PR #26806 完整报告

sgl-project/sglang

Add the KV-canary write JIT kernel and reference implementation

合并时间：2026-05-31 09:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26806

执行摘要

- 一句话：新增 KV-canary 写 JIT 内核与参考实现
- 推荐动作：建议精读 write.py 的 WritePlan 数据结构和 launch_canary_write_kernel 的接口设计，注意其对 SWA 的预处理假设（主机端翻译）；CUDA kernel canary_write.cuh 展示了 chain hash 的高效 GPU 实现；测试套件的 invariant 模式值得借鉴。但需关注 review 中未解决的边界检查问题，建议后续 PR 补充。

功能与动机

KV-canary 通过在 KV 缓存中嵌入校验指纹来检测缓存损坏或篡改。本 PR 实现了写内核，负责在处理每个 token 时将 token ID、位置、链式哈希等字段写入指定 canary 槽位，为后续的 verify kernel 提供校验数据。PR body 指出 'The write kernel stamps canary slots into the KV cache'。

实现拆解

1. 定义 WritePlan 数据结构：在 write.py 中新增 WritePlan dataclass，包含 write_offsets（已排除前缀和偏移，int64 数组）、write_seed_slot_indices（每个请求的链种子槽位，int64 数组，-1 表示无前缀）和 write_num_valid_reqs（有效请求数，int32 标量）。提供 allocate 工厂方法和 zero_for_testing_ 辅助方法。
2. 实现 CUDA 写内核：在 canary_write.cuh 中实现 GPU kernel，每个 CUDA block 处理一个请求的写条目。每个线程串行遍历该请求的writeentries，读取nput_ids、positions、out_cache_loc（已由主机端做 SWA 翻译），将字段写入 canary_buf 的指定槽位，并计算链式哈希 splitmix64_mix3。初始 running_prev_hash 从种子槽位加载。
3. 实现 PyTorch 参考实现：在 write_ref.py 中提供完全基于 PyTorch 的 CPU 实现，与 CUDA 内核保持逻辑一致，支持 enable_write_input_assert 调试断言。参考实现用纯 Python 循环模拟 kernel 行为，用于验证 kernel 正确性。
4. 添加 Benchmark 和测试套件：新增 bench_write.py 用于性能基准测试（支持多种 pool_kind、extend_len 等场景）；新增 _invariants.py 定义一系列不变量断言（如偏移单调性、extras 位置等），供 hand 和 fuzz 测试共用；_canary_helpers.py 和 _fixtures.py 提供测试数据生成、假 violation 日志等公共设施。

关键文件：

- python/sglang/jit_kernel/kv_canary/write.py（模块 写内核；类别 source；类型 core-logic；符号 WritePlan, allocate, zero_for_testing_, launch_canary_write_kernel）：核心源码文件，定义了 WritePlan 数据结构和 launch_canary_write_kernel 入口，是 write kernel 的 Python 前端。
- python/sglang/jit_kernel/kv_canary/write_ref.py（模块 参考实现；类别 source；类型 dependency-wiring；符号 launch_canary_write_kernel_torch_reference）：PyTorch 参考实现，与 CUDA kernel 逻辑对等，用于验证与调试。
- python/sglang/jit_kernel/csrc/kv_canary/canary_write.cuh（模块 CUDA 内核；类别 other；类型 dependency-wiring）：CUDA kernel 实现，包含 WriteKernelParams 和核心写入逻辑，是性能关键模块。
- python/sglang/jit_kernel/benchmark/kv_canary/bench_write.py（模块 基准测试；类别 source；类型 core-logic；符号 _write_entry_count, _write_num_slots, _build_write_inputs, _build_context）：性能基准测试，验证 write kernel 在不同场景下的吞吐与延迟。
- python/sglang/jit_kernel/tests/kv_canary/_invariants.py（模块 不变性检查；类别 test；类型 test-coverage；符号 PlanInvariants, assert_all, _assert_write_offsets_monotone, _assert_write_offsets_total_matches_active_extend_sum）：定义核心不变量断言，被 hand 和 fuzz 测试共用，保障 kernel 输出格式正确。
- python/sglang/jit_kernel/tests/kv_canary/_canary_helpers.py（模块 测试辅助；类别 test；类型 test-coverage；符号 FakeViolationLog, allocate, make_canary_buf, make_canary_buf_pair）：提供测试用的辅助函数和 FakeViolationLog，简化测试编写。
- python/sglang/jit_kernel/tests/kv_canary/_fixtures.py（模块 测试夹具；类别 test；类型 test-coverage；符号 make_lut, make_req_to_token, make_padding_mask, derive_plan_capacity）：提供通用的测试 fixture，如 LUT、req_to_token、padding mask 等生成器。

