PR #30827 完整报告

sgl-project/sglang

feat: add cache salt support to KV cache events

合并时间：2026-08-13 07:14

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30827

执行摘要

- 一句话：为 KV 事件与 radix 缓存新增 cache_salt 命名空间隔离
- 推荐动作：值得精读，尤其关注三点设计决策：(1) wire 兼容策略——用同 tag 独立 struct 追加第 8 槽位而非给基类加可选字段，是 msgspec 生态中可复用的模式；(2) 加盐哈希链的迭代化 + 节点记忆化 + 分裂切分，规避了递归深度与 $O(T^2)$ 重算；(3) fail-loudly 设计——直接属性访问与 C++ 后端显式拒绝。对下游路由端，需先实测旧 7 槽位解码器对 8 槽位事件的兼容行为再升级。

功能与动机

PR body 明确指出：外部 KV-aware 路由器需要请求级缓存命名空间，且该命名空间必须与 SGLang 调用方自定义的 extra_key 保持区分，因为 Concatenating the two values can collide，而现有 KV 事件哈希与 payload 无法暴露类型化缓存命名空间。cache_salt 因此被设计为与 extra_key 独立的一等字段，用于隔离进程内 radix 缓存并命名空间化外部 KV 事件。

实现拆解

1. 请求入口与归一化（io_struct.py）：GenerateReqInput 与 TokenizedGenerateReqInput 新增 cache_salt 字段；_normalize_single_inputs 对 extra_key 与 cache_salt 做标量字符串校验并把空串归一为 None；新增 _normalize_cache_salt 处理批量展开与并行采样复制；__getitem__ 切片逐项透传；同时收紧 _normalize_extra_key 对列表元素类型与空串的处理，避免两键混淆。
2. 服务入口与转发（serving_base.py / serving_chat.py / serving_completions.py / encode_receiver.py）：删除 _compute_extra_key——此前 chat 路径把 cache_salt 拼进 extra_key，正是碰撞来源；现在 chat/completions/responses 各自把 cache_salt 作为独立字段转发；encode_receiver.create_req 顺带修复了此前完全丢弃 extra_key 的行为（行为变更 bugfix 搭车）。
3. Radix 键与缓存变体（radix_cache.py / swa_radix_cache.py / mamba_radix_cache.py / unified 系列 / flexkv）：RadixKey 增加 cache_salt slot，切片保留；child_key 对加盐键返回 ((extra_key, cache_salt), plain) 结构化二元组，未加盐保持原样；各缓存变体的插入 / 匹配路径从 req.cache_salt 直接取值（评审后不再用 getattr 静默降级）；C++ 实验后端通过 _reject_cache_salt 显式拒绝加盐请求，绝不静默丢盐。
4. 事件哈希命名空间（mem_cache/utils.py / mem_cache/events.py / unified_tree_core.py）：新增 compute_node_event_hash_values：以 SHA256(b"sglang-cache-salt-v1\0" +

cache_salt) 为根种子，迭代自顶向下填充路径上缺失的 event_hash_value，节点级记忆化，分裂时复用 split_node_hash_value 切割；_record_store_event / _record_remove_event 对加盐节点改用事件哈希并写入 BlockStoredMetadata。

5. 事件 wire 格式（disaggregation/kv_events.py）：新增 BlockStoredMetadata 与 BlockStoredWithMetadata（共享 BlockStored tag，kw_only 追加第 8 槽位）；未加盐事件保持 7 槽位旧布局；KVEventBatch 的 tagged union 刻意不引入新类型，现有消费方按基类解码并忽略尾随元数据。
6. 测试与 CI 配套：新增 / 扩展 test_io_struct、test_radix_cache_unit、test_mem_cache_utils、test_kv_events（wire 兼容）、test_encode_receiver、test_serving_completions、test_radix_cache_cpp_unit 等，覆盖归一化、碰撞隔离（拼接相等但 salt 不同的反例）、跨节点分裂哈希保持、1100 层路径迭代、msgspec round-trip 与 C++ 后端拒绝；CI 重跑 radix_cache / hicache / disaggregation 组直至通过。

