PR #26615 完整报告

sgl-project/sglang

[sgl] Window-aware LRU refresh for SWA prefix cache in unified cache

合并时间：2026-06-01 19:35

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26615

执行摘要

- 一句话：SWA LRU 窗口感知刷新，提升前缀缓存命中率
- 推荐动作：值得精读，尤其关注 refresh_lru 的接口设计（使用 phase 参数统一不同阶段的刷新逻辑）和窗口限制祖先刷新的实现。对于维护 SWA 或其他特殊缓存策略的开发者有参考价值。

功能与动机

PR 描述指出：当前 radix 缓存使用 LRU 逐出 SWA 缓存，每次前缀匹配和插入过程中的向下遍历都会将匹配节点的所有祖先节点刷到 MRU。这对 Full KV 是正确的，但对 SWA 错误，因为 SWA 只有最后 sliding_window_size 个 token 对当前解码步骤是可达的。导致窗口外节点不断被刷新而不被逐出，真正的热点缓存反而被提前逐出，多轮压力下 SWA 池被窗口外缓存占满，命中率下降，逐出颠簸增加。

实现拆解

- 步骤一：定义 LRU 刷新阶段枚举 在 tree_component.py 中添加 LRURefreshPhase 枚举（WALKDOWN、MATCH_END、INSERT_END），为不同操作阶段提供 LRU 刷新上下文。
- 步骤二：在基类中实现基础 refresh_lru 方法 在 TreeComponent 基类中添加 refresh_lru(phase, node, root_node) 方法，根据不同的 phase 执行不同的 LRU 刷新逻辑：WALKDOWN 时仅刷新当前节点；MATCH_END 时刷新节点及其所有祖先；INSERT_END 时无需额外操作。
- 步骤三：在 SWAComponent 中覆盖 refresh_lru 实现窗口感知刷新 SWA 组件的 refresh_lru 在 WALKDOWN 阶段什么也不做（避免遍历时刷新错误节点），在 MATCH_END 和 INSERT_END 阶段调用 reset_node_and_window_ancestors_mru 仅刷新处于滑动窗口 + cushion 范围内的祖先节点。
- 步骤四：在 UnifiedLRUList 中添加窗口限制的祖先刷新方法 在 unified_radix_cache.py 的 UnifiedLRUList 类中新增 reset_node_and_window_ancestors_mru 方法，该方法从给定节点向上遍历，累计 key 长度直到超过 window_size，将途经且满足 should_include 条件的节点移到 MRU。
- 步骤五：修改核心缓存操作的调用点 在 unified_radix_cache.py 的 _touch_node、_match_post_processor、_insert_helper 中，将原来直接调用 reset_node_mru / reset_node_and_parents_mru 替换为调用组件的 refresh_lru 方法，并传入对应的 phase。

同时确保 Full 组件不走 LRU（只有非 Base 组件才刷新）。

- 配套：更新 __init__.py 导出 LRURefreshPhase；新增大量单元测试覆盖窗口感知刷新的各种场景。

关键文件：

- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 测试；类别 test；类型 test-coverage；符号 _swa_lru_order, _swa_pinning_cfg_supported, test_swa_lru_walk_down_does_not_refresh_ancestors_during_insert, test_swa_lru_match_only_refreshes_window_cushion）：新增 5 个测试用例（共 +297 行），覆盖窗口感知刷新的核心行为：插入时不刷新祖先、匹配时只刷新窗口内祖先、旧祖先优先被逐出、cushion 边界计算、以及整体 sanity check。同时添加辅助方法 _swa_lru_order 和 _swa_pinning_cfg_supported。
- python/sglang/srt/mem_cache/unified_cache_components/tree_component.py（模块 缓存层；类别 source；类型 core-logic；符号 LRURefreshPhase, refresh_lru）：添加 LRURefreshPhase 枚举和 refresh_lru 基方法，定义了 LRU 刷新策略的框架。
- python/sglang/srt/mem_cache/unified_radix_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 reset_node_and_window_ancestors_mru）：新增 reset_node_and_window_ancestors_mru 实现窗口限制祖先刷新；修改 _touch_node、_match_post_processor、_insert_helper 调用点以使用组件的 refresh_lru。
- python/sglang/srt/mem_cache/unified_cache_components/swa_component.py（模块 缓存层；类别 source；类型 core-logic；符号 refresh_lru）：覆盖 refresh_lru 实现 SWA 特定的窗口感知刷新逻辑。
- python/sglang/srt/mem_cache/unified_cache_components/__init__.py（模块 缓存层；类别 source；类型 core-logic）：导出 LRURefreshPhase 以保持接口完整性。

关键符号：LRURefreshPhase, TreeComponent.refresh_lru, UnifiedLRUList.reset_node_and_window_ancestors_mru, SWAComponent.refresh_lru

