PR #27710 完整报告

sgl-project/sglang

Add UT guarding per-request bookkeeping clock ownership

合并时间：2026-06-10 08:11

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27710

执行摘要

- 一句话：新增 AST 契约测试防止 bookkeeping 时钟越权修改
- 推荐动作：此 PR 值得精读，尤其是其使用 AST 静态分析实现契约测试的设计模式，可以推广到其他需要保护关键数据修改点的场景。白名单 + 计数检查的方式既严格又灵活，值得借鉴。

功能与动机

PR body 指出，per-request bookkeeping 时钟的错误修改（如 draft worker 中重复运行）导致 SWA 前缀锁提前释放等静默问题，在 e2e CI 和 idle leak checker 中不可见。#27607 修复了 Frozen-KV MTP 中出现的 double-run bookkeeping 问题，此测试用于防止同类问题再次发生。

实现拆解

1. 定义跟踪属性与白名单：
   - 在 _TRACKED_ATTRS 中列出需要跟踪的 5 个属性（decode_batch_idx、extend_batch_idx、kv_committed_len、kv_allocated_len、spec_verify_ct）以及 maybe_evict_swa() 方法。
   - 在 _OWNER_SITES 中枚举所有经过审查的修改点，每个条目包含（相对路径、作用域、变体类型）和期望的修改次数。
2. AST 扫描与计数：
   - _scan_srt() 递归扫描 python/sglang/srt 目录下的所有 .py 文件，使用 Python ast 模块解析每个文件。
   - _scan_tree() 对每个 AST 树遍历节点，识别对跟踪属性的赋值操作（ast.Assign 或 ast.AugAssign）以及对 maybe_evict_swa 的方法调用，并记录所在的作用域（类名 + 方法名）。
   - _is_zero_reset() 判断赋值是否为 = 0 的重置操作，这类操作被豁免（不计入白名单计数）。
3. draft worker 独立检查：
   - 通过 _draft_worker_classes() 发现所有继承自 EagleDraftInputV2Mixin 的子类（即 spec-v2 draft worker）。

- _scan_class_subtree() 确保这些类中没有对跟踪属性的任何修改，因为其 bookkeeping 已由 scheduler-driven 的 prepare_for_decode mixin 或 resolve 路径负责。

4. 断言与失败信息：
   - 将实际计数与 _OWNER_SITES 对比，输出差异（新增、缺失、计数不等）并触发断言。
   - 任何未列入白名单的修改点都会导致测试失败，提示开发者审查并更新白名单。
5. CI 注册：
   - 通过 register_cpu_ci 注册为 CPU 测试，预估时间 8 秒，归入 base-a-test-cpu suite。

关键文件：

- test/registered/unit/spec/test_decode_bookkeeping_ownership.py（模块 所有权测试；类别 test；类型 test-coverage；符号 _iter_scoped_nodes, visit, _is_zero_reset, _scan_tree）：唯一变更文件，实现完整的 AST 契约测试，用于防止 per-request bookkeeping 时钟的越权修改。

关键符号：_iter_scoped_nodes, visit, _is_zero_reset, _scan_tree, _parse, _scan_srt, _draft_worker_classes, _scan_class_subtree