关键文件：

- python/sglang/srt/mem_cache/utils.py（模块 缓存工具；类别 source；类型 core-logic；符号 compute_node_event_hash_values）：新增 compute_node_event_hash_values：加盐事件哈希的唯一实现点，以 SHA256 根种子派生命名空间哈希链、迭代计算并节点记忆化，是隔离契约能否成立的核心。
- python/sglang/srt/mem_cache/radix_cache.py（模块 缓存核心；类别 source；类型 core-logic；符号 RadixKey, child_key, cache_finished_req, cache_unfinished_req）：RadixKey 增加 cache_salt 维度并改写 child_key 结构化索引，所有 Python radix 变体（含 swa/mamba/unified/flexkv）的隔离都依赖这里。
- python/sglang/srt/managers/io_struct.py（模块 请求归一；类别 source；类型 core-logic；符号 GenerateReqInput, TokenizedGenerateReqInput, _normalize_cache_salt, _normalize_single_inputs）：请求归一化层新增 cache_salt 字段与 _normalize_cache_salt，并收紧 extra_key 校验，是跨 43 文件透传的起点。
- python/sglang/srt/disaggregation/kv_events.py（模块 事件协议；类别 source；类型 data-contract；符号 BlockStoredMetadata, BlockStoredWithMetadata, KVEventBatch）：定义 BlockStoredMetadata / BlockStoredWithMetadata 与 KVEventBatch union 约束，决定外部路由器可见的 wire 契约。
- python/sglang/srt/mem_cache/events.py（模块 事件记录；类别 source；类型 core-logic；符号 _record_store_event, _record_remove_event）：_record_store_event / _record_remove_event 在加盐时选用事件哈希并附加 metadata，是事件负载实际拼接点。
- python/sglang/srt/mem_cache/unified_cache/unified_tree_core.py（模块 统一缓存；类别 source；类型 core-logic；符号 UnifiedTreeNode, prefetch_anchor_info, _split_node）：UnifiedTreeNode 增加 event_hash_value 并在分裂时同步切分，prefetch_anchor_info 返回值扩展为 (extra_key, cache_salt)。
- python/sglang/srt/mem_cache/radix_cache_cpp.py（模块 缓存后端；类别 source；类型 core-logic；符号 _reject_cache_salt）：通过 _reject_cache_salt 显式拒绝加盐请求，防止实验 C++ 后端静默丢盐导致跨命名空间共享缓存。

- python/sglang/srt/entrypoints/openai/serving_base.py（模块 服务入口；类别 source；类型 refactor；符号 _compute_extra_key）：删除 _compute_extra_key，消除 cache_salt 与 extra_key 拼接碰撞的源头。
- python/sglang/srt/managers/schedule_policy.py（模块 调度策略；类别 source；类型 core-logic；符号 match_prefix_for_req）：调度器匹配路径透传 req.cache_salt，并把 getattr 静默降级改为直接属性访问。
- 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 test_cache_salt_is_preserved_by_slicing, test_cache_salt_isolation_is_independent_of_extra_key, test_cache_salt_is_included_in_store_and_remove_events, test_cache_salt_event_hashes_are_preserved_across_node_split）：覆盖 salt 的切片保留、与 extra_key 拼接碰撞的反例隔离、store/remove 事件携带 metadata、跨节点分裂哈希保持，是本 PR 核心行为的回归防线。
- test/registered/unit/mem_cache/test_mem_cache_utils.py（模块 缓存测试；类别 test；类型 test-coverage；符号 test_cache_salt_seeds_root_hash_chain, test_cache_salt_event_hashes_are_memoized, test_cache_salt_event_hash_walk_is_iterative）：验证加盐根种子、记忆化与 1100 层路径的迭代 walk，守护哈希函数实现细节。
- test/registered/unit/disaggregation/test_kv_events.py（模块 事件测试；类别 test；类型 test-coverage；符号 TestBlockStoredWireFormat, test_unsalted_event_keeps_legacy_array_shape, test_salted_event_appends_typed_metadata, test_salted_event_remains_compatible_with_typed_batch_consumers）：msgspec wire 兼容性测试：未加盐保持 7 槽位、加盐追加第 8 槽位、typed batch 消费方 round-trip。
- test/registered/unit/disaggregation/test_encode_receiver.py（模块 编解码测试；类别 test；类型 test-coverage；符号 TestEncodeReceiverRequestConstruction, test_extra_key_and_cache_salt_are_forwarded）：新增测试确认 encode 路径同时转发 extra_key 与 cache_salt，覆盖捎带 bugfix。

