PR #22055 完整报告

sgl-project/sglang

[HiCache] return cached_tokens_details in sglext for streaming responses

合并时间：2026-05-05 03:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22055

执行摘要

- 一句话：修复流式响应中 cached_tokens_details 在 sglext 中缺失的问题
- 推荐动作：值得精读，因为展示了如何修复流式响应中字段缺失的常见模式，以及如何重构共享逻辑。设计决策：将辅助函数提取到 utils.py 以便复用，以及将 routed_experts 和 cached_tokens_details 合并到一个 sglext 块中。

功能与动机

PR body 指出：sglext.cached_tokens_details is returned correctly in non-streaming chat/completions responses, but silently dropped in streaming mode。后端已在每个请求的 meta_info 中填充 cached_tokens_details，流式循环也已收集该字段但从未提取并发出。本 PR 修复了该不一致。

实现拆解

1. 提取辅助函数：在 python/sglang/srt/entrypoints/openai/utils.py 中新增 cached_tokens_details_from_dict 函数，将原始字典转换为 CachedTokensDetails 对象，并重构 process_cached_tokens_details_from_ret 以复用该函数，消除代码重复。
2. 聊天流式服务变更：在 python/sglang/srt/entrypoints/openai/serving_chat.py 的 _generate_chat_stream 方法中，新增 cached_tokens_details 字典跟踪每个 index 的详情。流结束时，将 routed_experts 和 cached_tokens_details 合并到一个统一的 SglExt 块中发出。
3. 补全流式服务变更：在 python/sglang/srt/entrypoints/openai/serving_completions.py 的 _generate_completion_stream 方法中做完全相同变更，保持两个服务一致性。
4. 单元测试：在 test_serving_chat.py 和 test_serving_completions.py 各添加两个测试用例：test_non_streaming_cached_tokens_details_emits_sglext 和 test_streaming_cached_tokens_details_emits_sglext，使用 mock 验证 sglext.cached_tokens_details 在非流式和流式响应中都被正确填充。

关键文件：

- python/sglang/srt/entrypoints/openai/serving_chat.py（模块 服务层；类别 source；类型 core-logic；符号 _generate_chat_stream）：核心变更：在流式聊天响应中收集并发出 cached_tokens_details 到 sglext 块

- python/sglang/srt/entrypoints/openai/serving_completions.py（模块 服务层；类别 source；类型 core-logic；符号 _generate_completion_stream）：类似 serving_chat.py，在 completion 流式响应中收集并发出 cached_tokens_details
- python/sglang/srt/entrypoints/openai/utils.py（模块 工具层；类别 source；类型 core-logic；符号 cached_tokens_details_from_dict）：提取 cached_tokens_details_from_dict 辅助函数，减少冗余，被两个 serving 模块共用
- test/registered/unit/entrypoints/openai/test_serving_chat.py（模块 测试 - 聊天；类别 test；类型 test-coverage；符号 test_non_streaming_cached_tokens_details_emits_sglext, test_streaming_cached_tokens_details_emits_sglext, _mock_generate_with_cached_tokens_details, run_stream）：新增测试覆盖流式和非流式场景下 cached_tokens_details 在 sglext 中的输出
- test/registered/unit/entrypoints/openai/test_serving_completions.py（模块 测试 - 补全；类别 test；类型 test-coverage；符号 test_non_streaming_cached_tokens_details_emits_sglext, test_streaming_cached_tokens_details_emits_sglext, _mock_generate_with_cached_tokens_details, run_stream）：类似测试，覆盖 completion 端点

关键符号：cached_tokens_details_from_dict, _generate_chat_stream, _generate_completion_stream

关键源码片段

python/sglang/srt/entrypoints/openai/serving_chat.py

核心变更：在流式聊天响应中收集并发出 cached_tokens_details 到 sglext 块

```
# _generate_chat_stream 方法的结尾部分（流循环后）

# 收集 routed_experts 和 cached_tokens_details 的首个非 None 值
sglext_routed = None
if request.return_routed_experts and routed_experts:
    sglext_routed = next(
        (v for v in routed_experts.values() if v is not None), None
    )

sglext_details = None
if request.return_cached_tokens_details and cached_tokens_details:
    first_details = next(
        (v for v in cached_tokens_details.values() if v is not None), None
    )
    if first_details is not None:
        sglext_details = cached_tokens_details_from_dict(first_details)

# 若任一扩展信息存在，则合并为一个 sglext 块发出
if sglext_routed is not None or sglext_details is not None:
    sglext_chunk = ChatCompletionStreamResponse(
        id=content["meta_info"]["id"],
        created=int(time.time()),
```

```
            choices=[], # sglext 位于响应级别，非 choice 级别
            model=request.model,
            sglext=SglExt(
                routed_experts=sglext_routed,
                cached_tokens_details=sglext_details,
            ),
        )
        yield f"data: {sglext_chunk.model_dump_json()}

    "
```

python/sglang/srt/entrypoints/openai/utils.py

提取 cached_tokens_details_from_dict 辅助函数，减少冗余，被两个 serving 模块共用

```
# 新增的辅助函数，将原始 dict 转换为 CachedTokensDetails 对象

def cached_tokens_details_from_dict(
    details: Dict[str, Any],
) -> CachedTokensDetails:
    # 将原始缓存命中详情字典转换为 CachedTokensDetails 对象。
    # 支持可选的 L3 storage 字段（若不存在则返回基础版本）。
    if "storage" in details:
        return CachedTokensDetails(
            device=details.get("device", 0),
            host=details.get("host", 0),
            storage=details.get("storage", 0),
            storage_backend=details.get("storage_backend"),
        )
    else:
        return CachedTokensDetails(
            device=details.get("device", 0),
            host=details.get("host", 0),
        )
```

评论区精华

Kangyan-Zhou要求添加单元测试覆盖该场景，vladnosiv回应“测试已添加并通过”。后续 CI 运行成功，测试被验证通过。

gemini-code-assist[bot]提出简化 sglext_routed 变量初始化的建议（使用条件表达式直接赋值），但 PR 作者未采纳该风格建议，代码保持原有风格合并。

- 添加单元测试 (testing): 测试已添加并合并。
- 简化 sglext_routed 初始化 (style): 未采纳，但代码已合并，风格无重大影响。

风险与影响

- 风险：流式响应新增了一个包含 sglext 的数据块，客户端如果严格解析可能受影响（例如期望每个 chunk 都有 choices）。但本实现遵循已有 hidden_states 和 routed_experts 的模

式，且发出的 chunk choices 为空数组，兼容现有流式标准。重构辅助函数可能引入回归，但测试覆盖充分。

- 影响：用户现在可以在流式响应中通过 sglext.cached_tokens_details 获取缓存命中详情，实现非流式与流式的行为一致。团队维护成本降低，因为重复逻辑被提取到单一函数中。对性能影响极小。
- 风险标记：流式兼容性，JSON 解析影响

关联脉络

- 暂无明显关联 PR