PR #35424 完整报告

sgl-project/sglang

[Fix] Scale the req_to_token row headroom by attn_dcp_size

合并时间：2026-08-19 15:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35424

执行摘要

- 一句话：按 DCP 规模缩放 req_to_token 行头空间，修复边界越界
- 推荐动作：该 PR 值得精读，尤其是理解 DCP 场景下 allocator 页大小与头空间计算的一致性。值得注意的是，它引入了一个统一访问器，后续相关计算应复用该函数以避免类似问题。同时建议作者补充针对 DCP 的单元测试，覆盖接近上下文限制的场景。

功能与动机

在 DCP 场景下，allocator 以 page_size * attn_dcp_size 作为页大小构建（_build_token_to_kv_pool_allocator），且行会被 allocator 的页向上舍入，但头空间仅根据调度页大小计算，导致接近上下文限制时舍入后的写入落入相邻行。PR body 明确说明：'Under DCP the allocator is built with page_size * attn_dcp_size... but the headroom was sized from the scheduled page size alone, so near the context limit the rounded write lands in the neighbor row.'

实现拆解

实现拆解：

1. 在 python/sglang/srt/mem_cache/allocation_sizing.py 中新增 get_alloc_page_size() 函数，从 runtime_context 导入 get_parallel，返回 get_schedule().page_size * get_parallel().attn_dcp_size，并注明镜像 _build_token_to_kv_pool_allocator 的 DCP 分支，作为平台 allocator 的上界。
2. 将 get_alloc_len_per_decode 中的 page_size = get_schedule().page_size 替换为 get_alloc_page_size()，确保 spec 解码分配的页对齐计算使用实际 alloc 页大小。
3. 将 get_req_to_token_extra_context_len 中的 page_size 同样替换为 get_alloc_page_size()，使头空间计算基于实际页大小，避免越界。
4. 无测试文件变更，但 CI 中 rerun 了 spec 相关测试（test_spec_eagle_topk_page.py 等），说明通过现有测试验证。

关键文件：

- python/sglang/srt/mem_cache/allocation_sizing.py（模块 内存分配；类别 source；类型 core-logic；符号 get_alloc_page_size）：核心改动文件，新增 get_alloc_page_size 并替换两处 page_size 来源，修复 DCP 下头空间不足导致的越界问题。

关键符号：get_alloc_page_size, get_alloc_len_per_decode, get_req_to_token_extra_context_len

关键源码片段

python/sglang/srt/mem_cache/allocation_sizing.py

核心改动文件，新增 get_alloc_page_size 并替换两处 page_size 来源，修复 DCP 下头空间不足导致的越界问题。

```
# python/sglang/srt/mem_cache/allocation_sizing.py

def get_alloc_page_size() -> int:
    # Mirrors _build_token_to_kv_pool_allocator's DCP branch; the platform
    # allocators that skip it page smaller, so this is an upper bound for them.
    return get_schedule().page_size * get_parallel().attn_dcp_size


def get_alloc_len_per_decode() -> int:
    """KV length one request may allocate in a single decode step."""
    # ... 省略 spec 逻辑 ...
    page_size = get_alloc_page_size() # 现在使用实际 alloc 页大小
    # ... 其余逻辑 ...


def get_req_to_token_extra_context_len() -> int:
    """req_to_token row headroom beyond the model context length."""
    # FIXME(lsyin): temporary fix for the context length issue under spec decoding
    extra = 4 + (max_speculative_num_draft_tokens() or 0)
    page_size = get_alloc_page_size() # 使用 alloc 页大小，避免 DCP 下越界
    if get_spec().speculative_algorithm is not None and page_size > 1:
        # 接近上下文限制时，对齐 reserve 可能超出 page_size - 1，
        # 若无头空间则写入会静默落到相邻行。
        extra = max(extra, get_alloc_reserve_per_decode() + page_size - 1)
    return extra
```

评论区精华

Review 评论中，kpham-sgl 指出必须使用 allocator 的页大小，即 get_schedule().page_size * get_parallel().attn_dcp_size，该评论被采纳。另一条评论要求 double check 调用方是否已传入 allocator 页大小，但未留下结论，属于已解决的疑虑。

- 必须使用 allocator 页大小 (correctness): 已被采纳，新增 get_alloc_page_size 统一使用该公式。
- double check 调用方页大小 (question): 已确认并修复，未留下进一步争议。

风险与影响

- 风险：风险点：

1. 仅修改了 allocation_sizing.py，但未更新任何测试，缺少针对 DCP 场景的越界回归测试，风险在于现有测试可能未覆盖到 DCP + 接近上下文限制的场景。
2. get_alloc_page_size 假设 attn_dcp_size 始终可用，若某些平台 attr 缺失可能抛出异常，但代码注释表明平台 allocator 会跳过更小页，因此风险较低。
3. 改动仅影响 DCP 场景，非 DCP 时 attn_dcp_size=1 行为不变，回归风险低。 - 影响：影响范围：主要影响启用 DCP（attn_dcp_size > 1）的部署，修复潜在的显存越界写入，避免数据损坏和崩溃。对无 DCP 用户无影响。对团队而言，这是一个低风险但必要的修复，涉及核心内存分配逻辑，需要谨慎验证。 - 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #35401 Base PR for this stack (未提供详细标题): PR body 提到 'Stacks on #35401'，表明本 PR 依赖或堆叠在该 PR 之上，共享相关内存分配逻辑。