

PR #51304 完整报告

vllm-project/vllm

[V1] Copy NaN-in-logits counts to host asynchronously

合并时间: 2026-08-07 08:05

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/51304>

执行摘要

- 一句话: V1 异步调度下 NaN 计数改为设备侧异步拷贝, 消除阻塞 D2H
- 推荐动作: 值得精读, 展示了如何将同步 D2H 转换为异步复制, 设计模式可复用。重点关注 `_bookkeeping_sync` 返回契约变更和 `AsyncGPUModelRunnerOutput` 的异步复制机制。

功能与动机

PR body 指出: `VLLM_COMPUTE_NANS_IN_LOGITS` 使 `_bookkeeping_sync` 在 main stream 上执行 `logits.isnan().sum(dim=-1).cpu()`, 每个 step 一次阻塞 D2H, 破坏了异步调度的目的。团队计划在 CI 默认开启 NaN 检查, 因此需要避免它拖垮仍使用 MRV1 的测试性能。V2 已经用异步拷贝规避了该问题, 本 PR 将同一模式移植到 V1。

实现拆解

1. 新增 `count_nans_per_row` 和 `nans_to_dict` 两个辅助函数, 分别负责在设备上计算每行 NaN 数, 以及将行计数映射回请求 ID。
2. 修改 `AsyncGPUModelRunnerOutput` 构造函数, 新增 `num_nans` 参数, 并在异步复制阶段将设备张量拷贝到宿主侧 `_num_nans_cpu`; `get_output()` 在 `async_copy_ready_event.synchronize()` 之后调用 `nans_to_dict` 构建 `output.num_nans_in_logits`, 并在 `VLLM_RAISE_ON_LOGIT_NANS` 时触发异常。
3. 修改 `_bookkeeping_sync` 的返回契约, 新增 `num_nans` 返回项; 在异步调度下只返回设备张量, 同步调度仍走 `_get_nans_in_logits` 读回宿主。
4. 重写 `_get_nans_in_logits`, 复用新的辅助函数, 并明确其仅用于同步调度。
5. 调用处将 `num_nans` 传递给 `AsyncGPUModelRunnerOutput`。测试配套: 无新增测试, 依赖现有 `test_raise_on_logit_nans` 与 `test_get_nans_in_logits` 覆盖。

关键文件:

- `vllm/v1/worker/gpu_model_runner.py` (模块 模型运行器; 类别 source; 类型 data-contract; 符号 `count_nans_per_row`, `nans_to_dict`, `_get_nans_in_logits`): 唯一变更文件, 核心改动: 新增 NaN 计数辅助函数, 修改异步输出封装和 bookkeeping 流程。

关键符号: `count_nans_per_row`, `nans_to_dict`, `_get_nans_in_logits`, `AsyncGPUModelRunnerOutput.get_output`

关键源码片段

vllm/v1/worker/gpu_model_runner.py

唯一变更文件，核心改动：新增 NaN 计数辅助函数，修改异步输出封装和 bookkeeping 流程。

vllm/v1/worker/gpu_model_runner.py 中新增的辅助函数

```
def count_nans_per_row(logits: torch.Tensor) -> torch.Tensor:
    """Per-row NaN counts, left on device."""
    # 使用 int32 累计每个 token 是否是 NaN，结果保留在 GPU 设备上，
    # 避免在主计算流上触发 blocking D2H 拷贝，这是异步调度的核心诉求。
    return logits.isnan().sum(dim=-1, dtype=torch.int32)

def nans_to_dict(counts: list[int], req_id_to_index: dict[str, int]) -> dict[str, int]:
    """Map per-row NaN counts onto request ids.

    Rows 和 requests 不需要一一对应，没有对应行的请求记 0。
    """
    # req_id_to_index 提供请求 id 到行索引的映射，行数可能少于请求数，
    # 因此越界索引统一补 0，保证每个请求都有计数。
    return {
        req_id: int(counts[i]) if i < len(counts) else 0
        for req_id, i in req_id_to_index.items()
    }

# _bookkeeping_sync 中的异步 / 同步分流逻辑
if envs.VLLM_COMPUTE_NANS_IN_LOGITS:
    if self.use_async_scheduling:
        # 异步调度下，计数保留在设备张量中，随异步输出流一起拷贝，
        # 避免阻塞主流的 host sync。
        num_nans = None if logits is None else count_nans_per_row(logits)
    else:
        # 同步调度下，仍然立即读回宿主，因为该路径本身就会阻塞。
        num_nans_in_logits = self._get_nans_in_logits(logits)

# AsyncGPUModelRunnerOutput.get_output 中在异步拷贝完成后构建结果
self.async_copy_ready_event.synchronize()
# ... 其他输出处理 ...
if self._num_nans_cpu is not None:
    # 异步拷贝完成后在宿主侧构建 dict，并可选触发异常
    output.num_nans_in_logits = nans_to_dict(
        self._num_nans_cpu.tolist(), output.req_id_to_index
    )
    if envs.VLLM_RAISE_ON_LOGIT_NANS:
        raise_if_nan_logits(output.num_nans_in_logits)
del self._num_nans
```

评论区精华

无实质性 review 讨论，仅有维护者 tlmchlsmith 直接批准。claude[bot] 因 PR 来自 fork 而跳过自动审查。

- 暂无高价值评论线程

风险与影响

- 风险：改动集中在 vllm/v1/worker/gpu_model_runner.py。风险点：_bookkeeping_sync 返回值增加一项，若有其他调用者未同步更新会引发解包错误（当前仅一处调用）；异步路径下 get_output() 在异常时可能未销毁设备张量，但现有模式已处理类似情况；同步调度路径行为不变。整体风险较低。
- 影响：影响 MRV1 异步调度模式下启用 NaN 检查的性能，消除每步阻塞同步；为 CI 默认开启 NaN 检查铺路。不影响默认行为，对用户无感知。团队收益是测试性能提升，V2 与 V1 行为保持一致。
- 风险标记：数据契约变更，核心路径变更

关联脉络

- 暂无明显关联 PR