PR #31479 完整报告

sgl-project/sglang

perf(kv-events): coalesce cache events

合并时间：2026-08-14 04:36

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31479

执行摘要

- 一句话：合并 KV 缓存事件入队，降低事件数量与序列化开销
- 推荐动作：值得精读。核心的 _enqueue_kv_event 方法展示了如何在不破坏语义的前提下通过严格条件压缩事件流，是典型的性能优化模式。建议关注合并条件的完备性以及下游消费者的适配情况。

功能与动机

PR body 中明确说明：SGLang emitted a separate KV event for each page despite list-valued wire fields, adding avoidable event and serialization overhead.

实现拆解

实现步骤

1. 新增统一入队方法：在 python/sglang/srt/mem_cache/events.py 的 KVCacheEventMixin 中新增 _enqueue_kv_event(event)，作为所有 KV 缓存事件入队的唯一出口。
2. 定义合并规则：仅当相邻事件满足严格条件时合并。BlockRemoved 要求 medium 相同；BlockStored 要求 medium、lora_id、block_size、metadata（含 cache_salt）全等，且新事件的 parent_block_hash 等于队尾最后一块的哈希，保证父链连续。
3. 替换直接入队调用：将 _record_store_event、_record_remove_event、_record_all_cleared_event 中的 self.kv_event_queue.append(...) 全部替换为 self._enqueue_kv_event(...)。
4. 更新测试：新增 TestKVCacheEventQueue 覆盖合并成功与边界场景；将 RadixCache、Unified、Mamba、SWA、HiRadix 测试中“每页一个事件”的断言改为“多页合并为一个事件”；手动测试 test/manual/test_kv_events.py 同步支持多块事件。

关键文件：

- python/sglang/srt/mem_cache/events.py（模块 缓存事件；类别 source；类型 core-logic；符号 _enqueue_kv_event）：核心改动文件，新增 _enqueue_kv_event 统一事件入队并执行合并，直接影响所有 KV 缓存事件的生产路径。
- test/registered/unit/mem_cache/test_radix_cache_unit.py（模块 缓存测试；类别 test；类型 test-coverage；符号 _KVCacheEventQueue, TestKVCacheEventQueue,

test_enqueue_coalesces_compatible_stores,test_enqueue_coalesces_compatible_removes）：新增 TestKVCacheEventQueue 覆盖合并成功与边界场景，并更新既有断言以适配多块事件语义。

- test/registered/unit/mem_cache/test_unified_radix_cache_unittest.py（模块 缓存测试；类别 test；类型 test-coverage）：更新 UnifiedRadixCache 的 KV 事件断言，适配合并后的单事件多块语义。
- test/registered/unit/mem_cache/test_mamba_unittest.py（模块 缓存测试；类别 test；类型 test-coverage）：更新 MambaRadixCache 的事件测试，验证合并后的多块 token_ids 与哈希列表。
- test/registered/unit/mem_cache/test_swa_unittest.py（模块 缓存测试；类别 test；类型 test-coverage）：更新 SWA 缓存事件测试，适配合并后的 store 事件格式。
- test/registered/unit/mem_cache/test_hiradix_cache_unit.py（模块 缓存测试；类别 test；类型 test-coverage）：更新 HiRadix 缓存测试，验证拆分片段合并为单个父链事件。
- test/manual/test_kv_events.py（模块 手动测试；类别 test；类型 test-coverage）：端到端手动测试，验证真实服务器发出的合并事件包含多块。

关键符号：_enqueue_kv_event, _record_store_event, _record_remove_event, _record_all_cleared_event

关键源码片段

python/sglang/srt/mem_cache/events.py

核心改动文件，新增 _enqueue_kv_event 统一事件入队并执行合并，直接影响所有 KV 缓存事件的生产路径。

```
def _enqueue_kv_event(self, event):
    """Append an event, coalescing it with a compatible queue tail.

    KV event batches already support multiple block hashes.  Combining them
    here avoids emitting one event per page while preserving ordering and
    the parent-linked store chains consumers use to rebuild the cache tree.
    """
    if self.kv_event_queue:
        tail = self.kv_event_queue[-1]

        # 同类型 `BlockRemoved`：只要存储介质一致即可合并 block 哈希列表。
        if isinstance(tail, BlockRemoved) and isinstance(event, BlockRemoved):
            if tail.medium == event.medium:
                tail.block_hashes.extend(event.block_hashes)
                return

        # 同类型 `BlockStored`：要求介质、LoRA、块大小、元数据（含 cache_salt）
        # 完全一致，并且新事件的父哈希必须为队尾最后一块的哈希，
        # 才能保证合并后仍是一条连续的父链。
        elif isinstance(tail, BlockStored) and isinstance(event, BlockStored):
            tail_metadata = (
```

```
                tail.metadata if isinstance(tail, BlockStoredWithMetadata) else None
            )
            event_metadata = (
                event.metadata if isinstance(event, BlockStoredWithMetadata) else None
            )
            if (
                tail.medium == event.medium
                and tail.lora_id == event.lora_id
                and tail.block_size == event.block_size
                and tail_metadata == event_metadata
                and tail.block_hashes
                and event.parent_block_hash == tail.block_hashes[-1]
            ):
                tail.block_hashes.extend(event.block_hashes)
                tail.token_ids.extend(event.token_ids)
                return

    # 不满足合并条件（不同类型、不同介质、链断裂等）则直接入队。
    self.kv_event_queue.append(event)
```

