PR #26813 完整报告

sgl-project/sglang

Support EAGLE speculative decoding in the KV-canary

合并时间：2026-05-31 09:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26813

执行摘要

- 一句话：将 KV-canary 集成到 EAGLE 推测解码 draft 路径
- 推荐动作：值得阅读，特别是 CanaryManager 如何通过多个 SingleForwardManager 支持内部分步，以及如何在 EAGLE worker 的各个 forward 入口插入非侵入式 canary 钩子。该设计模式可复用于其他推测解码算法。

功能与动机

Wire kv-canary into the EAGLE speculative-decoding draft/verify forward paths so the canary write/verify runs during draft steps, in addition to the existing non-EAGLE model-runner integration. (PR body)

实现拆解

1. 传递推测参数：在 python/sglang/srt/kv_canary/api.py 的 install_canary 中，从 server_args 读取 speculative_num_steps，传递给 CanaryManager；根据 model_runner.is_draft_worker 设置 kv_token_id_vs_position_offset（draft worker 为 1，target 为 0）。
2. 分配多个 SingleForwardManager：在 python/sglang/srt/kv_canary/runner/canary_manager.py 的 CanaryManager.__init__ 中，根据 num_sfms = max(1, speculative_num_steps - 1) 创建对应数量的 SingleForwardManager，用于每个内部分步的 canary 状态管理。
3. EAGLE worker 集成：在 python/sglang/srt/speculative/eagle_worker_v2.py 的 draft、draft_forward、_draft_extend_for_prefill、_draft_extend_for_decode 等方法中，将原有的 contextlib.nullcontext() 占位替换为真实的 c.with_ops_outside_graph(...) 和 c.with_active_single_forward_manager(i) 调用，同时导入 context_tuple 以便同时使用多个 context。在 __init__ 末尾，若 canary 已激活则调用 mark_init_finished()。
4. Cuda Graph 捕获集成：在 python/sglang/srt/speculative/eagle_draft_extend_cuda_graph_runner.py 的 cuda graph 捕获路径中，将 nullcontext 替换为 c.with_active_single_forward_manager(0)。
5. 测试覆盖：新增单元测试 test_self_unit_runner_per_forward.py 验证 pre_ops_maybe_inside_graph 在未调用 bracket 时断言失败，以及正确分发到对应 SingleForwardManager；新增端到端测试 test_e2e_spec_eagle.py，使用 mock 模型运行

EAGLE 服务并确保无 canary violation。

关键文件：

- python/sglang/srt/speculative/eagle_worker_v2.py（模块 EAGLE 工作器；类别 source；类型 dependency-wiring；符号 init, draft, draft_forward, _draft_extend_for_prefill）：核心集成文件，在多个 draft 入口替换 canary context 占位，并增加初始化标记。
- python/sglang/srt/kv_canary/api.py（模块 Canary API；类别 source；类型 entrypoint；符号 install_canary）：入口文件，传递 spec 参数和 offset，控制 canary 的安装。
- python/sglang/srt/kv_canary/runner/canary_manager.py（模块 Canary 管理器；类别 source；类型 core-logic；符号 CanaryManager.init）：核心管理器，根据 speculative_num_steps 分配多个 SingleForwardManager。
- test/registered/kv_canary/test_self_unit_runner_per_forward.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestCanaryManagerActiveSingleForwardManagerDispatch, test_pre_ops_maybe_inside_graph_dispatches_to_bracketed_sfm, _record, test_pre_ops_maybe_inside_graph_asserts_outside_bracket）：新增单元测试，验证 pre_ops_maybe_inside_graph 的路由行为。
- test/registered/mock_model/test_e2e_spec_eagle.py（模块 端到端测试；类别 test；类型 test-coverage；符号 TestE2ESpeculativeEagle, test_spec_eagle_no_canary_violation）：新增端到端测试，使用 mock 模型验证 EAGLE 与 canary 集成时无 violation。
- python/sglang/srt/speculative/eagle_draft_extend_cuda_graph_runner.py（模块 Cuda Graph 扩展；类别 source；类型 core-logic；符号 EAGLEDraftExtendCudaGraphRunner.run_once）：在 cuda graph 捕获中集成 canary active context。

关键符号：install_canary, CanaryManager.init, EagleWorker.init, EagleWorker.draft, EagleWorker.draft_forward, EagleWorker._draft_extend_for_prefill, EagleWorker._draft_extend_for_decode, EAGLEDraftExtendCudaGraphRunner.run_once

