PR #27228 完整报告

sgl-project/sglang

Enable runtime busy memory check for speculation topk>1

合并时间：2026-06-05 04:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27228

执行摘要

- 一句话：启用 spec topk>1 的 busy 内存检查
- 推荐动作：建议精读，涉及 speculative decoding 内存管理的关键不变检查，以及测试基础设施中 MRO 合并环境覆盖的可复用模式。修复的身份比较问题是一个典型的 dataclass 陷阱，值得注意。

功能与动机

PR body 指出：busy-time KV-pool invariant check 在 speculative_eagle_topk > 1 时被早期返回禁用，该条件早于 uncached 统计功能，导致每次触发。通过减去每个请求的 uncached 量，现在不变式在 topk>1 时也能平衡。同时修复了 DP-attention 路径上触发的一个潜在崩溃（ScheduleBatch.__eq__比较 tensor 字段报错）。

实现拆解

1. 移除 early-return 和 warnings 导入：在 SchedulerInvariantChecker.self_check_during_busy 中删除 topk>1 的早期返回和 warnings.warn，以及不再需要的 import warnings。
2. 修复 batch 身份比较：在 _get_total_uncached_sizes 中将 not in (None, self.get_last_batch()) 替换为显式的 is not None and is not last_batch，避免因 ScheduleBatch.__eq__ 触发 tensor 比较崩溃。
3. 添加 MRO 环境覆盖合并：在 SpecEagleServerBase 中新增 _merged_env_overrides 类方法，沿 MRO 反向迭代，收集每个类的 env_overrides，使子类只需声明自己的覆盖，基类覆盖自动生效且子类可覆盖同名变量。
4. 测试类启用 busy 检查：在所有 spec EAGLE 测试类（包括 stress、topk、page、fa3、dp-attention 等）的 env_overrides 中添加 SGLANG_ENABLE_STRICT_MEM_CHECK_DURING_BUSY=1，并在部分类中移除不再需要的 SGLANG_ENABLE_SPEC_V2 覆盖。
5. 调整为 level 1 安静模式：从 level 2（verbose）降级到 level 1（quiet），仅在有泄漏时输出历史日志，减少 CI 日志噪声。

关键文件：

- python/sglang/srt/managers/scheduler_components/invariant_checker.py（模块 调度器；类别 source；类型 core-logic；符号 self_check_during_busy, _get_total_uncached_sizes）：核心逻辑变更：移除 topk>1 早期返回、修复 batch 身份比较、删除 warnings 导入。

- python/sglang/test/server_fixtures/spec_eagle_fixture.py（模块 测试夹具；类别 test；类型 test-infra；符号 _merged_env_overrides）：测试基础设施改进：新增 _merged_env_overrides 方法，沿 MRO 合并环境覆盖，并修改 setUpClass 使用该方法。
- test/registered/spec/eagle/test_spec_eagle_stress.py（模块 测试（stress）；类别 test；类型 test-coverage）：测试覆盖：所有 stress 测试类添加 SGLANG_ENABLE_STRICT_MEM_CHECK_DURING_BUSY=1，并移除不再需要的覆盖。
- test/registered/spec/eagle/test_spec_eagle_topk.py（模块 测试（topk）；类别 test；类型 test-coverage）：测试覆盖：所有 topk 测试类添加 SGLANG_ENABLE_STRICT_MEM_CHECK_DURING_BUSY=1，并导入 envs。
- test/registered/spec/eagle/test_spec_eagle_page.py（模块 测试（page）；类别 test；类型 test-coverage）：测试覆盖：调整页面大小测试类的 env_overrides，启用 busy 检查并移除 SGLANG_ENABLE_SPEC_V2 覆盖。

关键符号：SchedulerInvariantChecker.self_check_during_busy, SchedulerInvariantChecker._get_total_uncached_sizes, SpecEagleServerBase._merged_env_overrides, SpecEagleServerBase.setUpClass

