PR #43909 完整报告

vllm-project/vllm

[Bug] Fix gemma4 MTP IMA issue when TP>1, `CUDA error: an illegal memory access was encountered`

合并时间：2026-05-30 22:34

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43909

执行摘要

- 一句话：修复 Gemma4 MTP TP>1 时 CUDA 非法内存访问
- 推荐动作：值得精读。该 PR 展示了 CUDA graph replay 场景下的经典问题：中间张量生命周期短于 graph 重放周期，导致非法内存访问。采用缓存 tensor 并确保 contiguous 的修复模式可作为团队内部处理类似问题的参考。

功能与动机

在 torch.cuda.CUDAGraph replay 过程中，_get_full_lm_head_weight 内通过 tensor_model_parallel_all_gather 返回的中间张量（lm_head_weight）可能被释放或失效，导致 CUDA error: an illegal memory access was encountered。PR 作者在 body 中提供了复现命令和完整错误栈，定位到错误发生在 _greedy_sample 的 graph replay 调用链中。

实现拆解

1. 在 __init__ 中新增缓存字段：在 Gemma4MTP.__init__ 中初始化 self._stable_full_lm_head_weight: torch.Tensor | None = None，用于缓存 all-gather 后的完整 lm_head 权重张量。
2. 修改 _get_full_lm_head_weight 方法：首先检查缓存是否存在，若存在则直接返回，避免重复 all-gather 和切片操作；当 tp_size > 1 时，将切片后的结果 .contiguous() 并赋值给缓存，确保张量在 CUDA graph replay 期间保持有效。
3. 在 load_weights 中重置缓存：在权重加载前将 _stable_full_lm_head_weight 置为 None，确保新权重生效。

关键文件：

- vllm/model_executor/models/gemma4_mtp.py（模块 模型实现；类别 source；类型 core-logic；符号 _get_full_lm_head_weight, _stable_full_lm_head_weight, load_weights, init）：唯一修改的文件，包含核心修复：引入 _stable_full_lm_head_weight 缓存并修改 _get_full_lm_head_weight 方法，解决 CUDA graph replay 时的非法内存访问。

关键符号：_get_full_lm_head_weight, load_weights

关键源码片段

vllm/model_executor/models/gemma4_mtp.py

唯一修改的文件，包含核心修复：引入 _stable_full_lm_head_weight 缓存并修改 _get_full_lm_head_weight 方法，解决 CUDA graph replay 时的非法内存访问。

```
# vllm/model_executor/models/gemma4_mtp.py (after fix)

class Gemma4MTP(nn.Module):
    def __init__(self, *, vllm_config: VllmConfig, prefix: str = ""):
        super().__init__()
        config = vllm_config.speculative_config.draft_model_config.hf_config
        text_config = _get_text_config(config)
        self.config = config
        # 新增：缓存 all-gather 后的完整 lm_head 权重，用于 CUDA graph replay
        self._stable_full_lm_head_weight: torch.Tensor | None = None

        # ... 其余初始化代码不变 ...

    def _get_full_lm_head_weight(self) -> torch.Tensor:
        # 若缓存有效，直接返回，避免重复 all-gather 和切片，防止 CUDA graph replay 时张量失效
        if self._stable_full_lm_head_weight is not None:
            return self._stable_full_lm_head_weight
        lm_head_weight = self.lm_head.weight
        tp_size = get_tensor_model_parallel_world_size()
        if tp_size > 1:
            lm_head_weight = tensor_model_parallel_all_gather(
                lm_head_weight,
                dim=0,
            )
        lm_head_weight = lm_head_weight[: self.masked_embedding.vocab_size]
        # 仅在 TP>1 时缓存，保持 TP=1 性能不变
        if tp_size > 1:
            lm_head_weight = lm_head_weight.contiguous()
            self._stable_full_lm_head_weight = lm_head_weight
        return lm_head_weight

    def load_weights(self, weights: Iterable[tuple[str, torch.Tensor]]) -> set[str]:
        # 重置缓存，确保权重更新后重新计算
        self._stable_full_lm_head_weight = None
        loader = AutoWeightsLoader(self)
        return loader.load_weights(weights, mapper=self.hf_to_vllm_mapper)
```

评论区精华

提交者 @yewentao256 在评论中仅提及 CC @lucianommartins，未展开讨论。审核者 @sfeng33 给出 LGTM 批准。无其他讨论线索。

- 暂无高价值评论线程

风险与影响

- 风险：风险极低。变更仅在该文件内部，_stable_full_lm_head_weight 缓存仅在 tp_size > 1 时被写入，且 load_weights 中重置保证权重更新正确性。但需注意：若后续有代码在 load_weights 后手动修改 lm_head.weight（如权重绑定），缓存可能失效，不过当前设计下 lm_head.weight 与 embed_tokens.weight 共享引用，且 load_weights 重置了缓存，因此影响可控。
- 影响：直接影响 Gemma4 MTP 模型在 tensor parallelism > 1 场景下的 CUDA graph replay 稳定性，修复了导致推理崩溃的 IMA 问题。不影响非 MTP 模型或 TP=1 的 Gemma4 推理。由于仅涉及单个文件 9 行增加、1 行删除，回归风险低。
- 风险标记：CUDA graph replay 风险，TP>1 特定场景

关联脉络

- 暂无明显关联 PR