# PR #22434 完整报告

sgl-project/sglang

[CPU][sgl-kernel] `extend_attention_cpu` and `flash_attn_varlen_func`: fix `nan` for large seq

合并时间：2026-04-17 13:01

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/22434

## 执行摘要

本 PR 修复了 CPU 上 extend_attention_cpu 和 flash_attn_varlen_func 内核中因果掩码逻辑错误，解决了输入序列长度超过 4096 时出现的 NaN 崩溃问题。通过调整掩码条件和钳位 last_col，避免了越界写入和掩码不全，并新增测试用例确保修复有效性。

## 功能与动机

为什么做：Issue 20051 报告了在 CPU 上使用 BF16 精度时，当 prefill 长度超过约 4096 tokens 时，GDN 内核产生 NaN 值，导致 torch.multinomial 崩溃。PR body 进一步指出，在 extend_attention_cpu 和 flash_attn_varlen_func 内核中，因果掩码逻辑错误导致大序列输入时出现 NaN。具体地，当 BLOCK_M=512、BLOCK_N=768 时，原有掩码条件 num_keys - n <= BLOCK_N 仅对最后一个键块生效，忽略了首个键块中的未来键，且 last_col 为负时引发越界写入。

## 实现拆解

1. 修改因果掩码条件：在 sgl-kernel/csrc/cpu/extend.cpp 和 sgl-kernel/csrc/cpu/flash_attn.cpp 中，将掩码条件从 num_keys - n <= BLOCK_N 更新为 n + n_size - 1 > m。这个新条件基于块中最后一个键的位置（n + n_size - 1）是否严格大于首个查询位置（m），从而正确识别需要掩码的块。cpp // 示例代码片段来自 extend.cpp if (n + n_size - 1 > m) { for (int row = 0; row < m_size; ++row) { int last_col = m + row - n; last_col = std::max(last_col, -1); // 钳位到-1，避免负索引 float* row_ptr = s_i + row * BLOCK_N; fill_stub(row_ptr + last_col + 1, -inf, n_size - last_col - 1); } }
2. 钳位 last_col 到 -1：添加 last_col = std::max(last_col, -1)，防止当 last_col 为负时 fill_stub 写入非法内存地址，确保整个行被掩码为 -inf。
3. 扩展测试覆盖：在 test/srt/cpu/test_extend.py 中，修改 _test_extend_attention_once 以支持固定长度参数，并新增 test_extend_attention_large_seq_causal_mask 测试序列长度 5000。在 test/srt/cpu/test_flash_attn.py 中，类似地新增 _test_flash_attn_large_seq_causal_mask_once 和 test_flash_attn_large_seq_causal_mask，覆盖单序列和多序列场景。
4. 测试代码重构：根据 review 反馈，作者通过提交重构测试代码，减少 _test_extend_attention_once 和 _test_extend_attention_fixed_lens 之间的重复逻辑，提升可维护性。

### 关键源码片段

#### sgl-kernel/csrc/cpu/extend.cpp

核心注意力扩展内核，修复因果掩码逻辑错误，防止大序列输入时 NaN 产生和越界写入。

```
// apply causal mask
// [Note] condition to apply causal mask.
// Mask any block whose last key (n + n_size - 1) is strictly after the first query position (m), i.e.
n + n_size - 1 > m.
// 原条件 num_keys - n <= BLOCK_N 仅在最后一个n-block生效，但BLOCK_M=512, BLOCK_N=
768时，首个n-block可能包含未来键。
if (n + n_size - 1 > m) {
  for (int row = 0; row < m_size; ++row) {
    int last_col = m + row - n;
    // [Note] mask the entire row if last_col < 0.
    // 当 n > m + row 时，该块所有键都是未来键，需要掩码整个行。
    // 钳位到-1，避免 last_col+1 <= 0 导致 fill_stub 越界写入。
    last_col = std::max(last_col, -1);
    // fill [last_col + 1, n_size) to -inf
    float* row_ptr = s_i + row * BLOCK_N;
    fill_stub(row_ptr + last_col + 1, -std::numeric_limits<float>::infinity(), n_size - last_col - 1);
  }
}
```

## 评论区精华

reviewer gemini-code-assist[bot] 指出测试辅助函数存在代码重复问题：

> “The new helper function _test_extend_attention_fixed_lens shares a lot of logic with the existing _test_extend_attention_once, leading to significant code duplication. To improve maintainability, I recommend refactoring these two methods.”

作者在后续提交中解决了此问题，通过提取公共逻辑简化了测试代码。

## 风险与影响

- 技术风险：修改涉及核心注意力路径，可能引入回归（如掩码过度或不足），但新增测试用例降低了风险；额外钳位操作对性能影响可忽略。
- 影响范围：对用户，解决了长序列处理中的 NaN 崩溃，提升 CPU 上模型的稳定性；对系统，确保注意力计算正确性，避免 NaN 传播；对团队，提供了更健壮的内核实现，便于未来扩展 CPU 功能。

## 关联脉络

- 关联 Issue 20051：直接驱动了本 PR，报告了 GDN 内核的 NaN 问题，本 PR 修复了更通用的注意力内核问题。

- 历史 PR 关联：与 PR 22842（新增 CPU 内核）和 PR 22990（修复调度逻辑）类似，都涉及核心模块的 bugfix 和优化，反映仓库在 CPU 支持和系统稳定性上的持续演进。近期 PR 如 22406（优化 KV 缓存）也展示了内核层的性能改进趋势。