关键源码片段

python/sglang/srt/managers/scheduler_components/invariant_checker.py

核心逻辑变更：移除 topk>1 早期返回、修复 batch 身份比较、删除 warnings 导入。

```
# python/sglang/srt/managers/scheduler_components/invariant_checker.py

def self_check_during_busy(self):
    if self.get_last_batch() is None:
        return

    # 不再有 speculative_eagle_topk > 1 的早期返回，
    # 因为 _get_total_uncached_sizes 已经通过减去 uncached 量
    # 使不变式在 topk>1 时也能平衡。
    ps = self.pool_stats_observer.get_pool_stats()
    full_uncached, swa_uncached = self._get_total_uncached_sizes()

    full_leak, full_msg = self._check_full_pool(ps, uncached=full_uncached)
    swa_leak, swa_msg = False, ""
    if self.is_hybrid_swa:
        swa_leak, swa_msg = self._check_swa_pool(ps, uncached=swa_uncached)

    level = envs.SGLANG_ENABLE_STRICT_MEM_CHECK_DURING_BUSY.get()
    full_line = f"[Mem Check (BUSY)] {full_msg}"
    swa_line = f"[Mem Check (BUSY)] {swa_msg}" if swa_msg else None

    if level > 1:
        logger.info(full_line)
        if swa_line:
            logger.info(swa_line)
```

```
    elif level == 1:
        self.recent_busy_msgs.append(full_line)
        if swa_line:
            self.recent_busy_msgs.append(swa_line)
        if full_leak or swa_leak:
            for msg in self.recent_busy_msgs:
                logger.info(msg)

    assert not full_leak, f"Full Pool Mem Leak Detected! {full_msg}"
    # SWA 泄漏仅在启用时断言
    if self.is_hybrid_swa:
        assert not swa_leak, f"SWA Pool Mem Leak Detected! {swa_msg}"


def _get_total_uncached_sizes(self) -> Tuple[int, int]:
    """Sum uncached tokens for full and SWA pools across all active batches."""
    last_batch = self.get_last_batch()
    running_batch = self.get_running_batch()
    # 使用 identity (is / is not)，因为 ScheduleBatch 的 dataclass __eq__
    # 会比较 tensor 字段，在 empty tensor 时报 RuntimeError。
    batches = [last_batch]
    if (
        running_batch is not None
        and running_batch is not last_batch
        and not running_batch.is_empty()
    ):
        batches.append(running_batch)
    full_uncached = 0
    swa_uncached = 0
    for batch in batches:
        for req in batch.reqs:
            assert req.kv_committed_freed == req.kv_overallocated_freed
            if req.kv_committed_freed or req.req_pool_idx is None:
                continue
            allocated_len = req.kv_allocated_len
            if self.page_size > 1:
                allocated_len = ceil_align(allocated_len, self.page_size)
                assert req.cache_protected_len % self.page_size == 0
            full_uncached += allocated_len - req.cache_protected_len
            if self.is_hybrid_swa:
                swa_uncached += allocated_len - max(
                    req.cache_protected_len, req.swa_evicted_seqlen
                )
    return full_uncached, swa_uncached
```

python/sglang/test/server_fixtures/spec_eagle_fixture.py

测试基础设施改进：新增 _merged_env_overrides 方法，沿 MRO 合并环境覆盖，并修改 setUpClass 使用该方法。

```
# python/sglang/test/server_fixtures/spec_eagle_fixture.py

class SpecEagleServerBase(CustomTestCase):
    # ... 其他属性 ...

    # env_overrides: (env_var_obj, value) pairs applied only around launch.
    # Declare ONLY this class's own; _merged_env_overrides() unions them down the
    # MRO (base first), so never restate a base's. Derived wins on a repeated env.
    env_overrides = ()

    @classmethod
    def _merged_env_overrides(cls):
        """Base first so a derived class wins for a repeated env var."""
        merged = []
        for klass in reversed(cls.__mro__):
            merged.extend(klass.__dict__.get("env_overrides", ()))
        return merged

    @classmethod
    def setUpClass(cls):
        cls.base_url = DEFAULT_URL_FOR_TEST
        cls.target_model = cls.model
        cls._tokenizer = None
        with contextlib.ExitStack() as stack:
            stack.enter_context(envs.SGLANG_ENABLE_ASYNC_ASSERT.override(True))
            stack.enter_context(
                envs.SGLANG_ALLOW_OVERWRITE_LONGER_CONTEXT_LEN.override(True)
            )
            for env_var, value in cls._merged_env_overrides(): # 使用合并方法
                stack.enter_context(env_var.override(value))
            cls.process = popen_launch_server(
                cls.model,
                cls.base_url,
                timeout=DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
                other_args=cls._launch_args(),
            )
```

评论区精华

虽无 review 评论，但 PR 描述和提交历史记录了关键决策：

- identity vs equality：_get_total_uncached_sizes 中原使用 not in 意图为身份比较，但触发了 ScheduleBatch.__eq__ 的 tensor 比较，改为 is 修复。
- level 选择：从 level 2 降级到 level 1（安静模式），仅在泄漏时 dump 最近日志，避免每次迭代输出。
- env_overrides 合并：引入 _merged_env_overrides 沿 MRO 合并，使测试类职责清晰，基类覆盖自动生效。

- identity vs equality in _get_total_uncached_sizes (correctness): 使用 is 比较，避免调用 __eq__。
- busy check level 选择：安静模式 (design): 采用 level 1 安静模式，提高可读性。
- 环境覆盖合并机制 (design): 继承友好，保持单一事实来源。

风险与影响

- 风险：启用检查可能暴露之前隐藏的 KV 池泄漏，导致 CI 测试失败，但 level 1 安静模式仅在泄漏时输出，降低了日志噪声。身份比较的修复是安全的，因为语义就是对象身份。另外，若未来新增 spec 测试类未继承正确基类或未设置 env_overrides，可能遗漏检查，但已有足够覆盖。
- 影响：对用户无直接影响，但增强了 speculative decoding 路径的内存安全。对系统性能影响极小（仅多一步分配计算）。对团队：CI 现在能在 topk>1 路径上捕获 KV 池泄漏，降低回归风险。测试基础设施改进（_merged_env_overrides）可供后续测试复用。
- 风险标记：核心路径变更，测试覆盖广但可能误报，静默日志可能隐藏细节

关联脉络

- PR #27238 Add quiet mode for busy mem check (level 1: buffer + dump on leak): 本 PR 依赖并扩展了 level 1 安静模式，将其应用于 topk>1 场景。