PR #26430 完整报告

sgl-project/sglang

Fix GemmaRMSNorm gemma_weight buffer storage for Qwen3.5

合并时间：2026-05-28 18:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26430

执行摘要

- 一句话：修复 GemmaRMSNorm buffer 存储导致 CUDA Graph 失效
- 推荐动作：这是一个值得精读的微型实例：演示了 PyTorch 中 = 赋值与原地操作在 CUDA Graph 上下文下的关键区别。团队可借鉴此模式审查其他存在 buffer = expr 赋值且参与 CUDA Graph 捕获的模块。

功能与动机

来自 PR body：运行 Qwen3.5 时，首次权重更新后 logp 高达约 3.0。根本原因是 #22673 将 weight 输入改为 self.gemma_weight，但 weight_loader 中 self.gemma_weight = param.data + 1.0 会分配新内存，导致 CUDA Graph 仍使用旧权重。

实现拆解

1. 修改初始化：在 __init__ 中将 gemma_weight buffer 的初始化从 self.weight.data + 1.0（创建新 tensor）改为 torch.ones_like(self.weight)（分配持久 buffer）。这样 buffer 的存储地址在模型生命周期内固定不变。
2. 原地更新权重：在 _weight_loader 中将 self.gemma_weight = param.data + 1.0（重新赋值，新内存）替换为 torch.add(param.data, 1.0, out=self.gemma_weight)（原地加法，写入同一 buffer）。确保 CUDA Graph 捕获的 buffer 引用始终有效。
3. 不变之处：_forward_impl 和其余前向逻辑保持不变，因为 _forward_impl 实际使用 self.weight.data 而非 self.gemma_weight，避免了符号歧义。

关键文件：

- python/sglang/srt/layers/layernorm.py（模块 层归一化；类别 source；类型 core-logic；符号 GemmaRMSNorm, _weight_loader）：唯一修改的文件，包含 GemmaRMSNorm 类的 __init__ 和 _weight_loader 方法，修复了 buffer 存储不稳定导致 CUDA Graph 失效的问题。

关键符号：GemmaRMSNorm.init, GemmaRMSNorm._weight_loader

关键源码片段

python/sglang/srt/layers/layernorm.py

唯一修改的文件，包含 GemmaRMSNorm 类的 __init__ 和 _weight_loader 方法，修复了 buffer 存储不稳定导致 CUDA Graph 失效的问题。

```
# python/sglang/srt/layers/layernorm.py
# GemmaRMSNorm 类核心变更：保持 buffer 内存地址固定，以兼容 CUDA Graph

class GemmaRMSNorm(MultiPlatformOp):
    def __init__(
        self,
        hidden_size: int,
        eps: float = 1e-6,
    ) -> None:
        super().__init__()
        self.weight = nn.Parameter(torch.zeros(hidden_size))
        self.variance_epsilon = eps
        # 改动 1：用 torch.ones_like 创建持久 buffer
        # 原代码 self.register_buffer("gemma_weight", self.weight.data + 1.0, ...)
        # 会分配新内存，导致后续赋值时丢弃原 buffer 的固定地址
        self.register_buffer(
            "gemma_weight", torch.ones_like(self.weight), persistent=False
        )
        self.weight.weight_loader = self._weight_loader

    def _weight_loader(self, param: torch.Tensor, loaded_weight: torch.Tensor) -> None:
        assert param.size() == loaded_weight.size()
        param.data.copy_(loaded_weight)
        # 改动 2：使用 torch.add 原地更新，确保 CUDA Graph 捕获的 buffer 引用指向同一内存
        # 原代码 self.gemma_weight = param.data + 1.0 会破坏存储稳定性
        torch.add(param.data, 1.0, out=self.gemma_weight)

    # _forward_impl 使用 self.weight.data 而非 self.gemma_weight，因此无需修改
    def _forward_impl(self, x, residual=None, post_residual_addition=None):
        # ... ( 保持不变 )
        out = gemma_rmsnorm(x, self.weight.data, self.variance_epsilon)
        return out
```

评论区精华

PR 无 review 评论，仅有两位 maintainer 的快速 approval。说明变更逻辑清晰、风险可控，社区共识高。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. CUDA Graph 兼容性：修改确保 buffer 存储固定，与 CUDA Graph 捕获兼容，风险已消除。
  2. 数值正确性：torch.add 原地操作与 new_tensor = a + 1.0 结果一致，无精度差异。

3. 回归风险：仅改动 GemmaRMSNorm 内部实现，不涉及其他模块，回归概率低。但缺少对应测试用例，未来修改可能覆盖不到。 - 影响：影响范围集中：仅修复使用 GemmaRMSNorm 且权重可变的模型（如 Qwen3.5、Gemma 系列）。用户无需感知代码变更，但推理正确性得到保障。CUDA Graph 在权重更新后能正确捕获新权重，提升模型微调 / 持续学习场景的稳定性。 - 风险标记：缺少测试覆盖，CUDA Graph 兼容性问题

关联脉络

- PR #22673 Previous change that introduced gemma_weight usage in forward: 本 PR 修复了 #22673 引入的副作用——gemma_weight 的内存分配问题导致 CUDA Graph 失效。