PR #26817 完整报告

sgl-project/sglang

Add real-data KV verification to the KV-canary

合并时间：2026-05-31 09:58

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26817

执行摘要

- 一句话：新增真实 KV 数据指纹验证到 KV-canary
- 推荐动作：建议深入阅读 RealKvSource 数据类及其工厂函数的设计，理解如何通过 page_size 和 num_bytes_per_token 适配不同 KV 缓存布局。该 PR 是 KV-canary 功能线的关键一环，对了解 SGLang 的缓存可观测性架构有参考价值。

功能与动机

在纯合成金丝雀基础上，增加对模型真实 KV 字节的哈希和验证，从而捕获实际缓存 KV 的损坏，而不仅仅是合成标记的异常。该功能是 KV-canary 体系的关键补充，使故障检测覆盖到真实数据层面。

实现拆解

1. 数据源抽象：在 python/sglang/jit_kernel/kv_canary/verify.py 中定义 RealKvSource 数据类，封装 tensor、page_size、num_bytes_per_token 和 read_bytes，并添加严格的 16 字节对齐校验（__post_init__）；同时新增 RealKvHashMode 枚举（NONE/PARTIAL/ALL）。
2. 源构造工厂与配置：在 python/sglang/srt/kv_canary/pool_patcher/buffer_alloc.py 中添加 resolve_real_kv_read_bytes 根据配置计算读取字节数，以及 make_row_source 和 make_packed_source 将不同布局的 KV 张量转化为 RealKvSource 元组，并对齐校验。
3. CUDA 内核扩展：修改 write/verify CUDA 内核，新增对 field[3] 的 real_kv 哈希存储与验证，基于 splitmix64 算法，支持多 source 折叠。
4. 运行时串联与扰动：通过 CanaryConfig 新增 real_kv_hash_mode，在 endpoint.py 的 _resolve_real_kv_sources 中生成 sources 并传递；perturb/utils.py 新增 flip_random_source_byte_and_log 使扰动可直接作用于真实 KV 源。
5. 测试覆盖：在 test_verify_hand.py 和 test_write_hand.py 中新增 TestRealKvHash 类覆盖所有模式和多 source 折叠场景；更新 test_self_unit_buffer_alloc.py 和 test_self_unit_perturb.py 验证新工具函数。

关键文件：

- python/sglang/jit_kernel/kv_canary/verify.py（模块 内核层；类别 source；类型 core-logic；符号 RealKvSource, post_init, _build_real_kv_source_abi）：核心数据抽象：定义了 RealKvSource 类及其对齐校验逻辑，是后续一切操作的基础。

- python/sglang/srt/kv_canary/pool_patcher/buffer_alloc.py（模块 缓冲区分配；类别 source；类型 dependency-wiring；符号 resolve_real_kv_read_bytes, _clip_read_bytes_aligned, make_row_source, make_packed_source）：源构造工厂：提供 resolve_real_kv_read_bytes、make_row_source 和 make_packed_source，将不同 KV 缓存布局适配为 RealKvSource 元组。
- python/sglang/srt/kv_canary/endpoint.py（模块 端点层；类别 source；类型 core-logic；符号 _resolve_real_kv_sources）：运行时串联：在 endpoint 中调用工厂函数构建 real_kv_sources 并传递至 buffer group。
- python/sglang/srt/kv_canary/perturb/utils.py（模块 扰动工具；类别 source；类型 core-logic；符号 flip_random_source_byte_and_log, flip_first_byte_in_source）：扰动适配：新增 flip_random_source_byte_and_log 等函数，使扰动能作用于真实 KV 源，增强测试覆盖。
- python/sglang/jit_kernel/tests/kv_canary/test_verify_hand.py（模块 验证测试；类别 test；类型 test-coverage；符号 _stamp_clean_kv_chain, test_violation_real_kv_hash_mismatch, TestRealKvHash, test_real_kv_mode_off_yields_zero）：核心验证测试：新增 TestRealKvHash 类，全面测试各 hash 模式、多 source 折叠、对齐错误等场景。
- python/sglang/srt/kv_canary/config.py（模块 配置；类别 source；类型 configuration；符号 real_kv_hash_mode）：配置扩展：新增 real_kv_hash_mode 字段，通过 --kv-canary-real-data 参数控制。

关键符号：RealKvSource.post_init, resolve_real_kv_read_bytes, _clip_read_bytes_aligned, make_row_source, make_packed_source, _resolve_real_kv_sources, _compute_real_kv_hash_scalar, flip_random_source_byte_and_log, flip_first_byte_in_source, launch_canary_write_kernel, launch_canary_verify_kernel

