PR #52311 完整报告

vllm-project/vllm

[Bugfix][Model Runner V2][Spec Decode] Fix off-by-one in bad_words draft-prefix matching

合并时间：2026-08-16 17:02

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52311

执行摘要

- 一句话：修复 bad_words 内核 spec-decode 前缀读偏一格，补首个单测
- 推荐动作：值得精读。虽然改动只有一行，但根因分析展示了同一内核族内索引约定一致性检查的价值：_penalties_kernel、_load_effective_token 与 _bad_words_kernel 对 input_ids 局部布局的处理必须对齐。测试设计（四轴覆盖：跨边界、draft 内命中、边界不重复计数、纯 committed 基线）与 2 万场景交叉验证方法都值得借鉴；合并前将硬编码槽位改为动态推导也是应对内部状态分配耦合的规范做法。

功能与动机

PR body 明确指出：sampler 传入的 input_ids 是按 logits_indices 收集的局部布局，对 spec-decode 请求，局部位置 0 是最后一个 committed token，局部位置 j 是第 j - 1 个 draft token；兄弟内核 _penalties_kernel 使用 start_idx + prev_pos + 1，thinking budget 的 _load_effective_token 也使用 + 1 并附显式注释，唯独 _bad_words_kernel 读了 input_ids_ptr + cur_req_first_pos + spec_offset 而缺少 + 1，导致每个 draft 区读取整体前移一格、边界 token 被读取两次。后果是：前缀在 draft 区结束的匹配被漏掉（被禁 token 可能被采出 / 接受）、跨边界前缀晚一行生效、重复读边界 token 可能误禁无辜 token。作者还说明已检索 open PR/issue，确认这不是重复修复（最近的 #34213 只涉及 penalties）。

实现拆解

1. 根因定位：对比同一采样内核族中 _penalties_kernel 与 thinking budget _load_effective_token 的索引约定，确认 _bad_words_kernel 在 from_spec_input 分支读取 input_ids 时漏掉局部布局的 + 1 偏移，问题自 #33433 引入且仅影响 spec-decode 且前缀伸入 draft 区的多 token bad word 场景。
2. 单行修复：在 vllm/v1/worker/gpu/sample/bad_words.py 中将 tl.load(input_ids_ptr + cur_req_first_pos + spec_offset) 改为 + spec_offset + 1，并补充两行中文注释说明 input_ids 的局部布局约定。单 token bad word 与非 spec 行不受影响；Model Runner V1 有独立实现，也不受影响。
3. 测试配套：新增 tests/v1/worker/test_gpu_bad_words.py，这是 _bad_words_kernel 的首个单元测试。场景为 committed 输出 [10, 11]、draft tokens [12, 13]，四个用例分别覆盖：跨边界前缀在完成行屏蔽（核心回归）、draft 区内部命中、边界 token 不重复计数、纯 committed 基线；断言比较完整 logits 张量。

4. Review 反馈修正：Copilot 指出测试硬编码 req_idx = 3 会依赖 RequestState.add_request() 的槽位分配实现（free_indices.pop()），作者在提交 53b989f39 中改为从 req_states.req_id_to_index["req"] 动态推导并贯穿 _make_state/_apply。
5. 验证与交叉校验：A100 上修复后 4 个用例全过；将父提交的内核换回后 3 个 spec 分支用例失败、纯 committed 基线通过；另把内核标量循环转写为 Python，与独立参考 matcher 在 2 万随机场景对比，带 + 1 全部一致，不带时约 24% 场景发散。

关键文件：

- vllm/v1/worker/gpu/sample/bad_words.py（模块 采样器；类别 source；类型 core-logic；符号 _bad_words_kernel）：核心修复文件。_bad_words_kernel 的 spec-decode 分支读取 draft token 时补上 + 1 偏移，与 _penalties_kernel、thinking budget _load_effective_token 的局部布局约定对齐，是整个 PR 的价值所在。
- tests/v1/worker/test_gpu_bad_words.py（模块 采样器；类别 test；类型 test-coverage；符号 _make_state, _apply, test_v2_bad_words_prefix_inside_draft_tokens, test_v2_bad_words_prefix_spanning_committed_and_draft_tokens）：新增 _bad_words_kernel 的首个单元测试文件，构造 committed [10, 11] + drafts [12, 13] 场景，用 4 个用例钉住跨边界、draft 内、边界不重复计数、纯 committed 四条覆盖轴，并采用完整 logits 张量断言；合并前按 Copilot 建议改为动态推导请求索引。