关键符号：WritePlan.allocate, launch_canary_write_kernel, launch_canary_write_kernel_torch_reference, PlanInvariants.assert_all, make_write_plan

关键源码片段

python/sglang/jit_kernel/kv_canary/write.py

核心源码文件，定义了 WritePlan 数据结构和 launch_canary_write_kernel 入口，是 write kernel 的 Python 前端。

```
from __future__ import annotations
from dataclasses import dataclass
from typing import TYPE_CHECKING
import torch
from sglang.jit_kernel.kv_canary.verify import VerifyOrWriteContext

def launch_canary_write_kernel(
    *,
    context: VerifyOrWriteContext,
```

```
    plan: "WritePlan",
    input_ids: torch.Tensor,
    positions: torch.Tensor,
    out_cache_loc: torch.Tensor,
    enable_write_input_assert: bool,
    expected_input_tokens: torch.Tensor | None,
    expected_input_positions: torch.Tensor | None,
) -> None:
    """写入金丝雀指纹到 canary 缓冲区。每个 CUDA block 处理一个请求的写条目，
    串行遍历该请求的 token，计算链式哈希并存储到指定槽位。"""
    ...

@dataclass(frozen=True, slots=True, kw_only=True)
class WritePlan:
    """Write plan 包含写请求的偏移、种子槽位和有效计数。
    SWA 翻译由主机端完成，kernel 直接使用翻译后的槽位索引。"""
    write_offsets: torch.Tensor
    write_seed_slot_indices: torch.Tensor
    write_num_valid_reqs: torch.Tensor

    @classmethod
    def allocate(
        cls, *, write_req_capacity: int, device: torch.device
    ) -> "WritePlan":
        # 参数校验：capacity 必须为正
        if write_req_capacity <= 0:
            raise ValueError(
                f"kv-canary: WritePlan write_req_capacity must be positive, got {write_req_capacity}"
            )
        return cls(
            write_offsets=torch.empty(
                write_req_capacity + 1, dtype=torch.int64, device=device
            ),
            write_seed_slot_indices=torch.empty(
                write_req_capacity, dtype=torch.int64, device=device
            ),
            write_num_valid_reqs=torch.empty(1, dtype=torch.int32, device=device),
        )
```

评论区精华

Reviewer（gemini-code-assist）提出两个核心安全 / 正确性问题：1）GPU 边界检查缺失：CUDA kernel 和 Python 参考实现均未对 canary_buf 的槽位索引做范围检查，可能导致 OOB 内存读写，需在 WriteKernelParams 中添加 num_canary_slots 字段并在读写前校验。2）原子类型不匹配：对 p.kernel_run_counter（int64_t）的 atomicAdd 使用了 unsigned long long，建议改为 long long 避免严格别名违例。这些建议在 PR 合并时未被采纳，仍为未解决安全隐患。

- GPU kernel 边界检查与类型安全 (correctness): PR 已合并，但未在提交中包含这些修改，边界检查与类型安全问题仍悬而未决。

风险与影响

- 风险：GPU OOB 风险（高）：CUDA kernel 在写入 canary_buf 前未检查 slot 是否在有效范围内，若 out_cache_loc 包含非法值可能造成内存破坏，影响整个 KV 缓存可靠性。原子类型安全（中）：atomicAdd 类型不匹配可能引发未定义行为，但 GPU 上实际影响因编译器行为而异。性能：kernel 设计为每请求单线程串行，对短 extend 场景开销可控，长链场景可能成为瓶颈（但链本是串行特性）。兼容性：依赖 TVM FFI 管理 kernel 参数，若接口变更需同步更新。
- 影响：对用户：无直接用户影响，KV-canary 为内部可观测性模块。对系统：提供 KV-canary 写能力，是 verify kernel 的前置依赖，完整启用后可自动检测 KV 缓存异常。对团队：定义了 WritePlan 接口与测试方法（invariant 模式），后续 kernel 开发可复用同一套测试框架。Benchmark 为性能回归提供基线。
- 风险标记：GPU OOB 风险，atomicAdd 类型不匹配

关联脉络

- PR #26807 Add the KV-canary plan JIT kernels: plan JIT kernel 生成 WritePlan 供 write kernel 使用，两者构成 KV-canary 的写入流水线。
- PR #26808 Add the KV-canary core: data layer, MHA KV-pool patcher, and per-forward runner: core 模块集成了 write kernel 的调用，是 write 能力的实际消费者。
- PR #26809 Add the KV-canary install API and forward-path wiring: install API 将 write kernel 接入模型前向路径，实现端到端的 canary 写入。