PR #23898 完整报告

sgl-project/sglang

fix(bench): wire request_func in bench_long_context ContextWorkloadGenerator

合并时间：2026-04-29 05:45

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23898

执行摘要

- 一句话：修复 bench_long_context 因缺 request_func 导致 AttributeError
- 推荐动作：此 PR 虽然改动小但具有教学意义：展示了基类初始化逻辑变更后子类易产生隐式回归，以及如何通过单元测试捕获契约属性。建议关注 #19724 的设计改变，并考虑是否需要对其他类似的子类做审查。测试代码的编写方式值得参考。

功能与动机

ContextWorkloadGenerator.init__完全替换了父类 WorkloadGenerator.__init，但父类原本在 #19724 中新增了 self.request_func = async_request_sglang_generate 的赋值。子类未能同步这一变动，导致调用 run() 时 request_sender 线程首次发送请求就会抛出 AttributeError。本 PR 旨在修复此回归并添加测试以防止再次发生。

实现拆解

实现分为两步：

1. 在 benchmark/hicache/bench_long_context.py 中导入 async_request_sglang_generate 并在 __init__中添加 self.request_func 赋值，同时将 URL 构建从两行简化为一行。
2. 新增 test/registered/unit/test_bench_long_context.py 单元测试文件，通过 CPU CI 注册运行，覆盖 request_func 存在性、基类契约属性、URL 格式和 ready_queue 大小等场景。测试使用临时文件模拟数据集并 mock tokenizer。

关键文件：

- test/registered/unit/test_bench_long_context.py（模块 基准测试；类别 test；类型 test-coverage；符号 _build_args, _fake_dataset, TestContextWorkloadGeneratorInit, setUp）：新增的单元测试文件，全面验证 ContextWorkloadGenerator 初始化后关键属性的存在性和正确性，防止回归。
- benchmark/hicache/bench_long_context.py（模块 基准脚本；类别 source；类型 dependency-wiring）：修复的核心文件：添加 request_func 赋值和简化 URL 构建，使 ContextWorkloadGenerator 能正常使用继承的 request_sender 方法。

关键符号：ContextWorkloadGenerator.init

关键源码片段

test/registered/unit/test_bench_long_context.py

新增的单元测试文件，全面验证 ContextWorkloadGenerator 初始化后关键属性的存在性和正确性，防止回归。

```
"""Unit test for benchmark/hicache/bench_long_context.py.

Guards against the regression where ContextWorkloadGenerator.__init__ replaces
WorkloadGenerator.__init__ entirely but forgets to set attributes the inherited
request_sender/handle_request methods need (e.g. self.request_func).
"""
import json
import sys
import tempfile
import unittest
from pathlib import Path
from types import SimpleNamespace
from unittest.mock import MagicMock, patch

from sglang.test.ci.ci_register import register_cpu_ci
from sglang.test.test_utils import CustomTestCase

register_cpu_ci(est_time=5, suite="stage-a-test-cpu")

REPO_ROOT = Path(__file__).resolve().parents[3]
HICACHE_DIR = REPO_ROOT / "benchmark" / "hicache"
if str(HICACHE_DIR) not in sys.path:
    sys.path.insert(0, str(HICACHE_DIR))

import bench_long_context # noqa: E402

from sglang.test.kits.cache_hit_kit import async_request_sglang_generate # noqa: E402


def _build_args(dataset_path: str) -> SimpleNamespace:
    # 构建与 bench_long_context.py 中预期一致的参数命名空间
    return SimpleNamespace(
        host="localhost",
        port=30000,
        model_path="meta-llama/Llama-3.2-1B-Instruct",
        distribution="poisson",
        request_rate=1.0,
        dataset_path=dataset_path,
        num_clients=2,
        max_parallel=2,
        log_file="performance_metrics.jsonl",
        tag="",
    )

def _fake_dataset() -> dict:
```

```
    # 创建一个最小的测试数据集，包含 2 条 query
    return {
        "contexts": ["ctx-zero ", "ctx-one "],
        "queries": [
            {"context": 0, "question": "q0", "reference_answer": "a0"},
            {"context": 1, "question": "q1", "reference_answer": "a1"},
        ],
    }


class TestContextWorkloadGeneratorInit(CustomTestCase):
    """验证 ContextWorkloadGenerator 初始化后所有继承方法依赖的属性和接口正确设置。"""

    def setUp(self):
        # 创建临时数据集文件
        self._tmp = tempfile.NamedTemporaryFile(mode="w", suffix=".json", delete=False)
        json.dump(_fake_dataset(), self._tmp)
        self._tmp.close()
        self.dataset_path = self._tmp.name

        # mock tokenizer 以避免实际加载模型
        mock_tokenizer = MagicMock()
        mock_tokenizer.encode.return_value = [1, 2, 3, 4]
        mock_tokenizer.return_value = {"input_ids": [5, 6]}

        # 将 bench_long_context 模块中的 get_tokenizer 替换为 mock
        self._tok_patch = patch.object(
            bench_long_context, "get_tokenizer", return_value=mock_tokenizer
        )
        self._tok_patch.start()

    def tearDown(self):
        self._tok_patch.stop()
        Path(self.dataset_path).unlink(missing_ok=True)

    def test_request_func_is_set(self):
        """核心回归防护：验证 request_func 存在且为 async_request_sglang_generate"""
        gen = bench_long_context.ContextWorkloadGenerator(
            _build_args(self.dataset_path)
        )
        self.assertTrue(callable(getattr(gen, "request_func", None)))
        self.assertIs(gen.request_func, async_request_sglang_generate)
```

