PR #27413 完整报告

sgl-project/sglang

Add scripted-runtime unit, core integration, and chunked-prefill tests

合并时间：2026-06-06 09:08

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27413

执行摘要

- 一句话：为 scripted-runtime 添加单元 / 集成和 chunked-prefill 测试
- 推荐动作：本 PR 值得精读，特别是对 sglang 测试基础设施感兴趣的团队成员。test_scripted_runtime_core.py 展示了如何通过生成器脚本驱动调度器步进测试，这种模式可复用于其他模块的集成测试。test_scripted_core_1gpu.py 中的生命周期暂停测试设计精巧，覆盖了 pause_generation(mode='retract') 后的 waiting_queue 行为和输出冻结验证。建议所有 scripted-runtime 的相关修改都运行这些测试以确保不破坏语义。

功能与动机

从 PR 标题和内容看，作者（同时也是合并者）旨在为 scripted-runtime 填补测试空白。SCripted-runtime 是 sglang 测试基础设施的一部分，此前缺乏系统的测试覆盖，导致调度器核心逻辑（如请求生命周期、chunked prefill、暂停恢复等）的稳定性依赖手动验证。本 PR 通过脚本驱动的方式，将调度器关键路径的验证自动化，防止后续重构引入回归。

实现拆解

1. 核心 API 集成测试（test_scripted_runtime_core.py）：基于 ScriptedTestCase 编写 9 个测试，覆盖自动 / 显式 rid 生成、find_req_by_rid 命中与缺失、is_finished 语义、ReqHandle.req/finished 属性、is_chunking 状态检测、abort_all 终止、finish 状态传递等场景，每个测试通过生成器脚本与调度器交互。
2. Chunked-prefill 集成测试（test_scripted_core_1gpu.py、test_scripted_core_4gpu.py）：验证多 chunk 预填充的冒烟、不同边界偏移、生命周期各阶段的暂停 /retract 后恢复、abort_all 以及流水线并行下 chunk 大小组合扫描。其中暂停测试严格检查调度器 waiting_queue 状态和 output_ids 冻结。
3. 辅助组件单元测试：test_background_http_poster.py（线程 / 协程生命周期、异常处理）、test_tokenizer_recv_proxy.py（消息队列 / 等待 / 过期处理）、test_http_server.py（执行脚本 / 回复匹配 / 超时）、test_scheduler_hook.py（生成器推进结果检测）、test_scripted_runtime_utils.py（函数路径解析）。均使用 fake/Mock 隔离外部依赖。
4. 通用测试帮助函数（scripted_runtime_chunked_helpers.py）：提供 run_until_finished、advance_to_nth_chunk、exhaust_row_pool 等可复用生成器，封装调度推进逻辑。
5. CI 注册：通过 register_cuda_ci/register_cpu_ci 将每个测试分配给特定 stage 和 GPU 配置，确保并行测试效率。

关键文件：

- test/registered/scripted_runtime/test_scripted_runtime_core.py（模块 运行核心测试；类别 test；类型 test-coverage；符号 _script_noop, TestScriptedRuntimeCore, test_start_req_auto_rid_and_finishes, _script_start_req_auto_rid_and_finishes）：核心集成测试，覆盖 scripted-runtime 主要 API 功能。
- test/registered/chunked_prefill/test_scripted_core_1gpu.py（模块 分块预填充测试；类别 test；类型 test-coverage；符号 _advance_to_stage, TestScriptedCore, test_chunked_prefill_smoke, _script_chunked_prefill_smoke）：单卡 chunked prefill 集成测试，包含边界偏移和生命周期暂停验证。
- python/sglang/test/scripted_runtime_chunked_helpers.py（模块 测试辅助库；类别 test；类型 test-coverage；符号 base_engine_kwargs, run_until, run_until_finished, run_until_all_finished）：通用测试辅助函数，被所有集成测试使用。
- test/registered/unit/scripted_runtime/test_background_http_poster.py（模块 HTTP 后台测试；类别 test；类型 test-coverage；符号 _FakeResponse, _FakePostCM, _FakeSession, TestBackgroundHttpPosterLifecycle）：BackgroundHttpPoster 组件的单元测试，覆盖线程生命周期、协程提交、异常处理等。
- test/registered/unit/scripted_runtime/test_tokenizer_recv_proxy.py（模块 代理接收测试；类别 test；类型 test-coverage；符号 _ControlMsg, _StartReq, _FakeUnderlyingSocket, TestScriptedTokenizerRecvProxyRecv）：TokenizerRecvProxy 的单元测试，验证消息队列和等待语义。
- test/registered/unit/scripted_runtime/test_http_server.py（模块 HTTP 服务测试；类别 test；类型 test-coverage；符号 _sample_script, _FakePairSocket, _FakeProcess, TestExecuteScriptReplyMatching）：ScriptedHttpServer 的单元测试，覆盖执行脚本、超时、异常回复等。
- test/registered/unit/scripted_runtime/test_scheduler_hook.py（模块 调度钩子测试；类别 test；类型 test-coverage；符号 _yielding_gen, _empty_gen, _raising_gen, TestAdvanceGenerator）：调度器 hook（生成器推进）的单元测试。
- test/registered/unit/scripted_runtime/test_scripted_runtime_utils.py（模块 工具函数测试；类别 test；类型 test-coverage；符号 TestResolveFn）：工具函数（函数路径解析）的单元测试。
- test/registered/chunked_prefill/test_scripted_core_4gpu.py（模块 分块预填充测试；类别 test；类型 test-coverage；符号 TestScriptedPpChunkSweep, test_pp_chunk_sweep, _script_pp_one_combo）：四卡流水线并行下 chunked prefill 的集成测试。
- test/registered/chunked_prefill/test_scripted_swa_1gpu.py（模块 SWA 分块测试；类别 test；类型 test-coverage；符号 TestScriptedSwaChunkedReqEarlyReturn, test_swa_chunked_req_early_return_no_double_free, _script_swa_chunked_req_early_return_no_double_free）：SWA（滑动窗口注意力）在 chunked prefill 下的测试。

