PR #27445 完整报告

sgl-project/sglang

Complete server warmup before scripted runtime scripts start

合并时间：2026-06-07 17:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27445

执行摘要

- 一句话：在脚本化测试开始前完成服务器预热
- 推荐动作：该 PR 展示了如何通过驱动调度器循环来确保测试环境清洁，值得测试基础设施和调度器相关开发者精读。_drive_engine_through_warmup 中的两轮 quiesce 逻辑是对 PP 异步特性的巧妙应对。讨论中关于 hasattr 的取舍也体现了对调度器事件循环初始化的深入理解。

功能与动机

脚本化运行时中，调度器循环在脚本命令之间冻结，导致预热线程的 /generate 请求在脚本执行期间任意时刻到达，作为外源流量污染调度器队列、批组成和重置排空，是 CI flaky 的放大器。同时 /health 在调度器冻结时无法完成，导致就绪检查超时。需要在脚本开始前完成预热并清除残留流量。

实现拆解

按以下步骤实现：

1. 在调度器钩子中驱动预热（scheduler_hook.py）：新增 _drive_engine_through_warmup 函数，在 _run_dispatch_loop 发送 HookReady 之前 yield from 调用。该函数驱动调度器循环，直到代理（ScriptedTokenizerRecvProxy）观察到至少一个工作请求且调度器 is_fully_idle() 持续两个 microbatch 轮转（补偿 PP 异步的瞬时空闲状态），并在 120 秒超时后报错。若设置 skip_server_warmup 则跳过。
2. 代理计数工作请求（tokenizer_recv_proxy.py）：为 ScriptedTokenizerRecvProxy 添加 work_reqs_seen 属性，在 _drain_underlying 中识别并计数进入的 TokenizedGenerateReqInput 等实际工作请求类型，供预热检测使用。
3. 修改 HTTP 就绪端点（http_server.py）：将 _await_http_ready 中的检查端点从 /health 改为 /model_info，因为 /health 在调度器被脚本驱动时会产生一个真正的生成请求，而该请求无法在脚本间隙完成；/model_info 仅返回元数据，证明端口已绑定且路由已注册。

配套测试：无额外测试，仅影响现有测试的运行方式。该变更是 PR 1/2，PR 2（#27446）修复 is_fully_idle 在 PP 下的遗漏问题。

关键文件：

- python/sglang/test/scripted_runtime/scheduler_hook.py（模块 调度钩子；类别 test；类型 test-coverage；符号 _drive_engine_through_warmup）：核心变更文件，新增 _drive_engine_through_warmup 函数，在发送 HookReady 前驱动调度器处理预热请求，是解决 flaky 的关键。
- python/sglang/test/scripted_runtime/tokenizer_recv_proxy.py（模块 接收代理；类别 test；类型 test-coverage）：新增工作请求计数，为预热检测提供观察点。
- python/sglang/test/scripted_runtime/http_server.py（模块 HTTP 服务器；类别 test；类型 test-coverage）：将健康检查端点从 /health 改为 /model_info，避免在调度器冻结时挂起。

关键符号：_drive_engine_through_warmup, _await_http_ready, ScriptedTokenizerRecvProxy._drain_underlying

