

PR #41133 完整报告

vllm-project/vllm

[Bugfix] Fix token loss in PP mode which causes degraded accuracy

合并时间: 2026-05-07 02:07

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/41133>

执行摘要

- 一句话: 修复 PP 模式 token 丢失导致精度下降
- 推荐动作: 建议阅读者重点关注 `_update_states` 中 token 索引的设计, 以及当数据依赖跨越多个函数 (如 `_update_states` 和 `condense`) 时, 如何确保不变性。该 PR 的 bug 分析方法论 (从精度退化到具体代码行定位) 值得团队借鉴。

功能与动机

PR 指出当前 PP 模式在并发请求下导致精度退化 (GSM8K strict-match 从 0.8741 降至 0.8324), 根因是 `_update_states` 中 `num_tokens_no_spec` 被错误地赋值为 `num_computed_tokens`, 对于 chunked prefill, `num_computed_tokens` 小于实际 prompt token 数, 导致 `condense()` 时仅复制部分 token, 其余丢失。该问题影响所有 PP 模式用户。

实现拆解

1. 修正 `_update_states` token 写入偏移 (`gpu_model_runner.py`): 非 last rank 更新 token buffer 时, 起始索引从 `num_computed_tokens` 改为 `self.input_batch.num_tokens_no_spec[req_index]`, 保留已有计数; 结束索引通过 `max(start_token_index, num_computed_tokens + len(new_token_ids))` 计算, 兼容 async 调度 (无 `new_token_ids`) 和 chunked prefill; 仅当范围有效时执行写入, 并设置 `is_token_ids = True`。
2. 补充 `is_token_ids` 更新 (`gpu_input_batch.py`): 在 `update_req_spec_token_ids` 中写入 spec token 时同步设置 `is_token_ids`, 与 `_update_states` 行为一致。
3. 新增单元测试 (`test_gpu_model_runner.py`): 添加两个测试函数覆盖 PP 非异步和异步调度场景, 验证 token buffer (`token_ids_cpu`、`num_tokens_no_spec`、`is_token_ids`) 在 rank 间的一致性。新增辅助函数 `_schedule_cached_requests` 构造缓存请求调度输出。
4. 回测验证: `lm_eval gsm8k` 精度从 0.8324 提升至 0.8772, 非异步调度对比验证正常。

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_states`): 核心修复文件, 修正 `_update_states` 中非 last rank 的 token 偏移逻辑, 从 `num_computed_tokens` 改为 `num_tokens_no_spec`, 并添加 `is_token_ids` 更新。

- `vllm/v1/worker/gpu_input_batch.py` (模块 输入批次; 类别 source; 类型 core-logic; 符号 `update_req_spec_token_ids`) : 补充 `update_req_spec_token_ids` 中 `is_token_ids` 的更新, 与 `_update_states` 保持一致, 避免 `prompt_embeds` 路径误判。
- `tests/v1/worker/test_gpu_model_runner.py` (模块 模型测试; 类别 test; 类型 test-coverage; 符号 `_schedule_cached_requests`, `test_update_states_pp_non_async_multi_request_keeps_token_buffers_consistent`, `test_update_states_pp_async_multi_request_keeps_rank_state_consistent`) : 新增两个单元测试和辅助函数, 覆盖 PP 非异步和异步场景, 验证 token buffer 在 rank 间的一致性。测试充分覆盖修复场景。

关键符号: `_update_states`, `update_req_spec_token_ids`, `_schedule_cached_requests`, `test_update_states_pp_non_async_multi_request_keeps_token_buffers_consistent`, `test_update_states_pp_async_multi_request_keeps_rank_state_consistent`

关键源码片段

`vllm/v1/worker/gpu_model_runner.py`

核心修复文件, 修正 `_update_states` 中非 last rank 的 token 偏移逻辑, 从 `num_computed_tokens` 改为 `num_tokens_no_spec`, 并添加 `is_token_ids` 更新。

```
if not is_last_rank:
    # 使用当前 num_tokens_no_spec 作为起始索引, 而不是 num_computed_tokens,
    # 因为 chunked prefill 下 num_computed_tokens 可能小于实际已写入的 token 数。
    start_token_index = self.input_batch.num_tokens_no_spec[req_index]

    # 结束索引为 start 与 (num_computed_tokens + len(new_token_ids)) 的最大值,
    # 兼容 async 调度 (无 new_token_ids) 和 chunked prefill 场景。
    end_token_index = max(
        start_token_index,
        num_computed_tokens + len(new_token_ids),
    )

    if end_token_index > start_token_index:
        if new_token_ids:
            # 只取需要追加的尾部 token, 避免覆盖已有正确内容。
            num_new_tokens = end_token_index - start_token_index
            tokens_to_append = new_token_ids[-num_new_tokens:]
            self.input_batch.token_ids_cpu[
                req_index, start_token_index:end_token_index
            ] = tokens_to_append
            # 标记该段为真实 token, 防止 enable_prompt_embeds 路径错误解释。
            self.input_batch.is_token_ids[
                req_index, start_token_index:end_token_index
            ] = True
            # 更新 num_tokens_no_spec 为最新结束索引。
            self.input_batch.num_tokens_no_spec[req_index] = end_token_index
```

