PR #27412 完整报告

sgl-project/sglang

Add scripted-runtime KV-pool and lock-ref exhauster primitives

合并时间：2026-06-06 09:07

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27412

执行摘要

本 PR 为 scripted runtime 测试框架新增了 KV 池和锁引用两个 " 耗尽器 " 原语，使测试能够精确模拟内存压力场景，为后续 chunked-prefill 等复杂测试提供基础设施。变更仅涉及测试辅助代码，无生产路径风险。

功能与动机

scripted runtime 测试需要验证调度器在 KV 缓存不足或 radix 树节点被锁定时的行为。原有框架无法精细控制这些压力条件，测试覆盖面受限。本 PR 引入 ScriptedKvPoolExhauster 和 ScriptedLockRefExhauster，填补了这一空白。

实现拆解

1. 新增 lock_ref_exhauster.py：ScriptedLockRefExhauster 类通过 DFS 遍历 radix 树，对锁引用为 0 的节点调用 inc_lock_ref，逐步减少可驱逐节点数，模拟节点锁耗尽。
2. 新增 kv_pool_exhauster.py：ScriptedKvPoolExhauster 类通过 token 分配器截留超过 leave_pages 的 token，模拟 KV 页池紧张。
3. 修改 api.py：在 ScriptedContext 中初始化两个 exhauster，添加 exhaust_kv、exhaust_lock_refs 方法供测试调用，以及 _release_exhausted_pools 用于清理。
4. 修改 scheduler_hook.py：在 _reset_engine_state 中先释放所有耗尽资源，避免状态残留影响后续测试。

ScriptedLockRefExhauster 实现

```
from __future__ import annotations
from typing import TYPE_CHECKING, Any, List

from sglang.test.scripted_runtime.context.radix import _node_lock_ref

if TYPE_CHECKING:
    from sglang.srt.managers.scheduler import Scheduler


class ScriptedLockRefExhauster:
    """用于在测试中模拟 radix 树节点锁引用耗尽压力的辅助类。"""

    def __init__(self, scheduler: "Scheduler") -> None:
        self.scheduler = scheduler
```

```
        self._locked: List[Any] = [] # 记录本批锁定的节点，用于后续释放

    def exhaust(self, *, leave_refs: int) -> None:
        """
        遍历 radix 树，将未锁定节点逐一加锁，
        直到剩余未锁定节点数不超过 leave_refs。
        """
        tree_cache = self.scheduler.tree_cache
        if tree_cache.disable:
            return

        while True:
            evictable = self._evictable_nodes()
            if len(evictable) <= leave_refs:
                return

            target = evictable[0]
            tree_cache.inc_lock_ref(target)

            newly_locked = [node for node in evictable if _node_lock_ref(node) > 0]
            if not newly_locked:
                return
            self._locked.append(target)

    def release(self) -> None:
        """释放本批所有锁定的节点。"""
        tree_cache = self.scheduler.tree_cache
        for node in self._locked:
            tree_cache.dec_lock_ref(node)
        self._locked.clear()

    def _evictable_nodes(self) -> List[Any]:
        """返回当前锁引用为 0 的所有节点（可驱逐节点）。"""
        evictable: List[Any] = []
        stack = list(self.scheduler.tree_cache.root_node.children.values())
        while stack:
            node = stack.pop()
            if _node_lock_ref(node) == 0:
                evictable.append(node)
            stack.extend(node.children.values())
        return evictable
```

ScriptedKvPoolExhauster 实现

```
from __future__ import annotations
from typing import TYPE_CHECKING, List

if TYPE_CHECKING:
    import torch
    from sglang.srt.managers.scheduler import Scheduler
```

```
class ScriptedKvPoolExhauster:
    """用于在测试中模拟 KV 页池不足压力的辅助类。"""

    def __init__(self, scheduler: "Scheduler") -> None:
        self.scheduler = scheduler
        self._held: List["torch.Tensor"] = [] # 保存已分配的张量，用于释放

    def exhaust(self, *, leave_pages: int) -> None:
        """
        从 token 分配器分配超出 leave_pages 的部分，
        使可用 token 数降低到目标值以下。
        """
        allocator = self.scheduler.token_to_kv_pool_allocator

        leave_tokens = leave_pages * self.scheduler.page_size
        need = allocator.available_size() - leave_tokens
        if need <= 0:
            return

        held = allocator.alloc(need)
        assert (
            held is not None
        ), f"exhaust_kv: allocator could not grab {need} tokens to create pressure"
        self._held.append(held)

    def release(self) -> None:
        """释放所有预分配的 token。"""
        for held in self._held:
            self.scheduler.token_to_kv_pool_allocator.free(held)
        self._held.clear()
```

评论区精华

无公开 review 评论。

风险与影响

- 风险：仅测试代码，无生产风险。但 ScriptedKvPoolExhauster.exhaust 中的 assert 在分配失败时会直接崩溃，测试编写者需确保参数合理。
- 影响：使测试能够精确模拟节点锁耗尽和 KV 池耗尽两种压力场景，提升调度器相关测试的覆盖率和可靠性。

关联脉络

本 PR 提供的原语将被 PR #27413 (chunked-prefill 测试) 直接使用，是 scripted runtime 测试能力提升的重要基石。