关键符号：_bad_words_kernel, apply_bad_words, BadWordsState, _make_state, _apply, test_v2_bad_words_prefix_spanning_committed_and_draft_tokens, test_v2_bad_words_no_spurious_match_from_last_committed_token

关键源码片段

vllm/v1/worker/gpu/sample/bad_words.py

核心修复文件。_bad_words_kernel 的 spec-decode 分支读取 draft token 时补上 + 1 偏移，与 _penalties_kernel、thinking budget _load_effective_token 的局部布局约定对齐，是整个 PR 的价值所在。

```
# _bad_words_kernel 的 spec-decode 分支（核心匹配逻辑）
# 背景：sampler 按 logits_indices 收集 input_ids，对 spec-decode 请求，
# 局部位置 0 是最后一个 committed token，局部位置 j 是第 j - 1 个 draft token。
# 兄弟内核 _penalties_kernel 与 thinking budget 的 _load_effective_token
# 均按此约定加 1，本内核修复前漏掉了该偏移。

    pos = tl.load(expanded_local_pos_ptr + token_idx)
    cur_req_first_pos = token_idx - pos

    prompt_len = tl.load(prompt_len_ptr + req_state_idx)
    total_len = tl.load(total_len_ptr + req_state_idx)
    output_len = total_len - prompt_len # 已 committed 的 output token 数
    effective_len = output_len + pos # 当前行对应的累计可见长度

    start = tl.load(bd_offsets_base + bw_idx)
```

```
    end = tl.load(bd_offsets_base + bw_idx + 1)
    bad_word_len = end - start
    prefix_len = bad_word_len - 1 # 除最后一个 token 外的前缀长度

    if prefix_len > effective_len:
        return # 前缀比当前可见历史还长，不可能命中

    last_token = tl.load(bd_tokens_base + end - 1)
    match = 1
    for i in range(prefix_len):
        expected = tl.load(bd_tokens_base + start + i)
        actual_pos = effective_len - prefix_len + i

        from_spec_input = actual_pos >= output_len
        if from_spec_input:
            # 前缀伸入 draft 区：读取 sampler 传入的 input_ids。
            # 修复前直接读 spec_offset，会把每个草稿位整体前移一格，
            # 重复读取边界 token，导致跨边界前缀晚一行生效或漏匹配；
            # 加 1 后与局部布局约定对齐，前缀在完成行即正确屏蔽。
            spec_offset = actual_pos - output_len
            actual = tl.load(input_ids_ptr + cur_req_first_pos + spec_offset + 1)
        else:
            actual = tl.load(output_base + actual_pos)

        match = match & (expected == actual)

    if match:
        # 前缀完整命中，屏蔽该 bad word 的最后一个 token
        tl.store(logits_ptr + token_idx * logits_stride + last_token, -float("inf"))
```

tests/v1/worker/test_gpu_bad_words.py

新增 _bad_words_kernel 的首个单元测试文件，构造 committed [10, 11] + drafts [12, 13] 场景，用 4 个用例钉住跨边界、draft 内、边界不重复计数、纯 committed 四条覆盖轴，并采用完整 logits 张量断言；合并前按 Copilot 建议改为动态推导请求索引。

```
# 测试场景：committed 输出 [10, 11]，draft tokens [12, 13]
# sampler 传入的 input_ids 为局部布局 [11, 12, 13]，LOCAL_POS 对应 [0, 1, 2]
PROMPT_LEN = 1
COMMITTED = [5, 10, 11]
INPUT_IDS = [11, 12, 13]
LOCAL_POS = [0, 1, 2]


def _apply(bad_words_token_ids: list[list[int]]) -> torch.Tensor:
    # 从 RequestState 反向解析请求索引，避免硬编码槽位使测试
    # 依赖 add_request() 的内部分配顺序（free_indices.pop()）
    state, req_idx = _make_state(bad_words_token_ids)
    num_logits = len(INPUT_IDS)
    logits = torch.zeros((num_logits, VOCAB_SIZE), device=DEVICE)
```

```
    idx_mapping_np = np.array([req_idx], dtype=np.intp)
    expanded_idx_mapping = torch.tensor(
        [req_idx] * num_logits, dtype=torch.int32, device=DEVICE
    )
    state.apply_bad_words(
        logits,
        expanded_idx_mapping,
        idx_mapping_np,
        torch.tensor(INPUT_IDS, dtype=torch.int32, device=DEVICE),
        torch.tensor(LOCAL_POS, dtype=torch.int32, device=DEVICE),
    )
    return logits.cpu()


def test_v2_bad_words_prefix_spanning_committed_and_draft_tokens():
    # 核心回归：前缀 [11, 12, 30] 跨 committed/draft 边界，
    # 必须在完成前缀的第 1 行屏蔽，而不是晚一行
    out = _apply([[11, 12, 30]])
    expected = torch.zeros_like(out)
    expected[1, 30] = -float("inf")
    torch.testing.assert_close(out, expected)


def test_v2_bad_words_no_spurious_match_from_last_committed_token():
    # 边界 token 不得被重复计数为第一个 draft token:
    # 输出 [10, 11] + 草稿 [12, 13] 中不应出现 [11, 11]
    out = _apply([[11, 11, 50]])
    expected = torch.zeros_like(out)
    torch.testing.assert_close(out, expected)
```