关键源码片段

test/registered/unit/spec/test_decode_bookkeeping_ownership.py

唯一变更文件，实现完整的 AST 契约测试，用于防止 per-request bookkeeping 时钟的越权修改。

```
"""Ownership contract for per-request bookkeeping clocks.

Per-request accounting state (`decode_batch_idx` / `extend_batch_idx` iter
clocks, `kv_committed_len` / `kv_allocated_len` KV watermarks,
`spec_verify_ct`, and the `maybe_evict_swa()` call) must only be advanced by
the reviewed owner sites in _OWNER_SITES; spec-v2 draft workers must not
repeat any of them (the scheduler-driven mixin / resolve path already does).
A clock that runs fast fires SWA eviction in the overlap race window and
releases the SWA prefix lock early; neither shows up in e2e CI or the idle
leak checker, hence this AST-level guard.
"""

# 被跟踪的 bookkeeping 属性名（赋值操作会触发计数）
_TRACKED_ATTRS = (
    "decode_batch_idx",
    "extend_batch_idx",
    "kv_committed_len",
    "kv_allocated_len",
    "spec_verify_ct",
)
# 被跟踪的 SWA 逐出方法名（调用会触发计数）
_EVICT_METHOD = "maybe_evict_swa"

# 白名单字典：{（相对路径，作用域，变体类型）: 期望次数}
```

```
# 变体类型为属性名或 "evict"
_OWNER_SITES = {
    # 非 spec 调度器
    ("managers/schedule_batch.py", "ScheduleBatch.prepare_for_decode", "decode_batch_idx"): 1,
    ("managers/schedule_batch.py", "ScheduleBatch.prepare_for_decode", "kv_committed_len"): 1,
    ("managers/schedule_batch.py", "ScheduleBatch.prepare_for_decode", "kv_allocated_len"): 1,
    # 省略其他已审查条目 ...
}

def _scan_srt() -> dict:
    """遍历 srt 目录下的所有 .py 文件，返回实际计数 dict。"""
    counts: dict = {}
    for py_file in sorted(_SRT_DIR.rglob("*.py")):
        if "site-packages" in py_file.parts:
            continue
        relative = py_file.relative_to(_SRT_DIR).as_posix()
        tree = ast.parse(py_file.read_text())
        # 对每个文件进行 AST 扫描
        _scan_tree(tree, relative, counts)
    return counts

def _scan_tree(tree: ast.AST, relative: str, counts: dict) -> None:
    """递归遍历 AST 树，记录赋值和 maybe_evict_swa 调用计数。"""
    scope_of = {}
    def visit(node, scope):
        if isinstance(node, (ast.FunctionDef, ast.AsyncFunctionDef, ast.ClassDef)):
            # 作用域：类名 . 方法名
            scope = f"{scope}.{node.name}" if scope else node.name
        scope_of[node] = scope
        # 处理赋值语句：获取被赋值的属性名
        if isinstance(node, ast.Assign):
            for target in node.targets:
                if isinstance(target, ast.Attribute) and target.attr in _TRACKED_ATTRS:
                    # 判断是否为 = 0 的重置（豁免）
                    if not _is_zero_reset(node.value):
                        key = (relative, scope, target.attr)
                        counts[key] = counts.get(key, 0) + 1
        # 处理方法调用：maybe_evict_swa()
        elif isinstance(node, ast.Expr) and isinstance(node.value, ast.Call):
            if isinstance(node.value.func, ast.Attribute) and node.value.func.attr == _EVICT_
            METHOD:
                key = (relative, scope, "evict")
                counts[key] = counts.get(key, 0) + 1
        # 递归子节点
        for child in ast.iter_child_nodes(node):
            visit(child, scope)
    visit(tree, "")
```

评论区精华

PR 没有 review 评论。PR 评论中只有 gemini-code-assist 的配额警告和作者触发的 /tag-and-rerun-ci 指令，无实质性讨论。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 白名单维护成本：当新增合法的 bookkeeping 操作时，必须同步更新 _OWNER_SITES，否则测试会失败，可能影响开发效率。
  2. AST 解析脆弱性：测试依赖 AST 树结构识别赋值和调用，如果代码风格变化（如使用 setattr 或动态属性）可能导致漏检或误报。
  3. 重构敏感：函数 / 类重命名、文件移动等会导致白名单条目失效，需同时更新测试。
  4. 无源码效率影响：测试仅在 CI 中运行约 8 秒，不影响运行时性能。 - 影响：对用户无直接影响。对系统，增加了 bookkeeping 变更时的编译期契约，防止静默回归。对团队，在修改相关代码时需要维护该测试文件的允许列表，但能显著降低难以调试的 bookkeeping 错误风险。 - 风险标记：白名单维护成本，AST 解析对重构敏感，无源码变更但需持续同步

关联脉络

- PR #27607 Support spec v2 for Frozen-KV MTP; remove v1 worker: 本 PR 的测试用于守卫 #27607 中修复的 double-run bookkeeping 错误，防止回归。