PR #25298 完整报告

sgl-project/sglang

Fix bench_serving non-stream reasoning content

合并时间：2026-05-21 02:41

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25298

执行摘要

- 一句话：修复 bench_serving 非流式推理模型内容为空导致崩溃
- 推荐动作：本 PR 改动虽小但修复明确、测试充分，建议快速合并。值得关注的设计决策是提炼共享函数而非在流式和 / 或非流式路径中分别维护内联拼接，这种做法提升了代码一致性和可维护性。对于编写基准测试或工具类脚本的工程师，这种小规模提取手法可借鉴。

功能与动机

关联 Issue #25267 报告：bench_serving --disable-stream 在推理模型（如 Qwen3、DeepSeek-R1）上因非流式路径仅读取 message.content 导致 content 为 null 时 generated_text=None，后续 calculate_metrics 中 tokenizer.encode 接收 None 抛出 ValueError。PR body 指出 streaming 路径已在 #23954 中修复，但非流式分支遗漏了同样的合并逻辑。

实现拆解

1. 新增共享文本合并函数：在 python/sglang/bench_serving.py 中定义 _combine_openai_chat_content(message)，以 (message.get("reasoning_content") or "") + (message.get("content") or "") 将两个字段拼接，确保任一字段为 None 时不会崩溃。
2. 替换非流式路径：在 async_request_openai_chat_completions 的 disable_stream 分支中，先提取 choices[0].message，再调用 _combine_openai_chat_content 赋值给 output.generated_text。
3. 统一流式路径：在流式 delta 处理中，将原本内联的 (delta.get("reasoning_content") or "") + (delta.get("content") or "") 替换为调用同一函数，消除代码重复。
4. 添加回归测试：在 test/registered/bench_fn/test_bench_serving_reasoning_stream.py 中新增 _JSONHandler 模拟非流式 JSON 响应、_StrictStringTokenizer 模拟 tokenizer 对非字符串输入的拒绝，以及 TestBenchServingReasoningNonStream 测试类覆盖三种响应形态（纯 reasoning、混合、纯 content），并验证 calculate_metrics 在严格 tokenizer 下正常工作。

关键文件：

- python/sglang/bench_serving.py（模块 基准测试；类别 source；类型 core-logic；符号 _combine_openai_chat_content）：核心变更文件，新增共享函数并替换两处调用点，是修复的核心逻辑。

- test/registered/bench_fn/test_bench_serving_reasoning_stream.py（模块 测试框架；类别 test；类型 test-coverage；符号 _JSONHandler, do_POST, log_message, _make_response）：增加了非流式场景的完整回归测试套件，包括模拟服务器、严格 tokenizer 和三种响应类型的测试。

关键符号：_combine_openai_chat_content, TestBenchServingReasoningNonStream._run, _make_response, _StrictStringTokenizer.encode

关键源码片段

python/sglang/bench_serving.py

核心变更文件，新增共享函数并替换两处调用点，是修复的核心逻辑。

```
# 新增共享函数，用于合并 OpenAI 聊天响应中的 reasoning_content 和 content
# 两个字段都可能为 None，or "" 确保始终返回字符串
def _combine_openai_chat_content(message: Dict[str, Any]) -> str:
    return (message.get("reasoning_content") or "") + (message.get("content") or "")

# 在非流式路径中替换原有直接读取 content 的逻辑
if args.disable_stream:
    response_json = await response.json()
    message = response_json["choices"][0]["message"]
    output.generated_text = _combine_openai_chat_content(message) # 原为 response_json[...]
    ["message"]["content"]

# 在流式 delta 路径中统一使用，消除重复内联拼接
delta = choices[0].get("delta") or {}
content = _combine_openai_chat_content(delta) # 原为 (delta.get("reasoning_content") or "") +
(delta.get("content") or "")
```

test/registered/bench_fn/test_bench_serving_reasoning_stream.py

增加了非流式场景的完整回归测试套件，包括模拟服务器、严格 tokenizer 和三种响应类型的测试。

```
# 模拟非流式 JSON 响应的 HTTP handler
class _JSONHandler(BaseHTTPRequestHandler):
    response_body: dict = {}
    request_bodies: list = []

    def do_POST(self):
        length = int(self.headers.get("Content-Length", "0"))
        if length:
            self.request_bodies.append(json.loads(self.rfile.read(length)))
        self.send_response(200)
        self.send_header("Content-Type", "application/json")
        self.end_headers()
        self.wfile.write(json.dumps(self.response_body).encode())
        self.wfile.flush()
```

```
# 严格 tokenizer，对非字符串输入抛出与原始 bug 相同的错误
class _StrictStringTokenizer:
    def encode(self, text, add_special_tokens=False):
        if not isinstance(text, str):
            raise ValueError("text input must be of type `str`")
        return text.split()

class TestBenchServingReasoningNonStream(CustomTestCase):
    def _run(self, response_body):
        set_global_args(Namespace(disable_stream=True, disable_ignore_eos=False, ...))
        port = _free_port()
        class Handler(_JSONHandler): pass
        Handler.response_body = response_body
        Handler.request_bodies = []
        server = HTTPServer(("127.0.0.1", port), Handler)
        thread = threading.Thread(target=server.serve_forever, daemon=True)
        thread.start()
        try:
            req = RequestFuncInput(prompt="hello", ...)
            return asyncio.run(async_request_openai_chat_completions(req)), Handler.request_
            bodies
        finally:
            server.shutdown()
            server.server_close()

    def test_reasoning_only_non_stream_metrics_retokenize_text(self):
        # content=None, reasoning_content="Let me think." 模拟 reasoning-only
        out, _ = self._run(
            _make_response(content=None, reasoning_content="Let me think.", completion_tokens=
            3)
        )
        self.assertTrue(out.success)
        self.assertEqual(out.generated_text, "Let me think.") # 验证拼接结果
        # 验证 calculate_metrics 能正常通过，不会触发 ValueError
        metrics = calculate_metrics([out], tokenizer=_StrictStringTokenizer(), ...)
        self.assertEqual(metrics["total_output_tokens"], 3)
```

评论区精华

本 PR 未产生实质性 Review 讨论，唯一 Review 来自 JustinTong0323 直接批准。Issue #25267 中的建议修复方案与实现一致。

- 暂无高价值评论线程

风险与影响

- 风险：风险极低。变更仅限基准测试脚本 bench_serving.py，不触及服务端核心路径。新增函数在流式和非流式路径中统一了行为，且测试覆盖了三种典型响应类型（reasoning-only、mixed、content-only），有效降低了回归风险。唯一潜在风险是少数用户可能依赖原始仅

content 的行为（忽略 reasoning_content），但 reasoning 模型的标准 OpenAI 响应中 reasoning_content 是显式字段，拼接逻辑符合直觉且与 streaming 路径对齐。

- 影响：影响范围：使用 bench_serving --disable-stream 对推理模型（如 Qwen3、DeepSeek-R1、Kimi-K2）进行基准测试的用户。这些用户之前会遇到崩溃，现在可以正常完成测试并获得指标。团队层面，消除了一个遗漏的 bug，统一了流式和非流式的文本拼接逻辑，提升了代码可维护性。无性能影响，测试 CI 时间略微增加（约 10s）。
- 风险标记：影响面小，测试覆盖三种场景，无安全或性能风险

关联脉络

- PR #25267 [Bug] bench_serving --disable-stream crashes on reasoning models (content is null): 该 issue 报告了 bug 并给出了建议修复，本次 PR 直接实现了该修复。