PR #2079 完整报告

radixark/miles

Fix padded -1 index handling in GLM-5 sparse-MLA tilelang kernels

合并时间：2026-08-08 14:37

原文链接：http://prhub.com.cn/radixark/miles/pull/2079

执行摘要

- 一句话：修复 GLM-5 sparse-MLA kernel 的 -1 索引越界与 NaN 梯度
- 推荐动作：值得精读：这是典型的 GPU kernel 数值安全修复，展示了如何处理 padding 索引越界、0 * inf、0/0 与越界 atomic 写四类问题，且设计上刻意让修复对真实行完全透明。同时建议优先补两条后续：把 standalone harness 入库并挂到 CI，以及同步 megatron-bridge 的 vendored 副本。

功能与动机

PR body 明确指出 GLM-5 稀疏 MLA tilelang kernel 存在两个 NaN 来源：padding 槽位用 -1 索引时读写越界（前向 0 * inf、反向 dKV atomic_addx4 越界写），以及全 padding 行 sumexp = 0 导致 0/0 与 LSE -inf。这些 NaN 在真实 4 节点 GLM-5.2 744B LoRA 训练中表现为 train/grad_norm = nan，而 loss 完全有限，因此需要单独修复 kernel。

实现拆解

1. 定位两个 NaN 来源：miles_plugins/models/glm5/ops/tilelang_sparse_mla_fwd.py 与 tilelang_sparse_mla_bwd.py 是 GLM-5 稀疏 MLA 的 tilelang kernel。top-k 索引张量用 -1 填充无用槽位，-1 实际寻址到 KV 张量前一个元素：前向中越界垃圾字节被吸收进 -inf 分数，但只要垃圾值溢出，与恰好为零的注意力权重相乘就变成 0 * inf = NaN；反向中同样的字节经 dO @ KV 进入 acc_dp，且 dKV 的 atomic_addx4 直接越界写。全 padding 行（所有索引都是 -1）则因 sumexp = 0 触发 0/0 = NaN 与 LSE -inf。
2. 前向 kernel 修复：在 main 中新增 kv_i fragment，先把索引用 T.max(Indices[..], 0) clamp 到合法范围，再通过 mask（Indices[..] != -1）用 T.if_then_else 让 padding 槽位加载真正的 0 值 key（KV_shared、K_tail_shared），使分数、注意力权重和最终输出都是精确零；随后对每个 head 的 sumexp 执行 T.max(sumexp[h_i], 1e-30) floor，让全 padding 行输出精确 0 且 LSE 不再为 -inf，同时真实行（sumexp >= 1）保持不变。
3. 反向 kernel 修复：在 sparse_mla_bwd_kernel 中同样新增 kv_i 并 clamp，KV_shared、KV_tail_shared 的 gather 都改为 mask + clamp 后的索引，保证 padding 贡献精确零，避免 acc_dp 中的垃圾字节；dKV 与 dKV_tail 的 T.atomic_addx4 也改用 kv_i，由于 padding 槽位对应的 P、dP 列是零，clamp 后原子写变成写回自身零值的 no-op，不再越界破坏 dKV 之前的分配。
4. 验证与配套：PR 通过生产形状（H=8、topk=2048、ragged 序列长度、包括全 -1 行）的 standalone kernel harness 验证，修复后前反向与 fp32 参考对齐到 3e-3，空行精确零；

并在 4 节点 GLM-5.2 744B LoRA 训练中确认每个 step 的 train/grad_norm 有限。改动未引入仓库内持久化测试（harness 未入库），且 megatron-bridge 中 vendored 的 kernel 副本需要同样修复，PR 未覆盖。

关键文件：

- miles_plugins/models/glm5/ops/tilelang_sparse_mla_bwd.py（模块 GLM5 反向核；类别 source；类型 core-logic；符号 sparse_mla_bwd_kernel）：反向 kernel 是 NaN 梯度与越界写的主要来源：-1 索引的垃圾字节进入 acc_dp 且 atomic_addx4 越界写 dKV。修复后通过 clamp + mask 让 padding 槽位贡献精确零，原子写变为 no-op。
- miles_plugins/models/glm5/ops/tilelang_sparse_mla_fwd.py（模块 GLM5 前向核；类别 source；类型 core-logic；符号 main）：前向 kernel 的第一个 NaN 来源：-1 索引越界读与全 padding 行的 0/0 输出。修复后 padding 槽位加载真正 0 值 key，sumexp floor 保证空行输出精确 0。

