PR #42202 完整报告

vllm-project/vllm

[Model Runner V2] Fix `seq_lens_cpu_upper_bound`

合并时间：2026-05-12 01:37

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42202

执行摘要

- 一句话：修复 MTP 模式下 CPU mirror 的 computed tokens 发散问题
- 推荐动作：建议精读 update_requests 和 is_prefilling 的改动逻辑，理解 computed tokens 状态同步方式；可关注后续是否添加性能优化。

功能与动机

跟随 PR #40654 之后，发现 num_computed_tokens_np 在每个 step 仅通过 num_scheduled_tokens 递增，在 MTP 场景下会无限发散。正确的做法是每个 step 从 scheduler 拿调整后的值直接刷新，而非在 model runner 侧递增。

实现拆解

1. 修改 update_requests 方法（model_runner.py）：将原来仅追加新 block 的逻辑扩展为同时从 scheduler 的 reqs.num_computed_tokens 直接覆写 self.req_states.num_computed_tokens_np[req_index]，消除了自增导致的发散。
2. 移除 postprocess/postprocess_pool 中的增量逻辑：删除原先在 postprocess 和 postprocess_pool 中对 num_computed_prefill_tokens 和 num_computed_tokens_np 的 += 操作，改为在 update_requests 中统一通过 np.minimum 同步计算 num_computed_prefill_tokens。
3. 重命名 any_prefills 为 is_prefilling（states.py）：将 ModelState.any_prefills 方法改为 is_prefilling，返回逐请求的 bool 数组而非标量 np.any，便于调用方直接使用。同时利用 Python 链式比较简化 add_request 中的条件判断。
4. 调用侧适配（model_runner.py）：在 prepare_inputs 中将内联的 is_prefilling 计算替换为 self.req_states.is_prefilling(idx_mapping_np) 调用，复用新方法。

关键文件：

- vllm/v1/worker/gpu/model_runner.py（模块 模型运行器；类别 source；类型 core-logic；符号 update_requests, prepare_inputs, postprocess, postprocess_pool）：核心变更文件：修改了 update_requests 和 prepare_inputs 方法，将 computed tokens 的刷新从 postprocess 移到 update_requests，并删除了 postprocess 中的增量逻辑。
- vllm/v1/worker/gpu/states.py（模块 状态管理；类别 source；类型 refactor；符号 is_prefilling, add_request）：重命名 any_prefills 为 is_prefilling，并改变返回值为逐请求数组而非聚合布尔值，简化调用方逻辑。

关键符号：update_requests, prepare_inputs, is_prefilling, add_request

关键源码片段

vllm/v1/worker/gpu/model_runner.py

核心变更文件：修改了 update_requests 和 prepare_inputs 方法，将 computed tokens 的刷新从 postprocess 移到 update_requests，并删除了 postprocess 中的增量逻辑。

```
# vllm/v1/worker/gpu/model_runner.py

def update_requests(self, scheduler_output: SchedulerOutput) -> None:
    # 更新现有请求的 block 和 num_computed_tokens
    reqs = scheduler_output.scheduled_cached_reqs
    num_computed_tokens_np = self.req_states.num_computed_tokens_np
    for req_id, num_computed_tokens, req_new_block_ids in zip(
        reqs.req_ids, reqs.num_computed_tokens, reqs.new_block_ids
    ):
        req_index = self.req_states.req_id_to_index[req_id]
        # 关键修复：直接用 scheduler 的值刷新，避免自增发散
        num_computed_tokens_np[req_index] = num_computed_tokens
        if req_new_block_ids is not None:
            self.block_tables.append_block_ids(
                req_index, req_new_block_ids, overwrite=False
            )

    # 同步更新 computed prefill tokens（取 min 限制不超过 prefill_len）
    np.minimum(
        self.req_states.num_computed_tokens_np,
        self.req_states.prefill_len.np,
        out=self.req_states.num_computed_prefill_tokens,
    )
```

vllm/v1/worker/gpu/states.py

重命名 any_prefills 为 is_prefilling，并改变返回值为逐请求数组而非聚合布尔值，简化调用方逻辑。

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

def is_prefilling(self, idx_mapping_np: np.ndarray) -> np.ndarray:
    # 返回逐请求 bool 数组，比原 any_prefills 的标量更灵活
    return (
        self.num_computed_prefill_tokens[idx_mapping_np]
        < self.prefill_len.np[idx_mapping_np]
    )

def add_request(self, req_id, prompt_len, all_token_ids, num_computed_tokens):
    # ... 其他逻辑 ...
    # 使用 Python 链式比较简化条件
    if 0 < num_computed_tokens <= prefill_len:
```

```
        # 设置 last_sampled_tokens
        self.last_sampled_tokens[req_idx : req_idx + 1] = all_token_ids[
            num_computed_tokens - 1
        ]
    # 移除重复的 num_computed_tokens_np 赋值（原先有两行）
```

评论区精华

Review 中 gemini-code-assist[bot] 指出新的 update_requests 循环中每次请求都要做字典查找和标量赋值，在大量缓存请求时可能成为性能瓶颈，建议未来矢量优化。WoosukKwon 最终 approve。

- update_requests 循环性能 (performance): 当前实现正确修复了发散问题，未来可考虑矢量优化。

风险与影响

- 风险：核心路径（每次 step 都会执行的 update_requests 和 prepare_inputs）变更，存在回归风险：如果 scheduler 传回的 num_computed_tokens 与 model runner 预期不一致，可能导致 computed tokens 错误。当前未包含测试文件，缺少覆盖此逻辑的单元测试或集成测试。
- 影响：影响范围集中在 v1 model runner 的请求状态管理，尤其对 MTP 和 PD disagg 场景有修复作用。用户无感知，但稳定性和正确性提升。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #40654 前置 PR（未在历史中列出，但 body 提及）：本 PR 是 PR#40654 的 follow-on，修复其引入的 computed tokens 发散问题。