# PR #48901 完整报告

vllm-project/vllm

[Bugfix][Pooling] Fix wrong scores for chunked prefill under torch.compile

合并时间：2026-07-17 13:19

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48901

## 执行摘要

- 一句话：修复 torch.compile 下 chunked prefill 错误分数
- 推荐动作：该 PR 涉及 torch.compile 与异步调度交互的典型案例，值得精读。修复本身虽简单，但问题排查过程（差值分析、同步实验）对理解编译图生命周期有启发。建议关注后续改进方案。

## 功能与动机

根据 Issue #48831 报告，Qwen3-Reranker 在长序列（>8K tokens）时返回错误分数（如 0.33 vs 正确 0.83）。作者排查发现根因是 torch.compile 下 chunked prefill 导致隐藏状态缓冲区被覆盖，而 enforce_eager 路径正常。PR 旨在快速修复用户可见的正确性 bug。

## 实现拆解

1. 在 vllm/v1/worker/gpu_model_runner.py 的 _pool 方法中，当 raw_pooler_output is None or not any(finished_mask) 时，提前返回 ModelRunnerOutput 之前增加 self._sync_device() 调用，确保之前提交的设备内核全部完成，避免后续读取脏数据。
2. 在测试文件 tests/models/language/pooling/test_all_pooling_plus_chunked_prefill.py 中新增回归测试 test_last_pool_score_chunked_prefill_matches_unchunked：使用 Qwen3-Reranker 模型，对比 chunked（2048 tokens）与 unchunked（16384 tokens）的得分，断言两者接近（容忍 0.05）。该测试运行在 torch.compile 模式下，而原有测试仅覆盖 enforce_eager 路径。

关键文件：

- vllm/v1/worker/gpu_model_runner.py（模块 模型执行器；类别 source；类型 core-logic）：核心修复文件：在 _pool 方法 early return 前添加 self._sync_device() 调用，确保设备同步，避免读取未完成的隐藏状态。
- tests/models/language/pooling/test_all_pooling_plus_chunked_prefill.py（模块 池化测试；类别 test；类型 test-coverage；符号 test_last_pool_score_chunked_prefill_matches_unchunked, score_with）：新增回归测试，验证 torch.compile 下 chunked prefill 与 unchunked 得分一致，覆盖了原有测试未触及的编译路径。

关键符号：_pool, test_last_pool_score_chunked_prefill_matches_unchunked

## 关键源码片段

## vllm/v1/worker/gpu_model_runner.py

核心修复文件：在 _pool 方法 early return 前添加 self._sync_device() 调用，确保设备同步，避免读取未完成的隐藏状态。

```
# vllm/v1/worker/gpu_model_runner.py 中 _pool 方法片段
if raw_pooler_output is None or not any(finished_mask):
    # 同步设备以确保在此之前提交到流的内核全部完成，
    # 避免后续对输入缓冲区的写入（如 query_start_loc）被提前读取。
    # 这是一个临时修复，V2 已通过双缓冲区彻底解决此竞争。
    self._sync_device()
    model_runner_output.pooler_output = [None] * num_reqs
    return model_runner_output
```

## tests/models/language/pooling/test_all_pooling_plus_chunked_prefill.py

新增回归测试，验证 torch.compile 下 chunked prefill 与 unchunked 得分一致，覆盖了原有测试未触及的编译路径。

```
# tests/models/language/pooling/test_all_pooling_plus_chunked_prefill.py
# 新增回归测试：验证 torch.compile 下 chunked prefill 与 unchunked 得分一致

_RERANKER_HF_OVERRIDES = {
    "architectures": ["Qwen3ForSequenceClassification"],
    "classifier_from_token": ["no", "yes"],
    "is_original_qwen3_reranker": True,
}
_RERANKER_TEMPLATE = VLLM_PATH / "examples/pooling/score/template/qwen3_reranker.
jinja"


@pytest.mark.parametrize("model", ["Qwen/Qwen3-Reranker-0.6B"])
@torch.inference_mode
def test_last_pool_score_chunked_prefill_matches_unchunked(vllm_runner, model: str):
    """LAST-pooling score must not depend on whether the prompt is chunked.

    回归测试：在 torch.compile 下（非 enforce_eager），切分 prefill
    不应改变最终得分。
    """
    chat_template = _RERANKER_TEMPLATE.read_text()
    query = "What organelle produces energy in the cell?"
    # 长 document 使 query+doc 超过 chunk size
    document = ("The mitochondria is the powerhouse of the cell. It generates most of "
                "the cell chemical energy through oxidative phosphorylation. ") * 400

    def score_with(max_num_batched_tokens: int) -> float:
        with vllm_runner(
            model,
            runner="pooling",
            hf_overrides=_RERANKER_HF_OVERRIDES,
            max_model_len=16384,
```

```
            max_num_batched_tokens=max_num_batched_tokens,
            enable_chunked_prefill=True,
            enable_prefix_caching=False,
        ) as vllm_model:
            return vllm_model.score(query, document,
                                    chat_template=chat_template)[0]

    # 2048 强制多 chunk，16384 保持单 chunk 作为参考
    chunked = score_with(2048)
    unchunked = score_with(16384)

    assert chunked == pytest.approx(unchunked, abs=5e-2), (
        f"chunked score {chunked} diverged from unchunked {unchunked}"
    )
```

# 评论区精华

1. 修复是否应使用显式同步：njhill 认为 self._sync_device() 完全破坏了异步调度，不应该是正确方案；noooop 同意这不是理想修复，但为了 v0.26 尽快发布，先采用此临时修复，后续再改进。
2. 深层原因与 Runner V2：noooop 指出 V1 只有单缓冲区，可能导致写前读竞争（如 query_start_loc），V2 已采用双缓冲区可彻底避免。并引用 PR #37775 作为更多竞争案例。

- 修复方案是否应采用显式同步 (design): 作为临时方案接受，后续需找更优雅的修复（如双缓冲区）。
- 深层原因与 Runner V2 改进 (design): V2 架构可根本解决该问题，V1 修复为临时措施。

# 风险与影响

- 风险：
  - 性能风险：引入显式设备同步，在池化请求提前返回时增加开销，但该路径只在批处理未完成时执行，非热点。
  - 未解决根本问题：同步掩码了真正的缓冲区竞争，后续若修改编译路径可能重现。
  - 兼容性：修复仅在 CUDA 类似设备上加同步，不影响 CPU/XPU（它们已有同步）。
  - 测试覆盖：新增回归测试确保 chunked 与 unchunked 得分一致，但未测试其他池化模型或更大规模的组合。
- 影响：
  - 用户：Qwen3-Reranker 等 LAST-pooling 模型在长序列下得分正确，之前是错误的。
  - 系统：池化路径增加一次同步，但正常 decode 不受影响。
  - 团队：修复是临时方案，需跟进更优雅的解决（如双缓冲区或自定义 op 注解）。
  - 风险标记：临时修复，核心路径变更，性能退化风险，隐含竞争未根治

# 关联脉络

- 暂无明显关联 PR