关键符号：_script_noop, start_req, find_req_by_rid, run_until_finished, advance_to_nth_chunk, advance_to_lifecycle_stage, pause_generation,

continue_generation, abort_all, execute_script, recv_pyobj,
wait_until_arrived,submit_coro, close, resolve_fn

关键源码片段

test/registered/scripted_runtime/test_scripted_runtime_core.py

核心集成测试，覆盖 scripted-runtime 主要 API 功能。

```
# test/registered/scripted_runtime/test_scripted_runtime_core.py
# 展示两个典型测试：自动 rid 与显式 rid

class TestScriptedRuntimeCore(ScriptedTestCase):
    ENGINE_KWARGS = _ENGINE_KWARGS

    def test_start_req_auto_rid_and_finishes(self):
        # 执行脚本，验证自动生成的 rid 以 "scripted-" 开头且请求最终完成
        self.server.execute_script(self._script_start_req_auto_rid_and_finishes)

    @staticmethod
    def _script_start_req_auto_rid_and_finishes(t: ScriptedContext):
        r = t.start_req(prompt_len=_SHORT_PROMPT_LEN, max_new_tokens=4)
        # 验证自动 rid 格式
        assert r.rid.startswith("scripted-"), f"unexpected auto rid {r.rid!r}"
        # 步进直到请求完成
        yield from run_until_finished(r)
        assert r.finished, "auto-rid req did not finish"

    def test_start_req_explicit_rid(self):
        self.server.execute_script(self._script_start_req_explicit_rid)

    @staticmethod
    def _script_start_req_explicit_rid(t: ScriptedContext):
        # 使用显式 rid 启动请求
        r = t.start_req(
            prompt_len=_SHORT_PROMPT_LEN, max_new_tokens=2, rid="explicit-rid-test"
        )
        # 验证 rid 被正确设置
        assert r.rid == "explicit-rid-test", f"explicit rid not honored: {r.rid!r}"
        yield
        # 一步之后，调度器应能看到该请求
        assert (
            t.find_req_by_rid("explicit-rid-test") is not None
        ), "explicit rid not visible to the scheduler after one step"
```

test/registered/chunked_prefill/test_scripted_core_1gpu.py

单卡 chunked prefill 集成测试，包含边界偏移和生命周期暂停验证。

```
# test/registered/chunked_prefill/test_scripted_core_1gpu.py
# 展示暂停 / 恢复测试：在生命周期每个阶段执行 pause(retract)，验证调度器状态冻结
```

```
class TestScriptedCore(ScriptedTestCase):
    ENGINE_KWARGS = base_engine_kwargs(chunked_prefill_size=_CHUNK_SIZE)

    def test_pause_retract_at_lifecycle_points_then_resume(self):
        # 遍历定义的 5 个生命周期阶段：first_chunk, last_chunk, first_decode, mid_decode, last_
        decode
        for stage in LIFECYCLE_STAGES:
            with self.subTest(stage=stage):
                self.server.execute_script(
                    self._script_pause_retract_at_stage,
                    args=(stage,),
                )

    @staticmethod
    def _script_pause_retract_at_stage(t: ScriptedContext, stage: str):
        r = t.start_req(
            prompt_len=_PROMPT_LEN, max_new_tokens=_LIFECYCLE_MAX_NEW_TOKENS
        )
        # 步进到指定生命周期阶段
        yield from _advance_to_stage(r, stage)
        assert r.req is not None, f"stage={stage}: req vanished before pause"

        # 执行 pause(retract) —— 请求应回退到 waiting_queue
        t.pause_generation(mode="retract")
        yield

        # 如果请求尚未结束（最后解码阶段可能立即完成），验证调度器状态
        if not r.finished:
            req = r.req
            # 必须回到 waiting_queue
            assert req is not None and req in t.scheduler.waiting_queue, (
                f"stage={stage}: pause(retract) should park the req back in waiting_queue"
            )
            output_tokens_after_pause = len(req.output_ids)
            # 再推进 3 步，输出 tokens 不得增加
            for _ in range(3):
                yield
                req = r.req
                assert (
                    req is not None and len(req.output_ids) == output_tokens_after_pause
                ), (
                    f"stage={stage}: paused engine advanced the req "
                    f"({len(req.output_ids) if req is not None else None} output tokens, expected
                    {output_tokens_after_pause})"
                )

        # 恢复生成，最终完成
        t.continue_generation()
        yield from run_until_finished(r)
```

```
        assert r.finished, f"stage={stage}: req did not finish after pause/continue"
```