关键源码片段

python/sglang/srt/mem_cache/unified_cache_components/tree_component.py

添加 LRURefreshPhase 枚举和 refresh_lru 基方法，定义了 LRU 刷新策略的框架。

```
class LRURefreshPhase(str, Enum):
    """定义 LRU 刷新发生的阶段。"""
    WALKDOWN = "walkdown" # 在遍历树时触碰节点
    MATCH_END = "match_end" # 成功前缀匹配后
    INSERT_END = "insert_end" # 新 / 更新叶节点提交后

class TreeComponent(ABC):
    # ...
    def refresh_lru(
        self,
        phase: LRURefreshPhase,
```

```
        node: UnifiedTreeNode,
        root_node: UnifiedTreeNode,
    ) -> None:
        ct = self.component_type
        match phase:
            case LRURefreshPhase.WALKDOWN:
                # 如果节点没有该组件数据，则跳过
                if node.component_data[ct].value is None:
                    return
                # 仅将该节点移到 MRU
                self.cache.lru_lists[ct].reset_node_mru(node)
            case LRURefreshPhase.MATCH_END:
                # 将节点及其所有祖先节点移到 MRU
                self.cache.lru_lists[ct].reset_node_and_parents_mru(
                    node, root_node, self.node_has_component_data
                )
            case LRURefreshPhase.INSERT_END:
                # WALKDOWN 已经刷新了插入路径上的每个节点，无需额外操作
                return
            case _:
                raise ValueError(f"Unknown LRURefreshPhase: {phase}")
```

python/sglang/srt/mem_cache/unified_cache_components/swa_component.py

覆盖 refresh_lru 实现 SWA 特定的窗口感知刷新逻辑。

```
class SWAComponent(TreeComponent):
    # ...
    def refresh_lru(
        self,
        phase: LRURefreshPhase,
        node: UnifiedTreeNode,
        root_node: UnifiedTreeNode,
    ) -> None:
        match phase:
            case LRURefreshPhase.WALKDOWN:
                # 遍历时刷新祖先会错误地将窗口外节点移到 MRU，
                # 因此 SWA 在 WALKDOWN 阶段不做任何操作；
                # 窗口限制的刷新将在 MATCH_END / INSERT_END 执行。
                return
            case LRURefreshPhase.MATCH_END | LRURefreshPhase.INSERT_END:
                self.cache.lru_lists[
                    self.component_type
                ].reset_node_and_window_ancestors_mru(
                    node,
                    root_node,
                    window_size=self.sliding_window_size + self.cache.page_size,
                    should_include=self.node_has_component_data,
                )
```

```
            case _:
                raise ValueError(f"Unknown LRURefreshPhase: {phase}")
```

评论区精华

- ispobock提出：能否也测试 page_size > 1 的场景？ 作者 bixue2010回复：已添加。 影响：后续提交中增加了 page_size > 1 的拓扑测试。
- hzh0425提出：应避免添加过多接口，建议合并为一个带 phase 参数的方法，并指出 refresh_lru_on_walkdown 尚未在 swa_component 中实现。 作者 bixue2010回复：已改为更少的接口并使用 phase，同时添加注释说明 walkdown 阶段 SWA 组件为何无需操作。 结果：最终实现采用单一 refresh_lru(phase, node, root_node) 方法。
- 建议添加 page_size > 1 测试 (testing): 作者在后续提交中添加了相关测试。
- 接口设计：避免过多单独方法，使用 phase 参数统一 (design): 作者采纳建议，改为单一 refresh_lru(phase,⋯) 方法，并对 SWA 组件的 walkdown 添加注释说明为何不操作。

风险与影响

- 风险：
  - 回归风险：未修改 Full 和 Mamba 组件的 LRU 行为（基类 refresh_lru 与原有逻辑一致）；SWA 组件的行为变更通过大量单元测试验证，但缺少端到端的多轮对话性能基准。建议合并后关注 SWA 命中率和缓存逐出指标。
  - 兼容性：新增枚举和方法，对旧版本无影响。
  - 并发 / 线程安全：未分析；原 LRU 操作非线程安全（假设在单线程调度中运行），不引入新风险。
- 影响：
  - 用户：对于使用 SWA（滑动窗口注意力）的模型（如 DeepSeek、Qwen 2.5 等），在多轮对话等长上下文场景中，前缀缓存命中率应显著提升，解码延迟降低。
  - 系统：SWA 缓存池的利用率更合理，避免窗口外缓存长时间占据热点位置，减少逐出颠簸。
  - 团队：新增接口设计（phase 参数模式）为后续其他组件的差异化 LRU 策略提供了可扩展框架。
  - 风险标记：LRU 逐出策略变更，缺少端到端性能基准，SWA 组件路径变更

关联脉络

- PR #26870 Make unified tree SWA hicache tests faithful to write-through backup: 都修改了 test_unified_radix_cache_unittest.py，且涉及 SWA 缓存测试增强，与本 PR 的测试扩展有间接关联。
- PR #26607 Do not cap DeepSeek V4 PD prefill by SWA pool size: 涉及 SWA 缓存容量限制，与本 PR 的逐出策略改进属于同一功能域。