PR #23300 完整报告

sgl-project/sglang

[bug] Fix cache salt and extra keys for prefix cache isolation

合并时间：2026-04-22 04:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23300

执行摘要

- 一句话：修复 cache_salt 隔离失效的 extra_key 传递 bug
- 推荐动作：该 PR 是典型的数据传递断裂 bug，代码改动量小但影响面大——直接破坏安全特性。推荐精读以理解 SGLang 请求处理链中 Req 构造的关键节点，建议未来类似 feature 在 PR 中增加端到端集成测试。

功能与动机

Issue #9163 提出的 cache_salt 安全隔离功能部署后未生效：不同 cache_salt 的请求本应使用不同的前缀缓存 key，但由于 extra_key 字段在请求链路（从请求入口到 Scheduler 构造 Req 对象）中断裂，所有请求实际使用相同的缓存 key，使得不同 tenant 之间的缓存隔离彻底失效。这直接违背了 feature 的设计目标——防止多租户场景下前缀缓存泄露隐私（参考《Leaking Secrets from Prefix Caches》论文提出的威胁模型）。

实现拆解

1. 在 Scheduler 入口补传 extra_key：在 python/sglang/srt/managers/scheduler.py 的 handle_generate_request 方法中，在构造 Req 对象时新增 extra_key=recv_req.extra_key 参数。该参数原本已存在于 recv_req 对象中，但由于未显式传递给 Req 构造函数，导致字段静默丢失。这是核心修复。
2. 在 Session 路径同时补传 extra_key：在 python/sglang/srt/session/session_controller.py 的 create_req 方法中，同样新增 extra_key=req.extra_key 字段。虽然 Session 场景使用 cache_salt 的场景较少，但保持入口一致性，避免未来出现类似的隔离失效问题。
3. 增强测试能力：在 test/manual/openai_server/features/test_cache_report.py 中新增 _get_cached_tokens 静态方法将 prompt_tokens_details 的 None 检查封装为 0 值兜底，减少重复代码。同时调整 test_cache_salt_effectiveness 中的断言逻辑：将原来判定 cached_tokens 完全不变的 assert 改为 cached_tokens_2_second > cached_tokens_2_first，即验证不同 salt 隔离下相同 salt 仍能命中（cached_tokens 增长），使得测试逻辑更贴近实际缓存行为。还新增了 --attention-backend=triton 的启动参数，确保测试环境一致。

关键文件：

- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 handle_generate_request）：核心修复点：在 handle_generate_request 中补传

extra_key 到 Req 构造函数。

- python/sglang/srt/session/session_controller.py（模块 会话管理；类别 source；类型 entrypoint；符号 create_req）：同步修复 Session 路径中 Req 构造缺失 extra_key 的问题。
- test/manual/openai_server/features/test_cache_report.py（模块 缓存报告；类别 test；类型 test-coverage；符号 _get_cached_tokens, test_cache_salt_effectiveness）：测试用例增强：提取 _ get _cached _tokens 辅助方法、调整断言逻辑并增加运行参数。

关键符号：handle_generate_request, create_req, _get_cached_tokens, test_cache_salt_effectiveness

关键源码片段

python/sglang/srt/managers/scheduler.py

核心修复点：在 handle_generate_request 中补传 extra_key 到 Req 构造函数。

```
# python/sglang/srt/managers/scheduler.py 中的 handle_generate_request 方法（约第 1878 行）
# 修复前：extra_key 字段未从 recv_req 传递到 Req，导致 cache_salt 隔离失效
# 修复后：显式传递 extra_key，使前缀缓存 key 能区分不同 cache_salt

# 构造 Req 对象时补传 extra_key
req = Req(
    rid=recv_req.rid,
    origin_input_text=recv_req.origin_input_text,
    origin_input_ids=recv_req.origin_input_ids,
    origin_input_ids_unpadded=recv_req.origin_input_ids_unpadded,
    sampling_params=sampling_params,
    lora_id=recv_req.lora_id,
    session=recv_req.session,
    custom_logit_processor=recv_req.custom_logit_processor,
    stream=recv_req.stream,
    return_logprob=recv_req.return_logprob,
    top_logprobs_num=recv_req.top_logprobs_num,
    token_ids_logprob=recv_req.token_ids_logprob,
    vocab_size=self.model_config.vocab_size,
    eos_token_ids=self.model_config.eos_token_ids,
    require_reasoning=recv_req.require_reasoning,
    return_hidden_states=recv_req.return_hidden_states,
    return_routed_experts=recv_req.return_routed_experts,
    priority=recv_req.priority,
    routing_key=recv_req.routing_key,
    extra_key=recv_req.extra_key, # 修复：补传该字段
    http_worker_ipc=recv_req.http_worker_ipc,
    dllm_config=self.dllm_config,
    time_stats=recv_req.time_stats,
)
```

python/sglang/srt/session/session_controller.py

同步修复 Session 路径中 Req 构造缺失 extra_key 的问题。

```
# python/sglang/srt/session/session_controller.py 中的 create_req 方法（约第 235 行）
# 修复前：extra_key 未传递，Session 中 Req 的缓存隔离同样失效
# 修复后：与调度器路径保持一致

new_req = Req(
    rid=req.rid,
    origin_input_text=None,
    origin_input_ids=input_ids,
    origin_input_ids_unpadded=input_ids_unpadded,
    sampling_params=req.sampling_params,
    lora_id=req.lora_id,
    session=self,
    custom_logit_processor=req.custom_logit_processor,
    stream=req.stream,
    return_logprob=req.return_logprob,
    top_logprobs_num=req.top_logprobs_num,
    token_ids_logprob=req.token_ids_logprob,
    vocab_size=vocab_size,
    eos_token_ids=eos_token_ids,
    require_reasoning=req.require_reasoning,
    return_hidden_states=req.return_hidden_states,
    return_routed_experts=req.return_routed_experts,
    priority=req.priority,
    routing_key=req.routing_key,
    extra_key=req.extra_key, # 修复：补传该字段
    http_worker_ipc=req.http_worker_ipc,
    time_stats=req.time_stats,
)
```

评论区精华

该 PR 没有产生 Review 评论，但从 PR 描述和提交历史来看，Contributor 基于 Issue #9163 发现并定位了 extra_key 断裂问题。Reviewer hnyls2002 直接批准，说明变更正确性较为明显。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 回归风险低：修复仅涉及参数传递，未改变 Req 构造以外的逻辑。测试文件中断言条件的调整（从严格相等改为大于）反而更符合缓存系统的实际行为，降低了误判风险。
  2. Session 场景可能未被充分覆盖：虽然 session_controller.py 补充了 extra_key，但现有测试用例未对 Session 中使用 cache_salt 的场景进行验证，存在潜在遗留问题。
  3. 测试环境约束：测试新增了 --attention-backend=triton 参数，若未来有其他 attention 后端与该参数冲突，该测试可能失效。

- 影响：
  1. 用户影响：所有使用 cache_salt 的多租户部署用户将受益，缓存隔离得以正确生效；此前已部署但未生效的用户需要升级后方可获得隔离。
  2. 系统影响：无性能影响，extra_key 字段在请求对象中早已存在，仅传递路径补齐。
  3. 团队影响：提醒团队在 feature 开发时需关注长调用链中参数的完整性，建议增加集成测试覆盖跨模块参数传递。 - 风险标记：核心路径变更，缺少 Session 场景测试覆盖

关联脉络

- PR #9163 [Feature] Support Cache Salting for secure prefix caching: 本 PR 修复了该 feature 的传递断裂 bug，是功能实现的必要补丁。