管线只需关注 prefetch_hit_queue 单队列语义；结合 #19320 的 hit-then-alloc 设计，新成员更容易理解本 PR 的清理动机。
- 风险标记：核心异步控制流变更，缺少测试覆盖，多文件同步删除需防遗漏引用

关联脉络

- PR #19320 Defer host memory allocation (hit-then-alloc): PR body 明确声明本 PR 是其 follow-up code cleaning；#19320 推迟 host 内存分配，使 revoke 队列不再必要。标题按 PR body 描述转述。
- PR #30393 [HiCache] Support packed and sidecar draft caches for MTP/EAGLE/DSpark: 同属 HiCache 存储 / 缓存管线演进，涉及 hybrid_cache_controller、radix cache 等文件，与本 PR 处于同一功能线，但不是直接依赖。