关键源码片段

python/sglang/test/scripted_runtime/scheduler_hook.py

核心变更文件，新增 _drive_engine_through_warmup 函数，在发送 HookReady 前驱动调度器处理预热请求，是解决 flaky 的关键。

```
# 文件：python/sglang/test/scripted_runtime/scheduler_hook.py
# 新增函数：_drive_engine_through_warmup
# 在 _run_dispatch_loop 中 yield from 调用，确保脚本开始前服务器已预热完毕

def _drive_engine_through_warmup(ctx: ScriptedContext) -> Generator:
    # Run the engine until the server warmup request has been received and
    # fully processed, so scripts never observe foreign warmup traffic.
    scheduler = ctx.scheduler
    server_args = scheduler.server_args
    if server_args.skip_server_warmup:
        logger.info('scripted_runtime: skip_server_warmup set, not driving warmup')
        return

    logger.info('scripted_runtime: driving engine until server warmup completes')
    start_time = time.monotonic()

    # is_fully_idle() 在 PP 下可能瞬时报告空闲（microbatch 仍在传输中），
    # 因此要求空闲状态连续保持两个 microbatch 轮转周期
    quiesce_iters = 2 * (server_args.pp_size + server_args.pp_async_batch_depth)
    proxy = ctx._tokenizer_recv_proxy
    deadline = start_time + WARMUP_DRIVE_TIMEOUT_S

    idle_streak = 0
    while idle_streak < quiesce_iters:
        if time.monotonic() >= deadline:
            raise RuntimeError(
                'scripted_runtime: server warmup did not complete within '
                f'{WARMUP_DRIVE_TIMEOUT_S}s '
                f'(work_reqs_seen={proxy.work_reqs_seen}, '
```

```
                f'idle_streak={idle_streak})'
            )
        yield # 让调度器循环前进一次
        # 当代理观察到至少一个工作请求且调度器完全空闲时，增加 idle_streak
        if proxy.work_reqs_seen > 0 and scheduler.is_fully_idle():
            idle_streak += 1
        else:
            idle_streak = 0

    logger.info(
        'scripted_runtime: server warmup drained in %.1fs (work_reqs_seen=%d)',
        time.monotonic() - start_time,
        proxy.work_reqs_seen,
    )
```

python/sglang/test/scripted_runtime/tokenizer_recv_proxy.py

新增工作请求计数，为预热检测提供观察点。

```
# 文件：python/sglang/test/scripted_runtime/tokenizer_recv_proxy.py
# 在 _drain_underlying 中计数实际工作请求

from sglang.srt.managers.io_struct import (
    BatchTokenizedEmbeddingReqInput,
    BatchTokenizedGenerateReqInput,
    TokenizedEmbeddingReqInput,
    TokenizedGenerateReqInput,
)

_WORK_REQ_TYPES = (
    TokenizedGenerateReqInput,
    TokenizedEmbeddingReqInput,
    BatchTokenizedGenerateReqInput,
    BatchTokenizedEmbeddingReqInput,
)


class ScriptedTokenizerRecvProxy:
    def __init__(self, *, underlying: zmq.Socket) -> None:
        self._underlying = underlying
        self._buffer: deque = deque()
        self.work_reqs_seen: int = 0 # 新增：记录进入的工作请求总数

    def _drain_underlying(self) -> None:
        while True:
            try:
                req = self._underlying.recv_pyobj(zmq.NOBLOCK)
            except zmq.ZMQError:
                break
            # 仅对真正的工作请求类型进行计数，忽略控制消息
```

```
if isinstance(req, _WORK_REQ_TYPES):
    self.work_reqs_seen += 1
self._buffer.append(req)
```

评论区精华

Gemini Code Assist 机器人建议在 is_fully_idle 中使用 hasattr 检查 running_mbs 以避免 AttributeError。作者 fzyzcjy 反驳：所有 is_fully_idle 调用点都在调度器事件循环内，且 init_pp_loop_state 已在 pp_size > 1 时初始化了这些属性，直接访问更安全且不会掩盖初始化顺序错误。该讨论已解决，未采纳建议。

- is_fully_idle 中对 running_mbs 的防御性检查 (correctness): 未采纳 hasattr 建议，保持直接访问。

风险与影响

- 风险：主要风险：若预热驱动逻辑未能正确检测完成，可能引入 120 秒超时并报错，导致测试失败（而非 flaky）。但超时机制比原有不可控的污染更可靠。若 is_fully_idle 在特定条件下返回错误值（例如 PR 2 修复前），可能导致过早结束预热，但 PR 2 会修复该问题。此外，改动仅限测试环境，无生产影响。
- 影响：对用户无影响；对系统：测试环境中的脚本化运行时代理和 HTTP 就绪检查行为改变，预热现在在脚本开始前完成，消除了外源流量引起的 flaky。对团队：CI 中与脚本化运行时相关的测试将更稳定，减少 rerun。该变更是测试基础设施改进，影响范围仅限于启用脚本化运行时的测试（主要涉及 chunked_prefill 和单元测试）。
- 风险标记：测试基础设施变更，依赖后续修复，超时风险

关联脉络

- PR #27446 Fix PP is_fully_idle missing in-flight microbatches: 本 PR 是 PR 1/2，PR 2（#27446）修复了 is_fully_idle 在 PP 下遗漏 in-flight microbatches 的问题，与本 PR 的预热驱动逻辑直接关联。