benchmark/hicache/bench_long_context.py

修复的核心文件：添加 request_func 赋值和简化 URL 构建，使 ContextWorkloadGenerator 能正常使用继承的 request_sender 方法。

```
import json
import queue
```

```
import time

import requests
from bench_multiturn import (
    ReadyQueue,
    WorkloadGenerator,
    gen_payload,
    log_to_jsonl_file,
    parse_args,
)
from tqdm.asyncio import tqdm

from sglang.benchmark.utils import get_tokenizer
from sglang.test.kits.cache_hit_kit import async_request_sglang_generate # 新增导入


class ContextWorkloadGenerator(WorkloadGenerator):
    def __init__(self, args):
        # 合并 URL 构建为一行，移除未使用的 self.baseurl
        self.url = f"http://{args.host}:{args.port}/generate"
        # 关键修复：设置 request_func，使继承的 request_sender 可正常工作
        self.request_func = async_request_sglang_generate

        self.tokenizer = get_tokenizer(args.model_path)
        self.distribution = args.distribution
        self.request_rate = args.request_rate
        self.start_time = None
        self.finished_time = None

        self.sent_requests = 0
        self.completed_requests = 0

        self.dataset = json.load(open(args.dataset_path))
        num_requests = min(args.num_clients, len(self.dataset["queries"]))

        init_requests = []
        for i in range(num_requests):
            context_id = self.dataset["queries"][i]["context"]
            prompt_text = (
                self.dataset["contexts"][context_id]
                + self.dataset["queries"][i]["question"]
            )
            input_ids = self.tokenizer.encode(prompt_text)
            output_len = len(
                self.tokenizer(self.dataset["queries"][i]["reference_answer"])[
                    "input_ids"
                ]
            )
            init_requests.append((i, gen_payload(input_ids, output_len)))
```

```
        self.ready_queue = ReadyQueue(init_requests=init_requests)

        self.response_queue = queue.Queue()
```

评论区精华

本 PR 无多轮讨论。仅 reviewer somnathr 批准（"Looks good to me"），未产生争议或未解问题。

- 暂无高价值评论线程

风险与影响

- 风险：风险极低。修复仅限于 benchmark 工具脚本，不影响核心服务。新增的单元测试在 CPU CI 中运行（约 5 秒），有效防止未来类似回归。唯一的极低风险是 mocked tokenizer 与真实行为可能略有差异，但测试聚焦于属性存在性而非功能性，影响有限。
- 影响：直接影响：benchmark/hicache/bench_long_context.py 脚本可正常运行，不再因缺少 request_func 而崩溃。间接影响：无。该 PR 对系统其他部分无影响。影响范围局限于 HiCache 场景下的长上下文基准测试。
- 风险标记：低影响范围，有测试覆盖

关联脉络

- 暂无明显关联 PR