# PR #39833 完整报告

vllm-project/vllm

[MRv2]fix: model accuracy regression caused by reusing the stale last_sampled_tokens and draft_tokens

合并时间：2026-04-22 00:30

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/39833

## 执行摘要

- 一句话：修复 V2 模型运行器复用请求槽时的模型精度回归
- 推荐动作：建议精读本 PR。这是一个典型的状态泄漏 bug 修复，展示了在多请求并发模型中正确管理共享状态的技巧。尤其值得关注 njhill 建议的无需同步的切片赋值手法，以及通过条件判断避免不必要写入的思路。

## 功能与动机

修复 PD 分离部署中由过期状态泄漏导致的显著模型精度回归。PR body 指出 GSM8K 准确率从 V1 基线的 0.93 降至 V2 修复前的 0.82，回归幅度达 13%。根因是 combine_sampled_and_draft_tokens 内核在解码首步读取了上一请求留下的 last_sampled_tokens。

## 实现拆解

1. 在 vllm/v1/worker/gpu/states.py 的 add_request() 方法中新增两行初始化逻辑：当 num_computed_tokens > 0（表示请求已预填充完成，将直接进入解码步）时，将 last_sampled_tokens[req_idx:req_idx+1] 赋值为最后一个已计算 token 的 ID；同时无条件将 draft_tokens[req_idx] 置零。
2. 避免 CPU-GPU 同步：采用 self.last_sampled_tokens[req_idx : req_idx + 1] = all_token_ids[num_computed_tokens - 1] 的切片赋值技巧（由 reviewer njhill 提出），该写法利用 PyTorch 内部行为绕过显式同步。
3. 条件限制：对于 num_computed_tokens == 0 的新预填充请求，跳过 last_sampled_tokens 的写入，因为此时 combine_sampled_and_draft_tokens 不会读取该值。draft_tokens 的置零无副作用，故无条件执行。
4. 无测试文件变更：PR 仅修改源码文件，未添加单元测试或集成测试。

关键文件：

- vllm/v1/worker/gpu/states.py（模块 模型运行器；类别 source；类型 core-logic；符号 add_request）：核心修复文件，在 add_request 方法中为复用槽初始化 last_sampled_tokens 和 draft_tokens，防止过期状态泄漏。

关键符号：add_request

## 关键源码片段

### vllm/v1/worker/gpu/states.py

核心修复文件，在 add_request 方法中为复用槽初始化 last_sampled_tokens 和 draft_tokens，防止过期状态泄漏。

```
# vllm/v1/worker/gpu/states.py (add_request 方法，新增 12 行 )

def add_request(
    self,
    req_id: str,
    prompt_len: int,
    all_token_ids: list[int],
    num_computed_tokens: int,
) -> None:
    assert len(self.free_indices) > 0, "No free indices"
    req_idx = self.free_indices.pop()
    self.req_id_to_index[req_id] = req_idx
    self.index_to_req_id[req_idx] = req_id

    self.prompt_len.np[req_idx] = prompt_len
    prefill_len = len(all_token_ids)
    assert prefill_len >= prompt_len, (
        f"prefill_len {prefill_len} < prompt_len {prompt_len}"
    )
    self.prefill_len.np[req_idx] = prefill_len
    self.total_len.stage_write_elem(req_idx, prefill_len)
    self.all_token_ids.stage_write(req_idx, 0, all_token_ids)
    self.num_computed_prefill_tokens[req_idx] = num_computed_tokens
    self.num_computed_tokens.stage_write_elem(req_idx, num_computed_tokens)

    # --- 新增修复开始 ---
    if num_computed_tokens > 0 and num_computed_tokens <= prefill_len:
        # 对于 PD disagg 或恢复的请求 ( 已预填充完成 ):
        # 将 last_sampled_tokens 设为最后一个已计算 token 的 ID,
        # 使得首个解码步能获得正确的 input_id。
        # 对于新预填充请求 (num_computed_tokens == 0)，该张量不会被
        # combine_sampled_and_draft_tokens 读取，所以跳过写入。
        # 使用切片赋值而非标量索引，以通过 fill_ 分发写入,
        # 避免 host/device 同步 ( 来自 PyTorch 内部行为 )。
        self.last_sampled_tokens[req_idx : req_idx + 1] = all_token_ids[
            num_computed_tokens - 1
        ]
    # 清零 draft_tokens，防止过期的 draft token 影响当前请求
    self.draft_tokens[req_idx].zero_()
    # --- 新增修复结束 ---

    # 原后续方法 ...
    def apply_staged_writes(self) -> None:
```

```
        self.prompt_len.copy_to_uva()
        # ...
```