关键符号：compute_node_event_hash_values, _normalize_cache_salt, RadixKey.init, RadixKey.child_key, _record_store_event, _record_remove_event, _reject_cache_salt, prefetch_anchor_info, _compute_extra_key, match_prefix_for_req

关键源码片段

python/sglang/srt/mem_cache/utils.py

新增 compute_node_event_hash_values：加盐事件哈希的唯一实现点，以 SHA256 根种子派生命名空间哈希链、迭代计算并节点记忆化，是隔离契约能否成立的核心。

```
# python/sglang/srt/mem_cache/utils.py

def compute_node_event_hash_values(node: Any, page_size: int) -> List[str]:
    """计算并缓存面向外部 KV 事件的命名空间哈希。
```

```
内部缓存哈希（compute_node_hash_values）只覆盖 token 序列，
不同 cache_salt 的请求会得到相同的内部哈希，因此这里用
cache_salt 派生独立根种子，再沿 radix 树做增量哈希链。
"""
cache_salt = node.key.cache_salt
# 未加盐请求直接复用内部哈希，保持旧行为完全不变
if cache_salt is None:
    return compute_node_hash_values(node, page_size)

# 已算过则直接返回：否则 chunked prefill 每个 chunk 都会
# 重算已插入前缀，长请求退化为 O(T^2 / chunk_size)
if node.event_hash_value is not None:
    return node.event_hash_value

# 迭代向上收集尚未填充 event_hash_value 的祖先节点。
# 递归版本在 page_size=1、共享前缀很深时会超过 1000 帧
# 限制触发 RecursionError，必须用显式栈
missing_nodes = []
current = node
while (
    current is not None
    and current.key is not None
    and len(current.key) > 0
    and current.event_hash_value is None
):
    # 同一条路径上混用不同 cache_salt 会使哈希链失去意义，宁可报错
    if current.key.cache_salt != cache_salt:
        raise ValueError("Radix path contains mismatched cache_salt values")
    missing_nodes.append(current)
    current = current.parent

if (
    current is not None
    and current.key is not None
    and len(current.key) > 0
    and current.key.cache_salt != cache_salt
):
    raise ValueError("Radix path contains mismatched cache_salt values")

# 根种子：SHA256("sglang-cache-salt-v1\0" + cache_salt)，
# 保证不同命名空间产生完全独立的哈希链
if current is not None and current.event_hash_value:
    parent_hash = current.event_hash_value[-1]
else:
    parent_hash = hashlib.sha256(
        b"sglang-cache-salt-v1\0" + cache_salt.encode("utf-8")
    ).hexdigest()

# 自顶向下逐节点计算并记忆化；节点分裂时直接复用
```

```
    # split_node_hash_value 按页切分，与内部 hash_value 同构
    for missing_node in reversed(missing_nodes):
        hash_values = get_hash_str(missing_node.key, parent_hash, page_size=page_size)
        assert isinstance(hash_values, list)
        missing_node.event_hash_value = hash_values
        if hash_values:
            parent_hash = hash_values[-1]

    assert node.event_hash_value is not None
    return node.event_hash_value
```

python/sglang/srt/mem_cache/radix_cache.py

RadixKey 增加 cache_salt 维度并改写 child_key 结构化索引，所有 Python radix 变体（含 swa/mamba/unified/flexkv）的隔离都依赖这里。

```
# python/sglang/srt/mem_cache/radix_cache.py

class RadixKey:
    __slots__ = ("token_ids", "extra_key", "cache_salt", "is_bigram", "limit")

    def __init__(
        self,
        token_ids: array[int],
        extra_key: Optional[str] = None,
        is_bigram: bool = False,
        limit: Optional[int] = None,
        cache_salt: Optional[str] = None,
    ):
        self.token_ids = token_ids
        # extra_key：调用方自定义的请求分类键（如 lora_id）
        self.extra_key = extra_key
        # cache_salt：缓存命名空间，必须与 extra_key 独立存放，
        # 否则拼接会产生碰撞；同时用于进程内 radix 树与外部
        # KV 事件的命名空间化，远端 L3 存储键不在此契约内
        self.cache_salt = cache_salt or None
        self.is_bigram = is_bigram
        self.limit = limit

    def child_key(self, page_size: int = 1):
        # ...（bigram / 单 token 页的 key 抽取逻辑略）

        # 加盐后按结构化二元组 (extra_key, cache_salt) 索引，
        # 未加盐请求保持原有索引行为，向后兼容
        if self.cache_salt is not None:
            return ((self.extra_key, self.cache_salt), plain)
        return plain if self.extra_key is None else (self.extra_key, plain)
```

