PR #36638 完整报告

sgl-project/sglang

Fix KeyError on batch requests whose state is freed before it is read

合并时间：2026-08-29 03:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36638

执行摘要

- 一句话：修复批处理请求状态提前释放导致的 KeyError，改为预解析状态
- 推荐动作：建议精读，该 PR 展示了异步生成器生命周期与状态管理的经典陷阱，值得学习。设计决策：将状态解析提前到生成器构建时，避免状态失效。

功能与动机

批处理请求中，若某个请求在迭代前完成，其状态会被 _handle_batch_output 删除，导致生成器推进时 KeyError，进而整个批次失败。多模态批次每次都会触发，导致夜间 test_vlms_perf.py 失败。

实现拆解

1. 将 _wait_one_response 改为普通函数：从 async def 改为 def，在函数体内立即执行 state = self.rid_to_state[obj.rid]，然后返回 self._stream_one_response(obj=obj, state=state, request=request)。这样状态查找在生成器创建时完成，而非推进时，避免状态删除后 KeyError。
2. 新增 _stream_one_response 异步生成器：将原 _wait_one_response 中的循环逻辑移到新函数，接受已解析的 state 参数，不再依赖 rid_to_state。
3. 保持调用点不变：所有五处调用 _wait_one_response 的地方无需修改，兼容性良好。
4. 添加回归测试：新增 TestWaitOneResponseAfterStateFreed，模拟先构建 waiter 再释放状态的场景，验证输出仍能传递。测试使用 _handle_batch_output 触发状态删除，并断言输出内容正确。

关键文件：

- python/sglang/srt/managers/tokenizer_manager.py（模块 调度器；类别 source；类型 core-logic；符号 _wait_one_response, _stream_one_response）：核心修复文件，修改 _wait_one_response 和新增 _stream_one_response，解决状态提前释放导致的 KeyError。
- test/registered/unit/managers/test_tokenizer_manager_rid_cleanup.py（模块 测试；类别 test；类型 test-coverage；符号 TestWaitOneResponseAfterStateFreed, test_generator_built_before_finish_still_delivers_output, drive）：新增回归测试，验证状态提前释放后生成器仍能传递输出，防止回归。

关键符号：_wait_one_response, _stream_one_response, test_generator_built_before_finish_still_delivers_output

关键源码片段

python/sglang/srt/managers/tokenizer_manager.py

核心修复文件，修改 _wait_one_response 和新增 _stream_one_response，解决状态提前释放导致的 KeyError。

```
# python/sglang/srt/managers/tokenizer_manager.py

def _wait_one_response(
    self,
    obj: Union[GenerateReqInput, EmbeddingReqInput],
    request: Optional[fastapi.Request] = None,
):
    # Batch dispatch builds every waiter before advancing any.
    # Both removers append the output after the del, so the ReqState stays valid.
    # 关键：状态查找在生成器创建时完成，避免生成器推进时状态已删除导致 KeyError。
    state = self.rid_to_state[obj.rid]
    return self._stream_one_response(obj=obj, state=state, request=request)

async def _stream_one_response(
    self,
    obj: Union[GenerateReqInput, EmbeddingReqInput],
    state: ReqState,
    request: Optional[fastapi.Request] = None,
):
    # 原 _wait_one_response 的异步逻辑移到这里，使用已解析的 state。
    is_stream = getattr(obj, "stream", False)
    while True:
        try:
            await asyncio.wait_for(
                state.event.wait(), timeout=_REQUEST_STATE_WAIT_TIMEOUT
            )
        except asyncio.TimeoutError:
            if (
                request is not None
                and not obj.background
                and await request.is_disconnected()
            ):
                # 处理客户端断连，中止请求。
                self.abort_request(obj.rid)
                raise ValueError(
                    f"Request is disconnected from the client side (type 1). Abort request {obj.rid=}"
                )
            continue

        # 原子地清空待处理输出。
```

```
        out_list = state.out_list
        state.out_list = []
        finished = state.finished
        state.event.clear()

        # 增量流式时合并多个分块，避免丢失 token 编号。
        incremental_stream = is_stream and self.incremental_streaming_output
        if incremental_stream and len(out_list) > 1:
            # ... 合并逻辑 ...
            pass
        # ... 其余处理 ...
```

test/registered/unit/managers/test_tokenizer_manager_rid_cleanup.py

新增回归测试，验证状态提前释放后生成器仍能传递输出，防止回归。

```
# test/registered/unit/managers/test_tokenizer_manager_rid_cleanup.py

class TestWaitOneResponseAfterStateFreed(CustomTestCase):
    """状态在读取前释放时，waiter 仍须传递输出。

    批处理派发会先构建所有 waiter，再统一推进；
    调度器响应路径在请求完成时立即移除 rid_to_state。
    """

    def test_generator_built_before_finish_still_delivers_output(self):
        tm = _make_tokenizer_manager(self)
        tm.request_logger = Mock()
        tm.request_metrics_exporter_manager = MagicMock()
        tm.request_metrics_exporter_manager.exporter_enabled.return_value = False
        rid = "freed_state_rid"
        state = _make_req_state(rid)
        state.obj.background = True # 跳过 fastapi 断连探测
        tm.rid_to_state[rid] = state

        async def drive():
            # 关键：先构建生成器，再触发状态删除。
            waiter = tm._wait_one_response(state.obj, None)
            await tm._handle_batch_output(_make_batch_str_output(rid))
            self.assertNotIn(rid, tm.rid_to_state)
            return await waiter.__anext__()

        out = asyncio.run(drive())
        # 断言输出仍能正确传递。
        self.assertEqual(out["meta_info"]["id"], rid)
        self.assertEqual(out["text"], "hello")
```

评论区精华

无

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。_wait_one_response 签名未变，调用点不变，行为差异仅在状态查找时机。可能的风险：如果其他代码依赖生成器创建时不执行状态查找（例如延迟错误抛出），则可能改变行为，但当前代码库中无此类依赖。测试新增覆盖了状态释放场景，但未涉及并发或多请求并发场景，建议后续补充。
- 影响：对用户而言，修复了多模态批处理请求的偶发 KeyError，提高了稳定性。对系统而言，改动核心路径但影响面小，所有调用点兼容。对团队而言，增强了测试覆盖，有助于防止回归。
- 风险标记：核心路径变更，测试覆盖薄弱（仅单测，缺并发场景）

关联脉络

- PR #36738 [HiCache] Fence load-back behind the forward stream: 两者都涉及调度器状态管理，但关联较弱。