PR #35778 完整报告

sgl-project/sglang

fix(grpc): derive choice count before normalization

合并时间：2026-08-22 02:29

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35778

执行摘要

- 一句话：修复 gRPC 流式请求在归一化前取字段导致的崩溃
- 推荐动作：值得精读。虽然改动只有几行，但它揭示了异步生成器初始化与请求归一化之间的隐性时序契约，以及"循环不变量上提"这类清理操作可能引入的回归模式。对入口层、gRPC bridge 或异步流水线的维护者尤其有参考价值。

功能与动机

PR body 明确指出：原生流式 gRPC 请求在产生第一个 token 前就会失败，因为 RuntimeHandle._run_generate 在创建 TokenizerManager.generate_request 异步生成器后立即读取 GenerateReqInput.batch_size 和 parallel_sample_num，而请求归一化只在生成器首次被推进时才初始化这些字段，导致每次流式请求都抛 AttributeError。该顺序问题源自 #32588 的 n > 1 lifecycle 工作：最初实现把 choice 计数计算放在流循环内（此时归一化已执行），后续清理把该循环不变量上提到循环外，无意中移动到了归一化之前。

实现拆解

1. 定位根因：python/sglang/srt/entrypoints/grpc_bridge.py 中 RuntimeHandle._run_generate 在构建 TokenizerManager.generate_request 生成器之后、开始迭代之前计算 expected_choices = obj.batch_size * obj.parallel_sample_num。由于归一化发生在生成器首次迭代时，流式请求必然在读取派生字段时崩溃。
2. 修改计数来源：将 expected_choices 改为 max(1, int(sampling_params.get("n", 1)))，其中 sampling_params = obj.sampling_params or {}，直接从原始请求的采样参数读取，确保在归一化发生前也能得到正确的预期 choice 数。
3. 保留 n > 1 终止语义：completed_choices 集合继续按 index / meta_info.id 跟踪已结束的 choice，finished = len(completed_choices) >= expected_choices 仍保证最后一个 choice 结束后才向前端发送 finished=True 并退出循环；生成器提前退出时仍补发防御性空块。
4. 收紧桥接测试：test/registered/unit/entrypoints/test_grpc_bridge.py 中 test_streaming_first_finished_choice_is_not_batch_terminal 的 fixture 从 SimpleNamespace(rid="logical", batch_size=1, parallel_sample_num=2) 改为 SimpleNamespace(rid="logical", sampling_params={"n": 2})，使测试对象与生产一致，不携带归一化后才存在的派生字段，从而覆盖本次回归。

5. 验证与配套：单元测试 2 个通过，ruff 与 check_registered_tests.py 通过；作者在 GPU 环境验证聚合与分离式服务路径 83/83 请求成功。无文档、schema 或部署配置改动。

关键文件：

- python/sglang/srt/entrypoints/grpc_bridge.py（模块 桥接层；类别 source；类型 core-logic；符号 _run_generate）：核心修复文件：RuntimeHandle._run_generate 的流式分支原先在生成器归一化前读取 batch_size * parallel_sample_num，现改为从原始 sampling_params["n"] 读取，修复所有流式 gRPC 请求的 AttributeError。
- test/registered/unit/entrypoints/test_grpc_bridge.py（模块 入口测试；类别 test；类型 test-coverage；符号 test_streaming_first_finished_choice_is_not_batch_terminal）：测试配套：将流式用例的 fixture 从含 batch_size / parallel_sample_num 的 SimpleNamespace 改为 sampling_params={"n": 2}，与生产环境未归一化对象一致，防止回归再次漏网。

