PR #39401 完整报告

vllm-project/vllm

[kv_offload+HMA][9/N]: Support lookup with multiple KV groups

合并时间：2026-04-24 20:32

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/39401

执行摘要

- 一句话：支持多 KV 组查找，移除单组限制
- 推荐动作：建议关注此 PR 的设计模式：如何逐步移除单组限制并引入循环。核心变更集中在单一文件，逻辑清晰，但缺少测试覆盖。后续系列 PR 需要密切配合验证。建议在合并前补充集成测试。

功能与动机

PR 正文说明：'This PR extends the offloading connector to support lookups where KVCacheConfig contains multiple groups.' 这是 [kv_offload+HMA] 系列的一部分，旨在支持异构内存访问场景下的多组 KV 缓存。

实现拆解

1. 初始化查找组：在 OffloadingConnectorScheduler.__init__ 中（scheduler.py 第 117 行左右），枚举 spec.kv_cache_config.kv_cache_groups 中的所有组，构建 self.lookup_groups 列表（当前视为全注意力组，后续可扩展）。
2. 移除单组断言：删除 get_num_new_matched_tokens 中的 assert len(self.config.kv_group_configs) == 1 和 assert len(req_status.group_states) == 1，改为遍历 self.lookup_groups。同时将 update_offload_keys() 调用移至请求首次创建时的 else 分支，确保键只初始化一次。
3. 循环处理各组：对于每个组，获取 GroupOffloadConfig 和对应的 offload_keys，计算块对齐的 max_hit_size_tokens，从 start_block_idx 切片后调用 _maximal_prefix_lookup 查找可用块，更新限制。
4. 聚合结果与延迟处理：聚合所有组的限制得到 num_hit_tokens；若某组查找返回 None 则标记 defer_lookup，若存在正在加载的块则标记 delay_request；最终根据标志返回延迟或实际命中数。

关键文件：

- vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py（模块 KV 卸载调度器；类别 source；类型 core-logic；符号 init, get_num_new_matched_tokens）：唯一变更文件，重构了调度器的核心查找方法以支持多 KV 组。

关键符号：init, get_num_new_matched_tokens

关键源码片段

vllm/distributed/kv_transfer/kv_connector/v1/offloading/scheduler.py

唯一变更文件，重构了调度器的核心查找方法以支持多 KV 组。

```
def get_num_new_matched_tokens(
    self, request: Request, num_computed_tokens: int
) -> tuple[int | None, bool]:
    """
    获取在 num_computed_tokens 之外还能匹配多少个新 token。
    这里支持多个 KV 组的查找，每个组独立进行前缀匹配后取交集。
    """
    if req_status := self._req_status.get(request.request_id):
        # 清除旧块 ID，重新加载
        for group_state in req_status.group_states:
            group_state.block_ids.clear()
    else:
        # 首次创建请求状态并预计算 offload keys
        req_status = RequestOffloadState(config=self.config, req=request)
        self._req_status[request.request_id] = req_status

    req_status.update_offload_keys()
    req_status.num_locally_computed_tokens = num_computed_tokens

    # 先 touch 所有组的 keys，确保异步加载信息新鲜
    for gs in req_status.group_states:
        self.manager.touch(gs.offload_keys)

    max_hit_size_tokens: int = req_status.req.num_tokens
    defer_lookup = False
    delay_request = False

    # 依次处理每个注意力组
    for group_idx in self.lookup_groups:
        group_config = self.config.kv_group_configs[group_idx]
        offloaded_block_size = group_config.offloaded_block_size
        offload_keys = req_status.group_states[group_idx].offload_keys
        num_blocks = max_hit_size_tokens // offloaded_block_size
        assert len(offload_keys) >= num_blocks

        # 对齐到块边界，限制最大候选大小
        max_hit_size_tokens = num_blocks * offloaded_block_size
        num_hit_tokens = max_hit_size_tokens - num_computed_tokens
        if num_hit_tokens < offloaded_block_size:
            return 0, False

        start_block_idx = num_computed_tokens // offloaded_block_size
        keys_to_lookup = offload_keys[start_block_idx:num_blocks]
```

```
            # 全注意力依赖所有历史块，找最长前缀匹配
            block_hits = self._maximal_prefix_lookup(
                keys_to_lookup, req_status.req_context
            )
            if block_hits == 0:
                return 0, False
            if block_hits is None:
                defer_lookup = True
            else:
                # 更严格的限制：仅取实际命中的部分
                max_hit_size_tokens = offloaded_block_size * (start_block_idx + block_hits)

        num_hit_tokens = max_hit_size_tokens - num_computed_tokens
        if num_hit_tokens < 0:
            return 0, False

        # 检查是否有任何块正在异步加载中
        for group_state in req_status.group_states:
            if group_state.block_ids:
                delay_request = True
                break

        if defer_lookup or delay_request:
            return None, True
        return num_hit_tokens, True
```

评论区精华

Review 中主要讨论了三点：

- 拼写错误：markmc 指出注释中 'relays' 应为 'relies'，已修复。
- 延迟查找日志：markmc 建议区分延迟查找和请求延迟的日志，让读者更清晰；orozery 增加了更多解释但保留了原有结构，认为可减少一次比较。
- 代码清晰度：markmc 建议简化最大命中大小的计算注释，orozery 采纳了部分注释但维持现有写法。
  - 拼写错误 'relays' 应为 'relies' (style): 已修复，head 版本中已使用正确拼写 'relies'。
  - 延迟查找与请求延迟的日志区分 (design): 作者增加日志细节，但未彻底重构；双方认可当前方案。
  - 代码清晰度与注释简化 (design): 部分采纳，增加注释但未改变结构。

风险与影响

- 风险：
  1. 回归风险：移除两个断言可能允许不符合预期配置的执行路径，需确保上游已正确配置多组。

2. 性能影响：循环遍历多个组可能增加每步调度开销，但当前假设所有组都参与查找，组数通常较小（2-3 个），影响可接受。
3. 正确性风险：max_hit_size_tokens 在多组间取交集，必须确保各组查找语义一致（目前均用 _maximal_prefix_lookup），未来若引入滑动窗口等需适配。
4. 缺少测试覆盖：变更仅涉及源码，无直接测试配套，可能存在边缘情况未覆盖。 - 影响：本 PR 是 kv_offload+HMA 系列的第 9 个，影响内部调度器行为。使用方（开发者）需确保配置了正确的 kv_cache_groups。用户侧无直接感知，但它是后续支持多组 KV 卸载和 HMA 的基础。对系统的影响范围限于卸载连接器调度器模块。 - 风险标记：核心路径变更，缺少测试覆盖，断言移除，多组兼容性

关联脉络

- 暂无明显关联 PR