关键符号：main (tilelang_sparse_mla_fwd.py), sparse_mla_bwd_kernel

关键源码片段

miles_plugins/models/glm5/ops/tilelang_sparse_mla_bwd.py

反向 kernel 是 NaN 梯度与越界写的主要来源：-1 索引的垃圾字节进入 acc_dp 且 atomic_addx4 越界写 dKV。修复后通过 clamp + mask 让 padding 槽位贡献精确零，原子写变为 no-op。

```
# miles_plugins/models/glm5/ops/tilelang_sparse_mla_bwd.py
# 稀疏 MLA 反向 kernel 的核心片段：修复 -1 索引越界读写

    # 计算 mask 并 clamp 索引：padding 槽位（-1）寻址到 KV 之前的元素，
    # 这些垃圾字节会经 dO @ KV 进入 acc_dp，再与恰好为 0 的 acc_p 相乘，
    # 产生 0 * inf = NaN；clamp 后用 mask 替换真正 0 值 key。
    for bi_i in T.Parallel(BS):
        mask[bi_i] = Indices[by, s_i, bz // NH, i_i * BS + bi_i] != -1
    for bi_i in T.Parallel(BS):
        kv_i[bi_i] = T.max(Indices[by, s_i, bz // NH, i_i * BS + bi_i], 0)

    # 加载 KV 与 KV_tail：padding 槽位贡献精确 0，避免垃圾字节进入 acc_dp
    for bi_i, d_i in T.Parallel(BS, D):
        KV_shared[bi_i, d_i] = T.if_then_else(mask[bi_i], KV[by, kv_i[bi_i], bz // NH, d_i], 0)
    for bi_i, d_i in T.Parallel(BS, D_tail):
        KV_tail_shared[bi_i, d_i] = T.if_then_else(mask[bi_i], KV[by, kv_i[bi_i], bz // NH, D + d_i], 0)

    # ... 计算 acc_p / acc_dp / acc_dkv ...

    # padding 槽位的 P 与 dP 列都是 0，因此 clamp 后的地址让原子写
    # 变成自己加 0 的 no-op，而不是越界写坏 dKV 之前的分配。
    for bi_i, d_i in T.Parallel(BS // split_store, D // 4):
        T.atomic_addx4(
            dKV[by, kv_i[bi_i + s * (BS // split_store)], bz // NH, d_i * 4],
            acc_dkv_shared[bi_i, d_i * 4],
```

```
        )
    for bi_i, d_i in T.Parallel(BS // split_store, D_tail // 4):
        T.atomic_addx4(
            dKV[by, kv_i[bi_i + s * (BS // split_store)], bz // NH, D + d_i * 4],
            acc_dkv_tail_shared[bi_i, d_i * 4],
        )
```

miles_plugins/models/glm5/ops/tilelang_sparse_mla_fwd.py

前向 kernel 的第一个 NaN 来源：-1 索引越界读与全 padding 行的 0/0 输出。修复后 padding 槽位加载真正 0 值 key，sumexp floor 保证空行输出精确 0。

```
# miles_plugins/models/glm5/ops/tilelang_sparse_mla_fwd.py
# 稀疏 MLA 前向 kernel 的核心片段：处理 padding 槽位 -1 索引与全 padding 行

    # 计算 mask：padding 槽位（索引 == -1）标记为 False
    for bi_i in T.Parallel(BI):
        mask[bi_i] = Indices[b_i, s_i, g_i, i_i * BI + bi_i] != -1

    # -1 会寻址到 KV 张量之前的元素（越界读）。这些垃圾字节被
    # 恰好为 0 的注意力权重相乘后产生 0 * inf = NaN。先把索引 clamp 到 0，
    # 再用 mask 把 padding 槽位替换成真正的 0 值 key，保证贡献精确 0。
    for bi_i in T.Parallel(BI):
        kv_i[bi_i] = T.max(Indices[b_i, s_i, g_i, i_i * BI + bi_i], 0)

    # 加载 KV 与 KV_tail：padding 槽位写入 0，而不是读取越界地址
    for bi_i, d_i in T.Parallel(BI, D):
        KV_shared[bi_i, d_i] = T.if_then_else(mask[bi_i], KV[b_i, kv_i[bi_i], g_i, d_i], 0)
    for bi_i, d_i in T.Parallel(BI, D_tail):
        K_tail_shared[bi_i, d_i] = T.if_then_else(mask[bi_i], KV[b_i, kv_i[bi_i], g_i, D + d_i], 0)

    # ... GEMM 计算注意力分数与输出 ...

    # 全 padding 行没有有效 key：sumexp = 0，rescale 时 0 / 0 = NaN，
    # LSE 变成 -inf，反向 exp2(-inf - -inf) = NaN。
    # 真实行至少有一个有效 key，sumexp >= 1（running max 项为 exp2(0)），
    # 因此 floor 到 1e-30 对真实行是 no-op，只保护空行输出精确 0。
    for h_i in T.Parallel(H_per_block):
        sumexp[h_i] = T.max(sumexp[h_i], 1e-30)
    for h_i, d_i in T.Parallel(H_per_block, D):
        acc_o[h_i, d_i] /= sumexp[h_i]
```