python/sglang/srt/disaggregation/kv_events.py

定义 BlockStoredMetadata / BlockStoredWithMetadata 与 KVEventBatch union 约束，决定外部路由器可见的 wire 契约。

```
# python/sglang/srt/disaggregation/kv_events.py

class BlockStoredMetadata(msgspec.Struct, omit_defaults=True, gc=False):
    """附加在加盐 BlockStored 上的类型化请求元数据。"""

    cache_salt: str


class BlockStored(KVCacheEvent):
    # 7 槽位旧布局：未加盐事件必须保持原样，外部消费者按
    # 位置解析 block_hashes / parent_block_hash / token_ids 等
    block_hashes: list[int]
    parent_block_hash: Optional[int] = None
    token_ids: list[int]
    block_size: int
    lora_id: Optional[int] = None
    medium: Optional[str] = None


class BlockStoredWithMetadata(BlockStored, tag="BlockStored", kw_only=True):
    """仅在存在类型化元数据时使用的 wire 扩展。

    独立 struct 而非给 BlockStored 加可选字段，是为了让未加盐
    事件序列化后仍为 7 个元素；若加可选字段，msgspec 会为
    未加盐事件写出尾随 null，破坏旧消费者。
    """

    metadata: BlockStoredMetadata


class KVEventBatch(EventBatch):
    # BlockStoredWithMetadata 刻意不加入这个 tagged union：
    # 现有类型化消费方按共享的 "BlockStored" tag 解码为基类并
    # 跳过尾随元素；若把两个类型都放进 union，msgspec 会因
    # 重复 tag 直接拒绝定义。将来想"修复"时必须先读这条注释。
    events: list[Union[BlockStored, BlockRemoved, AllBlocksCleared]]
```

评论区精华

评审中最重要的交锋来自作者自评与 6 条 inline review：

- getattr 静默降级是隔离特性最糟的失败模式：Req 始终定义 cache_salt，应直接 req.cache_salt；getattr(req, "cache_salt", None) 会把"代码路径忘记携带 salt"变成"静默不加盐、跨命名空间共享缓存"，缺失属性应当大声失败。
- 事件哈希递归重算存在性能与栈溢出双重风险：chunked prefill 每个 chunk 的 store 事件都会把 root→node 整条路径重算一遍（约 $O(T^2/\text{chunk\_size})$）；page_size=1 且共享前缀

很深时递归深度可超过 Python 默认 1000 帧限制触发 RecursionError。建议迭代化并在节点上记忆化。

- store 事件中 parent 链被重复计算：_record_store_event 对 parent 的哈希链在同一个事件里算了两遍，记忆化后自然消除。
- 删除 _compute_extra_key 丢失 eager 校验：list 形式的 cache_salt 会流进声明为 Optional[str] 的 TokenizedGenerateReqInput，在 msgspec IPC decode 深处报晦涩错误而非干净 400，需要在单请求入口补标量校验。
- C++ 实验后端静默丢盐：radix_cache_cpp.py 仍构造 RadixKey(token_ids, req.extra_key)，开启 SGLANG_EXPERIMENTAL_CPP_RADIX_TREE=1 时不同 salt 相同 token 的请求会共享缓存；最终选择显式拒绝而非透传。
- KVEventBatch union 需要注释：BlockStoredWithMetadata 刻意不进 union，否则 msgspec 因重复 tag 拒绝定义；已加注释防未来"顺手修复"。