评论区精华

核心讨论集中在测试质量与 CI 状态两方面：Copilot 指出测试硬编码请求槽位 `req_idx = 3`，依赖 `RequestState.add_request()` 当前的 `free_indices.pop()` 分配行为，槽位分配顺序变化时测试会失效，要求从真实状态反向推导；作者回应已在 53b989f39 中通过 `req_states.req_id_to_index["req"]` 动态推导并覆盖两处标记位置。CI 方面，jyan-R 主动说明 Multi-Modal Models job 两次失败均与 PR 无关（gemma4 warmup 触发 PyTorch 内部 NVML assert，见 #52403），并指出 cpu-language-generation-and-pooling-model-tests 曾因 agent 未接管而 Expired，覆盖本 PR 区域的 v1-sample-plus-logits、v1-others-cpu 均为绿色；最终 njhill 批准合并。

- 测试硬编码请求槽位 req_idx=3 导致脆弱 (testing): 作者在 53b989f39 中改为通过 req_states.req_id_to_index["req"] 动态推导请求索引，并贯穿 _make_state / _apply，覆盖 Copilot 标记的两处位置。
- CI 中 whisper job 失败与 PR 无关 (other): 未继续重试避免浪费 CI；infra 问题已提交 #52403 跟踪，最终合并时相关测试为绿色，njhill 批准。

风险与影响

- 风险：修复面很窄：仅当 spec-decode 开启、bad_word 前缀长度 >= 2 且前缀伸入 draft 区时才受影响，单 token bad word 与非 spec 行均无行为变化，因此回归风险低。但需注意两点：一是 + 1 偏移强依赖 sampler 传入 input_ids 的局部布局约定，后续 #34213 若改动 expanded_local_pos 批次状态或 sampler 的 gather 方式，该假设可能失效，需要回归本测试；二是该内核此前完全没有测试覆盖，MR V1 与 MR V2 的 bad_words 语义一致性尚未被任何测试锁定，未来重构时应补充跨实现对比。另外该内核为 Triton 核函数，改动会触发重新编译，但无性能影响。
- 影响：用户影响：此前在 spec-decode + bad_words 组合下，被禁 token 可能被采出或被接受，属于内容安全 / 合规风险；修复后跨边界前缀会在正确位置屏蔽，且不再误禁无辜 token。系统影响：仅触及 MR V2 采样器路径的 bad_words 屏蔽逻辑，单行偏移调整无性能开销，MR V1 与推理主干行为不受影响。团队影响：为 _bad_words_kernel 建立了首个可复用的 GPU 单测基座，验证方法（内核标量循环转写 + 独立参考 matcher + 随机场景交叉验证）对后续采样内核维护有直接参考价值。
- 风险标记：采样核心路径变更，Triton 内核首次测试覆盖，依赖 input_ids 局部布局约定，MR V1/V2 语义一致性未锁定

关联脉络

- PR #33433 [Model Runner V2] support bad_words sampling param: 引入 _bad_words_kernel 的原始 PR，本 PR 修复的就是该内核从引入起就存在的 off-by-one。
- PR #34213 [ModelRunner V2] Simplify penalties implementation: 同族采样内核的相邻改动（避免 expanded_local_pos 批次状态），与本 PR 的 input_ids 局部布局约定直接相关，作者明确说明该 PR 只涉及 penalties、不重复。
- PR #49613 [Bugfix][Sampling] Clear empty side on thinking-budget asymmetric SWAP: 同一采样链路的 thinking budget 状态修复，其 _load_effective_token 的 + 1 约定与本 PR 修复的偏移同源。
- PR #52419 [Bugfix][Spec Decode] Keep EAGLE cache registration on the partial-hash-hit path: 同属 spec-decode 路径的正确性修复，反映该功能线近期持续收敛采样与缓存类边界 bug。