## 评论区精华

1. CPU-GPU 同步问题：njhill 指出原始赋值可能引入同步开销，并建议使用切片赋值技巧 `self.last_sampled_tokens[req_idx:req_idx + 1] = all_token_ids[num_computed_tokens - 1]`，该写法避免了同步。
2. 条件判断范围：njhill 建议将条件从 `num_computed_tokens > 0 and num_computed_tokens <= prefill_len` 改为仅 `num_computed_tokens > 0`（已采纳），并指出完成预填充的请求（`num_computed_tokens` 已存在）无需额外判断。
3. 新请求 vs 槽复用场景：njhill 确认对于 `num_computed_tokens == 0` 的新请求，`combine_sampled_and_draft_tokens` 不会读取 `last_sampled_tokens`，因此可安全跳过写入。
4. next_prefill_tokens 初始化：gemini-code-assist 建议同时清零 `next_prefill_tokens`，但 liuzijing2014 解释该值仅在分块预填充路径中由 `prepare_prefill_inputs` 写入，不会在首次解码步被读取，因此无需处理。
5. 批量化优化建议：njhill 提出若每步有大量此类请求，可考虑在 `add_request` 中收集 token ID，在 `apply_staged_writes` 中通过异步拷贝和 scatter 批量写入，但目前不必要。

- CPU-GPU 同步优化 (performance): 已采纳。切片赋值利用 PyTorch 内部行为避免同步，并添加注释说明。
- 条件判断范围 (correctness): 条件优化为仅在 `num_computed_tokens > 0` 时写入 `last_sampled_tokens`，跳过新请求路径。
- next_prefill_tokens 初始化必要性 (correctness): 未采纳。PR 作者分析正确，`next_prefill_tokens` 在相关路径中总是先写入后读取。
- 批量化写入优化建议 (performance): 当前未实现，但作为可选的未来优化方向。

## 风险与影响

- 风险：风险较低。变更仅影响 `add_request` 方法，且仅当 `num_computed_tokens > 0` 时才写入 `last_sampled_tokens`；`draft_tokens` 无条件清零，对正常预填充路径无影响。主要风险在于切片赋值假设 PyTorch 内部行为可靠，若未来版本行为变化可能引入同步开销（但目前已验证）。无测试文件覆盖，但 PR 作者已通过全量 GSM8K 评估验证正确性。
- 影响：直接影响：修复 PD 分离部署中 V2 模型运行器的精度回归，GSM8K 准确率从 0.82 恢复至 0.94。影响范围：仅影响 V2 模型运行器（v1/v2 分离架构中的新版），且仅在使用请求状态槽复用的场景下生效。用户影响：使用 PD 分离部署（特别是 V2 后端）的用户将获得正确的模型输出。系统影响：无性能负效应，因为初始化逻辑在 CPU 上执行，且通过切片赋值避免了同步开销。
- 风险标记：核心路径变更，无新增测试覆盖，依赖 PyTorch 内部行为

## 关联脉络

- PR #42651 [Perf] Optimize CutlassFP8ScaledMMLinearKernel when padding needed by pre-weight processing, 13.5% TTFT improvement: 同属于性能修复类 PR，且与模型推理路径相关。本 PR 修复精度回归，该 PR 提升性能，两者在模型运行器层面有间接关联。