PR #26911 完整报告

sgl-project/sglang

[Bugfix] Gate DP-attention even-token padding to CP-enabled configs

合并时间：2026-06-03 14:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26911

执行摘要

- 一句话：修复 DP-attention 偶数 token 填充导致 NaN 崩溃
- 推荐动作：值得精读。这是一个典型的高影响力小修复案例：通过集中化逻辑和条件化对齐，解决了由之前 PR 引入的回归问题。设计决策（仅对 zigzag 模式应用 2x 对齐）体现了对 CP 内部机制的理解。建议工程师关注：
  - 如何通过提取函数避免两个调用点的重复逻辑和潜在不一致性。
  - Review 中的设计讨论如何推动了更简洁的实现。
  - FIXME 注释标记了一个潜在的未来改进方向（让 draft prefill-extend 容忍填充的虚拟 token）。

功能与动机

修复 EAGLE/MTP 推测解码中 DP attention 场景下的 NaN 崩溃 (Assertion 'NaN detected! draft_extend_for_prefill' failed)，该问题由 PR #23269 引入。该崩溃导致 nightly-8-gpu-b200 测试中的 TestEagleDPAttnServerLarge.test_a_gsm8k 失败。

实现拆解

1. 新增集中式对齐函数get_cp_padding_align_size() 于 python/sglang/srt/layers/utils/cp_utils.py：该函数根据当前 CP 模式返回适当的对齐大小——zigzag (in-seq-split) CP 返回 attn_cp_size * 2，否则返回 attn_cp_size（CP 关闭时 attn_cp_size == 1，因此 ceil_align 为无操作）。该函数通过本地导入 is_dsa_prefill_cp_in_seq_split 来覆盖 DSA 路径。
2. 修改 prepare_mlp_sync_batch（python/sglang/srt/model_executor/forward_batch_info.py）：移除对 get_attention_cp_size() 的直接调用和硬编码的 * 2，改为调用 get_cp_padding_align_size() 获取对齐大小。同时添加循环导入的保护性本地导入，并在注释中标记 FIXME 供后续改进。
3. 修改 cal_padded_tokens（python/sglang/srt/layers/attention/dsa/utils.py）：同步更新该 DSA 工具函数中的对齐逻辑，使其与 prepare_mlp_sync_batch 保持一致，同样调用 get_cp_padding_align_size()。

关键文件：

- python/sglang/srt/layers/utils/cp_utils.py（模块 CP 工具；类别 source；类型 core-logic；符号 get_cp_padding_align_size）：新增核心函数 get_cp_padding_align_size，集中管

理 CP 填充对齐大小逻辑，是本 PR 的中心变更。

- python/sglang/srt/model_executor/forward_batch_info.py（模块 前向批处理；类别 source；类型 data-contract；符号 prepare_mlp_sync_batch）：修改 prepare_mlp_sync_batch 方法，移除硬编码的 attn_cp_size * 2 并对齐到新函数。
- python/sglang/srt/layers/attention/dsa/utils.py（模块 DSA 工具；类别 source；类型 dependency-wiring；符号 cal_padded_tokens）：修改 cal_padded_tokens 函数以保持与 prepare_mlp_sync_batch 一致的填充逻辑，避免 DSA 路径产生不一致。

关键符号：get_cp_padding_align_size, prepare_mlp_sync_batch, cal_padded_tokens

关键源码片段

python/sglang/srt/layers/utils/cp_utils.py

新增核心函数 get_cp_padding_align_size，集中管理 CP 填充对齐大小逻辑，是本 PR 的中心变更。

```
def get_cp_padding_align_size() -> int:
    """Token-count alignment for CP padding of global_num_tokens: 2 * cp_size
    for zigzag (in-seq-split) CP, otherwise cp_size (1 when CP is off, so the
    padding is a no-op; extra padding breaks EAGLE/MTP draft prefill, see
    #23269). Keep prepare_mlp_sync_batch and cal_padded_tokens consistent
    through this helper.
    """
    # 本地导入避免循环依赖
    from sglang.srt.layers.attention.dsa.utils import is_dsa_prefill_cp_in_seq_split

    attn_cp_size = get_attention_cp_size()
    # 只有 zigzag (in-seq-split) 模式才需要 2x 对齐以平衡负载
    if is_prefill_cp_in_seq_split() or is_dsa_prefill_cp_in_seq_split():
        return attn_cp_size * 2
    # CP 关闭时 attn_cp_size == 1，ceil_align 无操作
    return attn_cp_size
```