关键符号：_run_generate

关键源码片段

python/sglang/srt/entrypoints/grpc_bridge.py

核心修复文件：RuntimeHandle._run_generate 的流式分支原先在生成器归一化前读取 batch_size * parallel_sample_num，现改为从原始 sampling_params["n"] 读取，修复所有流式 gRPC 请求的 AttributeError。

```
async def _run_generate(self, obj, chunk_callback, stream: bool, request):
    ready_event = None
    gen = None
    try:
        ready_event = self._install_on_ready(chunk_callback)
        # generate_request 返回的异步生成器只有在首次迭代时才执行
        # 请求归一化（填充 batch_size、parallel_sample_num 等派生字段），
        # 因此在迭代之前读取这些字段会触发 AttributeError。
        gen = self.tokenizer_manager.generate_request(obj, request=request)

        if stream:
            completed_choices = set()
            # 流式场景下，预期的 choice 数量从原始采样参数中读取，
            # 而不是从归一化后的派生字段读取；本 PR 修复的核心就在这里。
            sampling_params = obj.sampling_params or {}
            expected_choices = max(1, int(sampling_params.get("n", 1)))

            async for chunk in gen:
                choice_finished = (
                    chunk.get("meta_info", {}).get("finish_reason") is not None
                )
                if choice_finished:
                    choice_id = chunk.get(
                        "index", chunk.get("meta_info", {}).get("id")
                    )
```

```
                completed_choices.add(choice_id)
            # 所有 choice 都结束后才向前端发送 finished=True，
            # 保持 n > 1 时"最后一个 choice 结束后流才终止"的语义。
            finished = len(completed_choices) >= expected_choices
            keep_going = await self._send_with_backpressure(
                chunk_callback,
                ready_event,
                chunk,
                finished=finished,
                timeout_abort_rid=obj.rid,
            )
            if finished or not keep_going:
                return
        # 防御性处理：生成器提前退出且没有 finish_reason 时，
        # 仍要向前端补发一个 finished=True 的空块。
        self._safe_callback(chunk_callback, {}, finished=True)
    else:
        # 非流式路径：直接取生成器首个结果，不涉及 choice 计数。
        result = await gen.__anext__()
        chunks = result if isinstance(result, list) else [result]
        for index, chunk in enumerate(chunks):
            keep_going = await self._send_with_backpressure(
                chunk_callback,
                ready_event,
                chunk,
                finished=index == len(chunks) - 1,
                timeout_abort_rid=obj.rid,
            )
```

评论区精华

本 PR 没有产生实质性的 review 技术讨论：两位 reviewer ishandhanani 和 alexnails 均直接 approve；ishandhanani 的唯一评论是触发 CI 重跑的命令 /tag-and-rerun-ci，无技术结论。PR body 中作者对背景、根因和"为什么不继续读 batch_size / parallel_sample_num"给出了完整说明，是理解本次修复的关键材料。

- 暂无高价值评论线程

风险与影响

- 风险：
  - 回归风险：修复依赖 generate_request"归一化发生在首次迭代"的时序契约。若未来重构把归一化提前到 generate_request 调用时，本修复依然安全（此时 sampling_params["n"] 与归一化字段一致）；但若未来改变 sampling_params 结构或移除 n 默认语义，需要同步调整。
  - 测试覆盖：只有单测覆盖，没有真实 gRPC 流式端到端测试；test_grpc_bridge.py 用模拟响应驱动 _run_generate，无法捕获 generate_request 本身归一化时序的变化，未来同类回归可能再次漏网。

- 兼容性：int(sampling_params.get("n", 1)) 对 n 为 0/None 的情况用 max(1, ...) 兜底为 1，与非流式路径中“默认单 choice”语义一致；当前 GenerateReqInput.sampling_params 是 dict 结构，若未来变成非 dict 类型，.get 调用需要调整。
- 影响范围：仅 gRPC 流式入口受影响，HTTP 流式与非流式路径不变；对下游 KV 事件、disaggregation、取消和 NIXL handoff 的影响是“从不可用到恢复可用”，而非行为变更。
- 影响：对用户而言，原生流式 gRPC 请求（sidecar 场景）从“每个请求 HTTP 500”恢复为正常产出 token；对系统而言，流式路径恢复后可继续发布 KV 事件、执行缓存亲和与失效、取消处理以及 NIXL handoff，聚合与分离式服务路径均受益。改动量极小（2 个文件、6 行），不影响模型计算与输出值，团队无需额外迁移成本。
- 风险标记：gRPC 流式路径回归修复，依赖生成器归一化时序，仅单测覆盖无端到端测试，默认 n=1 语义假设

关联脉络

- PR #32588 n > 1 lifecycle work（标题未在上下文中提供）：PR body 明确指出本 bug 的回归来源：#32588 的清理把 choice 计数计算上提到循环外、移动到归一化之前；本 PR 是对该回归的直接修复。