test/registered/unit/mem_cache/test_radix_cache_unit.py

新增 TestKVCacheEventQueue 覆盖合并成功与边界场景，并更新既有断言以适配多块事件语义。

```
class _KVCacheEventQueue(KVCacheEventMixin):
    def __init__(self):
        self.enable_kv_cache_events = True
        self.kv_event_queue = []


class TestKVCacheEventQueue(unittest.TestCase):
    @staticmethod
    def _store(
        block_hash: int,
        parent_block_hash: int | None,
        *,
        block_size: int = 2,
        medium: StorageMedium = StorageMedium.GPU,
        lora_id: int | None = None,
        cache_salt: str | None = None,
    ) -> BlockStored:
        event_args = dict(
            block_hashes=[block_hash],
            parent_block_hash=parent_block_hash,
            token_ids=[block_hash, block_hash + 1][:block_size],
            block_size=block_size,
            lora_id=lora_id,
            medium=medium,
        )
```

```
        if cache_salt is None:
            return BlockStored(**event_args)
        return BlockStoredWithMetadata(
            **event_args,
            metadata=BlockStoredMetadata(cache_salt=cache_salt),
        )

    def test_enqueue_coalesces_compatible_stores(self):
        # 入队两个父链连续的 store 事件，应合并为一个多块事件。
        queue = _KVCacheEventQueue()
        queue._enqueue_kv_event(self._store(1, None))
        queue._enqueue_kv_event(self._store(2, 1))

        events = queue.take_events()
        self.assertEqual(len(events), 1)
        self.assertEqual(events[0].block_hashes, [1, 2])
        self.assertEqual(events[0].parent_block_hash, None)
        self.assertEqual(events[0].token_ids, [1, 2, 2, 3])

    def test_enqueue_preserves_fusion_boundaries(self):
        # 介质不同、LoRA 不同、块大小不同、父链断开、cache_salt 不同都必须保持独立。
        incompatible_stores = [
            self._store(2, 1, medium=StorageMedium.CPU),
            self._store(3, 1, lora_id=1),
            self._store(4, 1, block_size=1),
            self._store(5, None),
            self._store(2, 1, cache_salt="tenant-a"),
        ]
        for incoming in incompatible_stores:
            queue = _KVCacheEventQueue()
            queue._enqueue_kv_event(self._store(1, None))
            queue._enqueue_kv_event(incoming)
            self.assertEqual(len(queue.take_events()), 2)
```

评论区精华

PR 无正式 review 评论，10 条评论均为 CI 重跑指令。最后一个 commit 'fix(kv-events): preserve salted coalescing boundaries' 修复了一个关键边界：最初实现未区分 cache_salt，导致不同租户（tenant）的 BlockStored 事件被错误合并。修复后要求 metadata 完全一致才合并，并在 test_enqueue_preserves_fusion_boundaries 中覆盖了该场景。

- cache_salt 合并边界修复 (correctness): 在合并条件中加入 tail_metadata == event_metadata 比较，并新增 test_enqueue_preserves_fusion_boundaries 覆盖该场景。

风险与影响

- 风险：

1. 核心路径变更：KVCacheEventMixin 被所有 Radix 类缓存复用，合并逻辑一旦出错会影响所有启用 KV 事件的场景。
2. 下游消费者兼容性：合并后单个事件携带多个 block，下游消费者（如 dynamo）需处理多块语义，若未同步更新可能导致块信息丢失。
3. 合并边界复杂：条件较多（medium、lora_id、block_size、metadata、父链连续性），存在漏判或误判风险，测试覆盖了主要边界但仍有潜在分支（如跨节点拆分）。

- 影响：
  1. 性能收益：事件数量从每页一个变为每连续链一个，显著降低事件分发与序列化开销。
  2. 影响范围：所有启用 KV 缓存事件的路由场景（disaggregation、KV-aware 路由）均受益。
  3. 团队维护：测试改动较多，后续需要保持合并边界测试与实现的一致性。 - 风险标记：核心路径变更，下游消费者兼容性，合并边界复杂

关联脉络

- 暂无明显关联 PR