PR #48366 完整报告

vllm-project/vllm

[Bugfix] Prevent NaN poisoning in xpu_mla_sparse for fully-masked index chunks

合并时间：2026-07-27 09:06

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/48366

执行摘要

- 一句话：修复 xpu_mla_sparse 中全掩码块引发的 NaN 污染
- 推荐动作：该 PR 修复了一个重要的数值 Bug，改动极小（两行），新增的回归测试充分覆盖了边缘情况。虽然有轻微精度下降报告但可能是噪声，建议合并并后续监控。

功能与动机

修复 Issue #48364 报告的 NaN 污染：当一行 topk 索引的第一个 BLOCK_N（16）条目全为掩码时，_bf16_mla_sparse_kernel 输出 NaN，导致 DeepSeek-V4 XPU prefill 等路径产生无效结果。该问题属于纯数值 Bug，此前未被测试捕获。

实现拆解

1. 在 vllm/v1/attention/ops/xpu_mla_sparse.py 的 _bf16_mla_sparse_kernel 中，将 running max 初始化从 float('-inf') 改为 1.0e30（形式为 tl.zeros(...) - 1.0e30），避免全掩码块导致 re_scale = exp2(-inf - -inf) = NaN。
2. 同时，将掩码 logit 的填充值从 -float('inf') 改为 -1.0e30，确保被掩码块中 exp2(qk - max) 不为 NaN。
3. 在 tests/kernels/attention/test_xpu_mla_sparse.py 中新增回归测试 test_bf16_triton_sparse_mla_masked_chunks，覆盖三种场景：前中后掩码块（有效键在中间）、全有效行、全掩码行，验证输出有限且与参考实现一致。

关键文件：

- tests/kernels/attention/test_xpu_mla_sparse.py（模块 测试；类别 test；类型 test-coverage；符号 test_bf16_triton_sparse_mla_masked_chunks）：新增回归测试 test_bf16_triton_sparse_mla_masked_chunks，覆盖前中后掩码块和全掩码行，确保 NaN 不再出现且与参考一致。
- vllm/v1/attention/ops/xpu_mla_sparse.py（模块 注意力计算；类别 source；类型 core-logic；符号 _bf16_mla_sparse_kernel）：核心修复文件，修改两行将 -inf 替换为 -1e30，解决全掩码块导致的 NaN 污染。

关键符号：_bf16_mla_sparse_kernel, test_bf16_triton_sparse_mla_masked_chunks

关键源码片段

tests/kernels/attention/test_xpu_mla_sparse.py

新增回归测试 test_bf16_triton_sparse_mla_masked_chunks，覆盖前中后掩码块和全掩码行，确保 NaN 不再出现且与参考一致。

```
@pytest.mark.parametrize("device_str", ["xpu"])
@pytest.mark.parametrize("dtype", [torch.float16, torch.bfloat16])
@pytest.mark.skipif(not torch.xpu.is_available(), reason="XPU is required")
def test_bf16_triton_sparse_mla_masked_chunks(device_str, dtype):
    """Regression test: leading masked chunks must not produce NaN."""
    device = torch.device(device_str)
    # 构造 3 行：row0 有效键在块 1-2 (leading & trailing masked),
    # row1 全有效，row2 全掩码
    s_q, s_kv, h_q, h_kv, d_qk, d_v, topk = 3, 256, 64, 1, 576, 512, 128
    torch.random.manual_seed(1234)
    q = torch.randn((s_q, h_q, d_qk), dtype=dtype, device=device)
    kv = torch.randn((s_kv, h_kv, d_qk), dtype=dtype, device=device)
    indices = torch.full((s_q, h_kv, topk), -1, dtype=torch.int32, device=device)
    # row0: 前 16 个索引全掩码（默认 -1），16-47 为有效 key
    indices[0, 0, 16:48] = torch.arange(32, dtype=torch.int32, device=device)
    # row1: 全部 128 个索引有效
    indices[1, 0, :] = torch.arange(topk, dtype=torch.int32, device=device)
    # row2: 全掩码（默认 -1）
    sm_scale = d_qk ** -0.5
    out, max_logits, lse = triton_bf16_mla_sparse_interface(q, kv, indices, sm_scale, d_v)
    # 验证输出全部有限（无 NaN/Inf）
    assert out.isfinite().all()
    # 与参考实现对比前两行
    ref_out, _, ref_max_logits, ref_lse = reference_mla_sparse_prefill(q, kv, indices, sm_scale, d_
    v)
    assert torch.allclose(out[:2], ref_out[:2], atol=1e-2, rtol=1e-2)
    assert torch.allclose(max_logits[:2], ref_max_logits[:2], atol=1e-3, rtol=1e-3)
    assert torch.allclose(lse[:2], ref_lse[:2], atol=1e-3, rtol=1e-3)
    # 全掩码行输出应为零
    assert torch.allclose(out[2], torch.zeros_like(out[2]))
```

评论区精华

wuxun-zhang 请求 majian4work 使用 DeepSeek 模型验证精度回归；majian4work 回复 GSM8k 得分从 0.946 降至 0.94，认为是噪声。此外，nickus 由于无写入权限请求维护者协助合并。

- Accuracy regression check on DeepSeek model (correctness): 轻微下降可能是噪声，未进一步行动
- Merge permission request (other): jikunshang 合并了 PR

风险与影响

- 风险：主要风险是精度回归：majian4work 在 DeepSeek 模型上观察到 GSM8k 下降 0.6 个百分点，但由于噪声大，需更多验证。修复本身是数学等价的（-1e30 在 fp32 中与 -inf 在非掩码场景下行为相同），但全掩码行从 NaN 变为零，可能改变下游分布。影响面覆盖 xpu_mla_sparse 所有调用路径（XPU_MLA_SPARSE 后端、DeepSeek-V4 prefill、fp8 decode wrapper）。但改动仅限于两行，回归风险较低。
- 影响：影响所有使用 xpu_mla_sparse 内核的 Intel GPU 用户，修正 NaN 问题，使输出符合预期（零值）。无 API 变更，对性能无影响。
- 风险标记：数值边界修正，潜在精度回归

关联脉络

- PR #47629 [Attention] TRITON_MLA_SPARSE backend for SM80/SM121 sparse MLA (rebase & takeover of #38476): 该 PR 使用了与 #47629 相同的 sentinel 修复思路，且被修复的内核与 TRITON_MLA_SPARSE 后端有技术关联。