python/sglang/srt/model_executor/forward_batch_info.py

修改 prepare_mlp_sync_batch 方法，移除硬编码的 attn_cp_size * 2 并对齐到新函数。

```
def prepare_mlp_sync_batch(self, model_runner: ModelRunner):
    from sglang.srt.batch_overlap.two_batch_overlap import TboForwardBatchPreparer

    # 本地导入：模块级别导入 cp_utils 会导致循环引用 (#27014)
    from sglang.srt.layers.utils.cp_utils import get_cp_padding_align_size

    assert self.global_num_tokens_cpu is not None
    assert self.global_num_tokens_for_logprob_cpu is not None

    global_num_tokens = self.global_num_tokens_cpu
    sync_group_size = len(global_num_tokens)
    attn_tp_size = get_attention_tp_size()
```

```
    for i in range(sync_group_size):
        global_num_tokens[i] = ceil_align(global_num_tokens[i], attn_tp_size)

    # 确保每个 rank 有相同 token 数以进行集合通信。
    # Zigzag (in-seq-split) CP 填充到 2 * attn_cp_size 以平衡负载；
    # 其他 CP 模式填充到 attn_cp_size；CP 关闭时不填充（多余填充会
    # 破坏 EAGLE/MTP draft prefill，产生 NaN draft logits，见 #23269）。
    # FIXME(kpham-sgl): 重新审视使 draft prefill-extend 容忍填充的虚拟 token。
    cp_align_size = get_cp_padding_align_size()
    for i in range(sync_group_size):
        global_num_tokens[i] = ceil_align(global_num_tokens[i], cp_align_size)

    dp_padding_mode = DpPaddingMode.get_dp_padding_mode(
        self.is_extend_in_batch, global_num_tokens
    )
    self.dp_padding_mode = dp_padding_mode
    # ... 后续逻辑保持不变
```

python/sglang/srt/layers/attention/dsa/utils.py

修改 cal_padded_tokens 函数以保持与 prepare_mlp_sync_batch 一致的填充逻辑，避免 DSA 路径产生不一致。

```
def cal_padded_tokens(forward_batch: "ForwardBatch"):
    # 与 ForwardBatch.prepare_mlp_sync_batch 中的填充计算保持一致。
    from sglang.srt.layers.utils.cp_utils import get_cp_padding_align_size

    global_num_tokens = forward_batch.global_num_tokens_cpu.copy()
    sync_group_size = len(global_num_tokens)
    attn_cp_size = get_attention_cp_size()
    # 必须与 ForwardBatch.prepare_mlp_sync_batch 中的 CP 填充一致。
    cp_align_size = get_cp_padding_align_size()
    for i in range(sync_group_size):
        global_num_tokens[i] = ceil_align(global_num_tokens[i], cp_align_size)
    # ... 后续逻辑保持不变
```

评论区精华

Reviewer kpham-sgl 提出了两点关键设计反馈：

- 要求将偶数填充仅限制在 zigzag (in-seq-split) 模式，因为 round-robin CP 只需要 cp_size 对齐。作者通过将 2 * attn_cp_size 提取到 get_cp_padding_align_size 中并仅在 zigzag 模式下返回该值来响应。
- 指出 attn_cp_size == 1 时不会进入任何 CP 模式，因此可以简化函数实现。作者移除了早期返回和 cp_align_size > 1 守卫，依赖于 ceil_align 在单位对齐时无操作。
- 将偶数填充仅限制在 zigzag 模式 (design): 作者将 2 * attn_cp_size 提取到 get_cp_padding_align_size 中，仅在 zigzag 模式返回该值，并简化了实现：移除了早期返回和 cp_align_size > 1 守卫。

- 移除不必要的 if 守卫 (style): 作者移除了多余的 if 守卫，直接使用 *get_cp_padding_align_size*() 的返回值。

风险与影响

- 风险：低风险。核心修复是条件化填充逻辑，仅在 CP 开启时激活，且 CP 路径行为不变。主要风险在于：
  - 循环导入风险：forward_batch_info.py 中对 cp_utils 的本地导入已通过注释说明并正确处理。
  - DSA 路径一致性：*cal_padded_tokens* 已同步更新，但若未来 DSA 路径引入新的对齐需求，需要保持同步。
  - 缺少单元测试：本 PR 未新增测试用例，依赖现有的 nightly 测试（已通过）。
  - 影响：影响范围小，仅针对 DP-attention + 推测解码 + CP 关闭的配置。修复后该配置下 GSM8K 准确率从接近 0 提升至 ~0.97，接受长度 ~3.0。对 CP 开启的配置无任何影响（行为不变）。
  - 风险标记：循环导入风险，缺少单元测试覆盖，核心路径变更

关联脉络

- PR #23269 Support batch size > 1 when enable CP: 本 PR 修复了 #23269 引入的回归：将 global_num_tokens 硬编码对齐到 attn_cp_size * 2，在 CP 关闭时导致 NaN 崩溃。