`tests/v1/worker/test_gpu_model_runner.py`

新增两个单元测试和辅助函数，覆盖 PP 非异步和异步场景，验证 token buffer 在 rank 间的一致性。测试充分覆盖修复场景。

```
def test_update_states_pp_non_async_multi_request_keeps_token_buffers_consistent(
    model_runner, model_runner_2, dist_init, monkeypatch
):
    req_ids = ['req_0', 'req_1']
    non_last_runner = model_runner
    last_runner = model_runner_2
    non_last_runner.use_async_scheduling = False
    last_runner.use_async_scheduling = False

    # 模拟两个 rank 初始调度同一批请求
    monkeypatch.setattr(
        'vllm.v1.worker.gpu_model_runner.get_pp_group',
        lambda: SimpleNamespace(is_last_rank=False, world_size=2),
    )
    non_last_runner._update_states(_schedule_new_request(*req_ids))
    last_runner._update_states(_schedule_new_request(*req_ids))

    # 模拟 last rank 的采样结果已缓存到 token buffer
    sampled_by_last_rank = {req_ids[0]: 101, req_ids[1]: 201}
    for req_id, token_id in sampled_by_last_rank.items():
        req_index = last_runner.input_batch.req_id_to_index[req_id]
        start_idx = int(last_runner.input_batch.num_tokens_no_spec[req_index])
        end_idx = start_idx + 1
        last_runner.input_batch.token_ids_cpu[
            req_index, start_idx:end_idx
        ] = [token_id]
        last_runner.input_batch.is_token_ids[
            req_index, start_idx:end_idx
        ] = True
        last_runner.input_batch.num_tokens_no_spec[req_index] = end_idx
        last_runner.requests[req_id].output_token_ids.append(token_id)

    # 构造缓存请求的调度输出
    scheduler_output = _schedule_cached_requests(
        req_ids=req_ids,
        num_scheduled_tokens={req_ids[0]: 1, req_ids[1]: 1},
        new_token_ids=[[101], [201]],
        num_computed_tokens=[3, 3],
        num_output_tokens=[1, 1],
    )

    # 非 last rank 执行 _update_states
    monkeypatch.setattr(
        'vllm.v1.worker.gpu_model_runner.get_pp_group',
        lambda: SimpleNamespace(is_last_rank=False, world_size=2),
    )
```

```

non_last_runner._update_states(scheduler_output)

# 验证两个 rank 的 token buffer 一致
for req_id in req_ids:
    idx_nl = non_last_runner.input_batch.req_id_to_index[req_id]
    idx_lr = last_runner.input_batch.req_id_to_index[req_id]
    assert (
        non_last_runner.input_batch.num_tokens_no_spec[idx_nl] ==
        last_runner.input_batch.num_tokens_no_spec[idx_lr]
    )
    assert (
        non_last_runner.input_batch.token_ids_cpu[idx_nl] ==
        last_runner.input_batch.token_ids_cpu[idx_lr]
    ).all()
    assert (
        non_last_runner.input_batch.is_token_ids[idx_nl] ==
        last_runner.input_batch.is_token_ids[idx_lr]
    ).all()

```

评论区精华

- `num_new_tokens` 计算风险: Gemini Code Assist 和 Copilot 指出当 `req_state.output_token_ids` 已在前面被扩展时, 基于 `req_state.num_tokens` 计算的 `num_new_tokens` 会为 0, 导致跳过更新。作者最终改用 `num_tokens_no_spec` 作为基准, 通过 `max` 计算结束索引, 避免此问题。
- `is_token_ids` 缺失隐患: Copilot 发现写入 `token_ids_cpu` 后未更新 `is_token_ids`, 可能使 `enable_prompt_embeds` 分支将新 token 误认为 embed token。最初作者认为原代码也不更新, 但 reviewer @hsliuustc0106 要求修正并补充合理性说明。最终在 `_update_states` 和 `update_req_spec_token_ids` 中都添加了 `is_token_ids = True` 的赋值。
- 测试要求: @hsliuustc0106 要求添加针对 PP 场景的测试, 作者新增两个测试函数, 覆盖异步和非异步路径, 并验证 token buffer 在 rank 间的一致性。评审通过后合并。
 - `num_new_tokens` 计算可能为 0 导致 token 写入跳过 (correctness): 作者改用 `num_tokens_no_spec` 作为起始偏移, 通过 `max(start_token_index, num_computed_tokens + len(new_token_ids))` 计算结束索引, 确保写入正确。
 - `is_token_ids` 需要同步更新以支持 prompt_embeds (correctness): 在 `_update_states` 和 `update_req_spec_token_ids` 中都添加了 `self.input_batch.is_token_ids[req_index, start_token_index:end_token_index] = True`。
 - 需要添加单元测试覆盖 PP token 缓冲区一致性 (testing): 作者新增两个测试函数, 覆盖异步和非异步 PP 场景, hsliuustc0106 批准。

风险与影响

- 风险:
 - 回归风险: 低。变更集中在非 last rank 的 `_update_states` 分支, 且已通过新增测试和现有 `lm_eval` 验证。非 PP 模式完全不执行该分支, 无影响。

- 性能风险：无。执行路径无额外开销。
- 兼容性：无影响。修改保持了接口兼容。
- 影响范围：使用 PP 模式的高并发用户将显著提升精度（GSM8K +4.5%）；非 PP 用户不受影响。
- 影响：
 - 用户影响：使用 Pipeline Parallelism (PP) 的用户在高并发下精度大幅提升，尤其是长 prompt 的 chunked prefill 场景。非 PP 用户不受影响。
 - 系统影响：对系统负载、资源无变化。
 - 团队影响：为 PP 模式的状态管理提供了更稳健的实现，后续类似 bug 的排查可参考此修复模式。
 - 风险标记：核心路径变更，PP 模式兼容性，测试覆盖补充

关联脉络

- PR #40749 [Bugfix] Skip PP sampled-token receive on last rank during async scheduling: 共同修改 gpu_model_runner.py 中 PP 相关路径，改进 PP 模式状态管理