评论区精华

PR 没有实质 review 评论：yueming-yuan 直接 APPROVED，唯一的 comment 来自 gemini-code-assist[bot] 的下线公告。设计权衡主要在 PR body 中阐述：clamp + mask 替代越界读、sumexp floor 只保护空行（真实行 sumexp >= 1 所以无影响）、反向 atomic 在 padding 槽位变成无害 no-op。PR 同时指出与 #1571 的 shared-memory-merge 误编译相互独立且都需要修复，并提醒 megatron-bridge vendored 副本要同步处理。

- Review 结论 (other): PR 获批准并合入 main，无待解决疑虑。

风险与影响

- 风险：
  - 回归风险：修改的是 GLM-5 稀疏 MLA 训练核心 kernel 路径，任何索引计算或 masking 的偏差都会直接影响梯度。虽然 standalone harness 验证了与 fp32 参考 3e-3 对齐，但 harness 未入库，缺少 CI 回归保护。
  - 未同步 vendored 副本：PR body 明确说明 megatron-bridge 中还有一份同一 kernel 的副本需要同样处理，当前合并后从 bridge provider 构建 layer spec 的 launcher 仍可能遇到 NaN 与越界写，形成安全隐患。
  - 数值假设：修复依赖两个假设——padding 槽位的 P/dP 列严格为零（atomic 才会是 no-op），以及真实行 sumexp >= 1（floor 才无副作用）；如果未来 kernel 变体改变 padding 行为，这两点需要重新验证。
  - 性能：新增一次 mask 判断与 if_then_else 访存，padding 行会写入额外零值，相对 GEMM 主计算开销可忽略。
  - 影响：影响范围：所有使用 miles_plugins GLM-5 稀疏 MLA tilelang kernel 的训练流程，尤其是包含 padding 行的 LoRA/PPO 训练；修复后从 step 0 起即可得到有限 grad_norm，不再需要规避 batch 含 padding 行。对不经过此 kernel 的后端无影响。团队侧需要跟进 megatron-bridge vendored 副本的同步修复，否则 bridge 路径仍带同一隐患。
  - 风险标记：核心数值路径变更，缺少仓库内回归测试，megatron-bridge vendored 副本未同步

关联脉络

- PR #1571 shared-memory-merge miscompile 修复（被本 PR body 引用）：PR body 明确指出本修复与 #1571 的 shared-memory-merge 误编译问题互补，两者独立且都需要才能获得有限梯度。
- PR #2215 fix(mtp): double-shift GPT-path MTP labels: 同为模型训练正确性修复，涉及 miles_plugins megatron_bridge 与 model.py，属同一 GLM 系列稳定性收敛线。
- PR #2226 fix: preserve routing-replay state around MTP spec creation: 同为 GLM 训练崩溃修复（critic 崩溃），说明 GLM 系列 kernel/ 模型路径近期有多处正确性收敛。
- PR #2223 Remove --disable-weights-backuper; default eligible colocate launchers to rematerialize: 涉及 scripts/run_glm5_744b_a40b.py 与 GLM-5.2 744B LoRA 训练 CI，正是本 PR 修复验证的训练场景。