关键源码片段

python/sglang/srt/speculative/eagle_worker_v2.py

核心集成文件，在多个 draft 入口替换 canary context 占位，并增加初始化标记。

```
# python/sglang/srt/speculative/eagle_worker_v2.py (partial)

from sglang.srt.kv_canary.runner.canary_manager import context_tuple

class EagleWorker:
    def __init__(self, ...):
        # ... existing init code ...
        self.init_cuda_graphs()
        # 若 canary 已激活，标记初始化完成，以便单步 forward 管理就绪
        if (c := self.draft_runner.canary_manager) is not None:
            c.mark_init_finished()
```

```
def draft(self, batch: ScheduleBatch):
    n_inner = self.speculative_num_steps - 1
    # 创建 canary 外部 context，包裹所有 draft 步骤
    canary_outside_ctx = (
        c.with_ops_outside_graph(
            single_forward_indices=list(range(n_inner)),
            maybe_inaccurate_forward_batch=forward_batch,
        )
        if (c := self.draft_runner.canary_manager) is not None
        else contextlib.nullcontext()
    )
    with canary_outside_ctx:
        # ... run draft ...

def draft_forward(self, forward_batch: ForwardBatch):
    for i in range(...):
        # 每个 draft step 使用对应索引的 active context
        canary_index_ctx = (
            c.with_active_single_forward_manager(i)
            if (c := self.draft_runner.canary_manager) is not None
            else contextlib.nullcontext()
        )
        with forward_context(...), canary_index_ctx:
            # 执行单个 draft forward
            ...

def _draft_extend_for_prefill(self, ...):
    # 使用 context_tuple 同时包裹外部和内部 context
    canary_ctx = (
        context_tuple(
            c.with_ops_outside_graph(
                single_forward_indices=[0],
                maybe_inaccurate_forward_batch=forward_batch,
            ),
            c.with_active_single_forward_manager(0),
        )
        if (c := self.draft_runner.canary_manager) is not None
        else contextlib.nullcontext()
    )
    with canary_ctx:
        logits_output = self.draft_runner.forward(forward_batch).logits_output

def _draft_extend_for_decode(self, ...):
    # 类似替换
    canary_ctx = (
        context_tuple(
            c.with_ops_outside_graph(
                single_forward_indices=[0],
                maybe_inaccurate_forward_batch=forward_batch,
```

```
                ),
                c.with_active_single_forward_manager(0),
            )
            if (c := self.draft_runner.canary_manager) is not None
            else contextlib.nullcontext()
        )
        with canary_ctx:
            if can_cuda_graph:
                ...
```

python/sglang/srt/kv_canary/api.py

入口文件，传递 spec 参数和 offset，控制 canary 的安装。

```
# python/sglang/srt/kv_canary/api.py (partial)

def install_canary(
    *,
    server_args: "ServerArgs",
    model_runner: "ModelRunner",
) -> Optional[CanaryManager]:
    # ... config detection ...
    device = torch.device(model_runner.device)
    # EAGLE draft worker 中 pools 旋转 input_ids 导致 slot p 存储 token p+1 的 K/V,
    # target pools 无此偏移。此偏移在 plan 侧 expected-token gather kernel 中使用。
    kv_token_id_vs_position_offset = 1 if model_runner.is_draft_worker else 0
    buffer_groups = attach_canary_buffers(
        pool=model_runner.token_to_kv_pool,
        config=config,
        device=device,
        kv_token_id_vs_position_offset=kv_token_id_vs_position_offset, # 传入偏移
    )
    # 从 server_args 读取 speculative_num_steps，缺省为 1（非推测解码场景）
    speculative_num_steps = int(server_args.speculative_num_steps or 1)
    manager = CanaryManager(
        ...
        speculative_num_steps=speculative_num_steps, # 传递给管理器
    )
    # ...
    return manager
```

评论区精华

No review discussions were available for this PR.

- 暂无高价值评论线程

风险与影响

- 风险：主要风险： 1) 多步 draft 中 SingleForwardManager 索引管理错误可能导致 use-after-free 或状态污染； 2) 若 EAGLE 配置变化（如 speculative_num_steps 动态调整），当前不支持动态调整，可能创建过多或过少； 3) cuda graph 捕获时若 canary 状态未正确初始化可能导致 graph 失效； 4) draft worker 的 token 偏移需要与 kernel 一致，不匹配可能导致指纹验证失败。但这些风险通过默认关闭（canary mode=NONE）隔离。
  - 影响：对用户：启用 KV-canary 后，EAGLE 推测解码的 KV 缓存将受到有效性验证，提升可靠性；性能影响仅在 canary 开启时存在。对系统：增加了 canary 与推测解码的耦合，但通过条件判断保持零开销关闭。对团队：为 future canary 功能（如扰动、PD 测试）在推测解码场景落地奠定了基础。
  - 风险标记：多步 draft 索引管理，偏移配置一致性，cuda graph 捕获状态，动态步骤数未支持

关联脉络

- PR #26814 Add rids/bootstrap-room int-hash plumbing for deterministic per-request identification: 为 canary 提供请求 ID 哈希基础，本 PR 的 EAGLE 集成依赖该确定性标识。
- PR #26818 Add token-id verification to the KV-canary: 增加 token-id 验证功能，本 PR 在 EAGLE draft 中同样使用 token-id 验证，并处理 draft 偏移。
- PR #26820 Add a sliding-window-attention divergence reporter for the KV-canary: 增加 SWA 发散报告能力，本 PR 的 EAGLE 路径也集成了 SWA 上下文。
- PR #26821 Add periodic KV-canary stats logging and kernel-run-counter health check: 同属 kv-canary 可观测性功能线，本 PR 的集成受益于统计日志和健康检查。