# PR #29436 完整报告

sgl-project/sglang

feat: first-class session identity in SGLang

合并时间：2026-06-28 22:16

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29436

## 执行摘要

- 一句话：引入顶层 session_id 作为稳定请求标识
- 推荐动作：值得精读，尤其是 session 身份与 session_params 解耦的设计，以及 radix session 隐式启用模式。注意兼容性迁移。

## 功能与动机

PR body 指出：需要一种稳定的请求身份标识来驱动 session radix cache，而不必依赖显式的 session_params。通过顶层 session_id 简化用户交互，并且保持现有 session_params 路径不变。

## 实现拆解

1. 在 io_struct.py 的 GenerateReqInput 和 TokenizedGenerateReqInput 中添加可选 session_id 字段，并在 normalize_batch_and_arguments 中增加互斥校验。
2. 在 Rust gRPC 请求映射 request_utils.rs 的 build_text_generate_dict 和 build_generate_dict 中，当 session_id 存在时插入到请求字典。
3. 在 scheduler.py 的 handle_generate_request 中，将 radix_native_session 的判断条件从 session_params.id 改为顶层 session_id，移除了对 session_params 的校验，并将 session_id 传入 Req 对象。同时调整 open_session 和 close_session 逻辑：open_session 只转发给 session_controller（传统路径），close_session 如果启用 radix cache 则调用 release_radix_session，否则调用 session_controller.close。
4. 在 session_radix_cache.py 中移除 register_session 方法，将 release_session 重命名为 release_radix_session，并在 _tag_session_leaf 中加入 enable_session_radix_cache 守卫。
5. 在 Engine、OpenAI、HTTP Server 等 entrypoint 添加 session_id 参数传递。
6. 新增测试用例覆盖禁用缓存时不标记、传统 release 不影响 radix session、close tombstone 阻止晚期 finish 等场景。

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, open_session, close_session）：核心调度逻辑，修改 radix_native_session 判断条件并移除对 session_params 的校验，将 session_id 传入 Req。

- python/sglang/srt/mem_cache/session_radix_cache.py（模块 缓存层；类别 source；类型 core-logic；符号 _tag_session_leaf, release_radix_session, release_session, register_session）：Radix session 核心逻辑，修改了标记和释放语义。
- python/sglang/srt/managers/io_struct.py（模块 请求结构；类别 source；类型 core-logic；符号 GenerateReqInput.session_id, TokenizedGenerateReqInput.session_id）：添加 session_id 字段并增加互斥校验。
- rust/sglang-grpc/src/utils/request_utils.rs（模块 gRPC 处理；类别 source；类型 core-logic；符号 build_text_generate_dict, build_generate_dict）：Rust gRPC 请求映射加入 session_id 传递。
- test/manual/core/test_session_radix_cache.py（模块 缓存层；类别 test；类型 test-coverage；符号 test_disabled_cache_does_not_tag_session_kv, test_legacy_release_does_not_release_radix_session, test_close_tombstone_blocks_late_finish）：测试 session radix cache 新行为，包括禁用时标记、传统 release 不释放 radix session 等。

关键符号：_tag_session_leaf, release_radix_session, handle_generate_request, build_text_generate_dict, build_generate_dict

## 关键源码片段

### python/sglang/srt/managers/scheduler.py

核心调度逻辑，修改 radix_native_session 判断条件并移除对 session_params 的校验，将 session_id 传入 Req。

```
def handle_generate_request(self, recv_req: TokenizedGenerateReqInput):
    # 从 session_params.id 获取传统 session id
    session_id = (
        recv_req.session_params.id if recv_req.session_params is not None else None
    )
    # Radix-native sessions use only the top-level session_id.
    radix_native_session = (
        recv_req.session_id is not None
        and self.server_args.enable_session_radix_cache
    )

    if session_id is None or radix_native_session:
        # Normal non-session request, or a radix-native session request
        # ... (mm inputs, bootstrap port, etc.)
        req = Req(
            recv_req.rid,
            recv_req.input_text,
            recv_req.input_ids,
            recv_req.sampling_params,
            return_logprob=recv_req.return_logprob,
            top_logprobs_num=recv_req.top_logprobs_num,
            token_ids_logprob=recv_req.token_ids_logprob,
            stream=recv_req.stream,
```

```
            lora_id=recv_req.lora_id,
            session_id=recv_req.session_id, # 将顶层 session_id 传入 Req
            # ... other fields
        )
    else:
        # Legacy session request with session_params
        # ...
```

## python/sglang/srt/mem_cache/session_radix_cache.py

Radix session 核心逻辑，修改了标记和释放语义。

```
  def _tag_session_leaf(self, req: Req, radix_key, node=None) -> None:
    # 将请求的 session_id 标记到叶子节点（仅当 radix cache 启用时）
    if not self.enable_session_radix_cache:
        return # 缓存未启用，不执行任何操作
    self._ensure_session_radix_state()
    sid = getattr(req, 'session_id', None)
    if sid is None or sid in self._closed_session_ids:
        return
    if node is None:
        logger.warning(
            '_tag_session_leaf 被调用时未提供 node，将回退到 match_prefix'
        )
        node = self.match_prefix(MatchPrefixParams(key=radix_key)).last_device_node
    if node is not None and node is not self.root_node:
        session_ids = getattr(node, 'session_ids', None)
        if session_ids is None:
            session_ids = set()
            node.session_ids = session_ids
        session_ids.add(sid)
        self._session_leaves[sid].add(node)

  def release_radix_session(self, session_id: str) -> int:
    # 关闭 session：从每个标记的叶子节点中移除该 session 的持有关系
    # 仅当节点不再被任何 session 持有且满足回收条件时才释放物理 KV 块
    self._ensure_session_radix_state()
    self._remember_closed_session(session_id)
    indexed = self._session_leaves.pop(session_id, set())
    freed = 0
    for leaf in indexed:
        if session_id not in getattr(leaf, 'session_ids', set()):
            continue
        node = leaf
        while True:
            session_ids = getattr(node, 'session_ids', None)
            if session_ids is not None:
                session_ids.discard(session_id)
                if not session_ids:
                    delattr(node, 'session_ids')
```

```
            # 停止条件：根节点、锁定的节点、有子节点的节点、不可逐出的节点、或被其他 session
            持用的节点
            if (
                node is self.root_node
                or node.lock_ref != 0
                or len(node.children) != 0
                or node not in self.evictable_leaves
                or getattr(node, 'session_ids', None)
            ):
                break
            parent = node.parent
            self.token_to_kv_pool_allocator.free(node.value)
            self._delete_leaf(node)
            freed += 1
            node = parent
    logger.info('released radix session %s: %d tokens freed', session_id, freed)
    return freed
```

## 评论区精华

无实质 review 讨论，作者自行合并。CI 全部通过。

- 整体评审 (other): 无

## 风险与影响

- 风险：
  1. 兼容性风险：新增 session_id 与 session_params 互斥约束，可能中断同时使用两者的旧请求。
  2. 行为变化：启用 enable_session_radix_cache 后，携带 session_id 的请求会自动开启 radix session，不再需要 /open_session，可能影响依赖显式打开的客户端。
  3. 废弃 register_session 方法：任何外部调用会报错，需迁移。 - 影响：影响范围中等：修改了请求定义（io_struct）、调度器核心逻辑、gRPC 映射、多个 entrypoint 和测试。对使用 session radix cache 的用户体验简化明显；对仅用传统 session 的用户无影响。 - 风险标记：session_id 与 session_params 互斥，隐式开启 radix session，移除 register_session 方法

## 关联脉络

- 暂无明显关联 PR