关键源码片段

python/sglang/jit_kernel/kv_canary/verify.py

核心数据抽象：定义了 RealKvSource 类及其对齐校验逻辑，是后续一切操作的基础。

```
# 文件：python/sglang/jit_kernel/kv_canary/verify.py
# 真实 KV 数据源抽象，描述 canary 应读取的实际 KV 字节片段
@dataclass(frozen=True, slots=True, kw_only=True)
class RealKvSource:
    tensor: torch.Tensor # 底层存储，至少 2 维，行内允许有空洞
    page_size: int # 每行打包的 slot 数，>= 1
    num_bytes_per_token: int # 每个 slot 在 dim-1 上的字节数，正 16 倍数
    read_bytes: int # 实际折叠进指纹的字节数，正 16 倍数且 <= num_bytes_per_token

    def __post_init__(self) -> None:
        # 对齐校验：所有关键参数必须为 16 的倍数
        if self.page_size < 1:
            raise ValueError(...)
```

```
        if self.num_bytes_per_token <= 0 or self.num_bytes_per_token % 16 != 0:
            raise ValueError(...)
        if self.read_bytes <= 0 or self.read_bytes > self.num_bytes_per_token or self.read_bytes
        % 16 != 0:
            raise ValueError(...)
        if self.tensor.ndim < 2:
            raise ValueError(...)
        row_stride_bytes = int(self.tensor.shape[1]) * self.tensor.element_size()
        if row_stride_bytes % 16 != 0:
            raise ValueError(...)
```

python/sglang/srt/kv_canary/pool_patcher/buffer_alloc.py

源构造工厂：提供 resolve_real_kv_read_bytes、make_row_source 和 make_packed_source，将不同 KV 缓存布局适配为 RealKvSource 元组。

```
# 文件：python/sglang/srt/kv_canary/pool_patcher/buffer_alloc.py
# 解析配置决定读取字节数
_PARTIAL_REAL_KV_READ_BYTES = 16
_REAL_KV_READ_ALIGN = 16

def resolve_real_kv_read_bytes(config: CanaryConfig) -> int:
    if config.real_kv_hash_mode is RealKvHashMode.NONE:
        return 0
    if config.real_kv_hash_mode is RealKvHashMode.ALL:
        return sys.maxsize # 发送给内核，让其使用完整行长度
    return _PARTIAL_REAL_KV_READ_BYTES

# 裁剪并校验读取字节数，确保 128 位对齐
def _clip_read_bytes_aligned(*, requested: int, num_bytes_per_token: int) -> int:
    # 要求每 token 字节数为 16 的倍数
    if num_bytes_per_token <= 0 or num_bytes_per_token % _REAL_KV_READ_ALIGN != 0:
        raise ValueError(...)
    if requested == 0:
        return 0
    if requested == sys.maxsize:
        return num_bytes_per_token
    if requested < 0 or requested > num_bytes_per_token:
        raise ValueError(...)
    if requested % _REAL_KV_READ_ALIGN != 0:
        raise ValueError(...)
    return requested

# 为 row-major 布局构造 RealKvSource（page_size=1）
def make_row_source(*, layer_buffer: torch.Tensor, read_bytes: int) -> Tuple[RealKvSource, ...]:
    contiguous = layer_buffer.contiguous()
    num_slots = int(contiguous.shape[0])
    if num_slots == 0 or read_bytes == 0:
        return ()
    flat = contiguous.view(torch.uint8).reshape(num_slots, -1)
```

```
    num_bytes_per_token = int(flat.shape[1])
    clipped = _clip_read_bytes_aligned(requested=read_bytes, num_bytes_per_token=num_bytes_
    per_token)
    if clipped == 0:
        return ()
    return (RealKvSource(tensor=flat, page_size=1, num_bytes_per_token=num_bytes_per_token,
    read_bytes=clipped),)
```

评论区精华

本次 PR 无实质性 Review 讨论，作者提交后通过 CI 直接合并。唯一的评论来自 Gemini Code Assist 的每日配额警告，未涉及技术内容。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 性能开销：ALL 模式下需读取全部 KV 字节，可能增加推理延迟，建议非诊断场景保持默认 NONE。
  2. 对齐约束：128 位对齐要求严格，若 KV 缓存布局不符合可能导致内核异常。
  3. 配置依赖：新参数 --kv-canary-real-data 未设置时功能静默，用户可能误以为已启用。
  4. 模块耦合：RealKvSource 抽象与现有 CanaryBufferGroup、endpoint 等紧密耦合，后续改造需同步调整。 - 影响：影响范围局限在 KV-canary 子系统，不影响核心推理路径（默认关闭）。用户可通过新参数启用诊断能力，团队需要维护额外的抽象层和测试套件。设计与代码质量较高，长期可维护性良好。 - 风险标记：性能开销，对齐约束，配置依赖，模块耦合

关联脉络

- PR #26818 Add token-id verification to the KV-canary: 同一 KV-canary 功能线，增加了 token-id 验证，共享框架和测试工具。
- PR #26819 Add the KV-canary perturb modes and PD-disaggregation e2e tests: 本 PR 是扰动模式的前置依赖（PR body 明确说明）。
- PR #26820 Add a sliding-window-attention divergence reporter for the KV-canary: 同为 KV-canary 可观测性增强，共用配置和运行时基础设施。
- PR #26821 Add periodic KV-canary stats logging and kernel-run-counter health check: 延续 KV-canary 健康检查体系，与本 PR 互补。