python/sglang/test/scripted_runtime_chunked_helpers.py

通用测试辅助函数，被所有集成测试使用。

```
# python/sglang/test/scripted_runtime_chunked_helpers.py
# 核心辅助函数：run_until 和 run_until_finished 驱动调度器步进直到条件满足

from __future__ import annotations
from typing import Any, Dict, List

DEFAULT_CHUNK_SIZE: int = 256
DEFAULT_MAX_STEPS: int = 400
SMALL_MODEL: str = "Qwen/Qwen3-0.6B"

def base_engine_kwargs(
    *,
    model_path: str = SMALL_MODEL,
    chunked_prefill_size: int = DEFAULT_CHUNK_SIZE,
    **overrides: Any,
) -> Dict[str, Any]:
    """生成引擎参数字典，默认使用小模型和指定chunk大小。"""
    kwargs: Dict[str, Any] = dict(
        model_path=model_path,
        chunked_prefill_size=chunked_prefill_size,
    )
    kwargs.update(overrides)
    return kwargs

def run_until(handle, predicate, *, max_steps: int = DEFAULT_MAX_STEPS):
    """生成器：每次yield后检查predicate(handle)，满足则返回，否则继续直到max_
    steps后断言失败。"""
    for _ in range(max_steps):
        if predicate(handle):
            return
        yield
    raise AssertionError(
        f"run_until: predicate never satisfied after {max_steps} steps "
        f"(handle rid={handle.rid!r}, finished={handle.finished})"
    )

def run_until_finished(handle, *, max_steps: int = DEFAULT_MAX_STEPS):
    """等待单个请求完成。"""
    yield from run_until(handle, lambda h: h.finished, max_steps=max_steps)

def run_until_all_finished(handles: List[Any], *, max_steps: int = DEFAULT_MAX_STEPS):
    """等待多个请求全部完成。"""
    for _ in range(max_steps):
        if all(h.finished for h in handles):
```

```
            return
        yield
    raise AssertionError(
        f"run_until_all_finished: not all reqs finished after {max_steps} steps "
        f"(finished={[h.finished for h in handles]})"
    )
# ... 其他函数如 warmup_radix, exhaust_row_pool, advance_to_nth_chunk 等
```

评论区精华

无 review 评论。PR 为作者自行合并，未经过公开讨论。

- 暂无高价值评论线程

风险与影响

- 风险：低风险。所有变更均为测试代码，未修改生产源码。潜在风险包括：（1）测试可能因环境依赖（如 GPU 型号、CUDA 版本）产生不稳定假阴性，尤其是依赖特定显存大小的 exhaust_row_pool 和 test_scripted_core_4gpu.py；（2）新增 CI 测试约 1550 秒的时间开销，可能延长 CI pipeline；（3）部分测试（如 test_scripted_runtime_core.py 中 FINISH_ABORT 的引入）需要确保与最新调度器行为一致，若调度器内部逻辑变化需同步更新测试。
- 影响：
  - 用户：无直接影响，用户不运行这些测试。
  - 系统：CI 流水线新增约 10 个测试用例，总执行时间约 26 分钟（CUDA + CPU），但分布在多个 stage 和 runner 上，不会阻塞关键路径。
  - 团队：开发者修改 scripted-runtime 或调度器相关代码时，可通过这些测试快速验证正确性，降低手动测试成本。测试本身也是文档，展示了 API 的预期用法和边界条件。
- 风险标记：新增测试可能影响 CI 稳定性，部分测试依赖具体 GPU 显存容量，测试与调度器行为耦合，需同步更新

关联脉络

- 暂无明显关联 PR