以上意见全部落地；唯一保留的开放项是 L3/ 远端存储键不按 cache_salt 命名空间的既有限制。

- cache_salt 取值应 fail loudly 而非 getattr 静默降级 (correctness): 改为直接访问 req.cache_salt，缺失属性时抛 AttributeError；最终所有缓存变体均直接透传 cache_salt。
- 事件哈希递归重算：性能与 RecursionError 风险 (performance): 重写为迭代自顶向下 walk + 节点级 event_hash_value 记忆化；分裂时用 split_node_hash_value 同步切分，测试覆盖 1100 层路径与 memoize。
- store 事件中 parent 链被重复计算 (performance): 随节点记忆化落地后重复计算被消除。
- chat 路径删除 _compute_extra_key 后丢失 eager 类型校验 (correctness): io_struct._normalize_single_inputs 对 extra_key / cache_salt 增加 str 类型校验并把空串归一为 None；新增 test_cache_salt_normalization 覆盖。
- C++ 实验 radix 后端静默丢弃 cache_salt (correctness): 最终选择显式拒绝：radix_cache_cpp.py 新增 _reject_cache_salt，启用实验 C++ 后端时对加盐请求抛错，配套 test_cache_salt_is_rejected_without_loading_cpp_extension。
- KVEventBatch union 刻意排除 BlockStoredWithMetadata 需要注释 (documentation): 已添加说明注释。
- encode 路径捎带修复 extra_key 丢失 (correctness): 已补充 extra_key / cache_salt 双转发与新增 test_encode_receiver.py 单测。
- L3/ 远端存储未按 cache_salt 命名空间（既有限制） (design): 作为既有限制记录在案，未在本 PR 内修复；PR body 明确 cache-salt-plus-LoRA 及远端语义不在范围内。

风险与影响

- 风险：
  - 兼容性风险（wire 格式）：加盐事件第 8 槽位对按 7 槽位解码的旧版外部路由器构成格式变更。测试覆盖了"8 槽位可由类型化消费方 round-trip"与"未加盐保持 7 槽位"，但未覆盖"旧版 7 槽位解码器遇到 8 槽位"的行为，msgspec 对定长数组可能直接报错，升级需与下游路由端协同。

    - 性能风险：加盐请求且开启 KV 事件时，首次访问需沿根路径填充整条 *event_hash_value* 链（已记忆化且仅一次），长前缀请求首次开销仍存在；内部缓存哈希 *get_native_hash* 路径不受影响。
    - 回归风险：*child_key* 返回结构变化影响全部 radix 变体；*_normalize_extra_key* 行为微调（空串转 None、元素类型校验）可能影响依赖旧行为的调用方；*_compute_extra_key* 删除改变了 *chat* 路径对非法类型的校验方式（已由 _normalize_single_inputs 补齐）。
    - 功能边界：SGLANG_EXPERIMENTAL_CPP_RADIX_TREE=1 与 *cache_salt* 同时使用时直接报错（显式拒绝而非降级）；远端 L3/*hicache* 存储键未加盐，加盐请求在设备 / 主机树 miss 后理论上仍可能预取其他命名空间的远端块（既有限制）。
- 影响：
    - 外部路由端：KV-aware 路由器首次能从 BlockStored 事件拿到类型化 *cache_salt* 元数据，可按命名空间做定向清理与路由，不再依赖危险的手工拼接。
    - 用户侧：Engine.*generate* / OpenAI 兼容接口新增可选 *cache_salt* 参数，纯增量；未加盐请求完全走旧路径，无行为变化。
    - 系统侧：radix 树隔离粒度提升，hash 链按命名空间独立；但 KV 事件新增第 8 槽位属于对下游消费方的格式演进，需要协同升级。
    - 团队侧：43 个文件的跨模块改动 + 大量单测，为后续 *cache*-salt-plus-LoRA 路由语义预留了正式契约。
    - 风险标记：核心缓存路径变更，事件 *wire* 格式扩展，跨模块参数透传（43 文件），C++ 后端显式拒绝加盐，哈希链首次计算开销，远端 L3 键未加盐，旧消费方 8 槽位兼容未实测

关联脉络

- PR #34607 Add bit-*exact* unified radix *cache* KL test for hybrid SWA + mamba: 同一 radix *cache* 子系统：本 PR 改动 RadixKey 索引结构与节点分裂时 *event_hash_value* 的切分，34607 的 bit-*exact* KL 回归测试正是守护这类哈希链一致性的用例。
- PR #34141 *Reserve* multimodal runtime *allocations* and *keep* padded inputs *aligned*: 同属 KV 缓存运行时链路（kv_*cache*_configurator / kv_pool_runtime）的预算与键契约演进，说明 radix 与缓存事件侧契约正在逐步显式化。
- PR #34335 metrics: don't *clock*-*rebase* unset time *sentinels* in ReqTimeStats *deserialization*: 同属 *disaggregation* 元数据链路的 *wire* 兼容性维护，呼应本 PR 对 KV 事件 msgspec 布局的谨慎处理。