PR #29343 完整报告

sgl-project/sglang

[dflash] fa3/fa4: device-side page table; drop seq_lens_cpu D2H sync

合并时间：2026-06-29 09:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29343

执行摘要

- 一句话：消除 dflash+fa3 中 seq_lens_cpu D2H 同步，设备端构建页表
- 推荐动作：该 PR 值得精读，尤其是以下设计点：
  1. 设备端自保护 kernel：build_trtllm_mha_page_table 通过 cache_seqlens 限制写入列，使静态上界成为可能。
  2. 渐进式作用域控制：先将优化限制到 dflash（风险最低），再逐步推广，是大型重构的稳健思路。
  3. 冷路径回退策略：_host_max_seq_len 辅助函数隔离了"可能需要主机最大值"的逻辑，使热路径保持纯净。
  4. 测试中的 sentinel 方法：直接检查未写入列是否保持 sentinel，精确验证 kernel 的写边界。

功能与动机

dflash+fa3 仍因 fa3 声明 needs_cpu_seq_lens = True 并读取 seq_lens_cpu.max().item() 用于动态页面表大小调整而付出 resolve_seq_lens_cpu D2H + synchronize() 开销（每次前向迭代约 370us）。trtllm_mha 在 #28106 中以设备端页面表构建器解决了该问题。此 PR 将同一模式复用于 fa3，且仅作用于 dflash。

实现拆解

1. 导入与初始化调整：在 flashattention_backend.py 中导入 build_trtllm_mha_page_table（来自 trtllm_mha_page_table）和 SpeculativeAlgorithm 枚举。在 FlashAttentionBackend.__init__ 中计算静态 self.max_num_pages（基于 max_context_len / page_size），并设置 self.needs_cpu_seq_lens = not is_dflash()，使 dflash 绕过 CPU mirror。
2. 设备端页面表构建热路径：在 init_forward_metadata 中，对 topk=1 且 forward_mode 为 decode/draft-decode/verify 的分支，调用 build_trtllm_mha_page_table 填充页面表，使用设备端 cache_seqlens 自保护写入边界。max_seq_len_k 设为静态 max_context_len，fa3 kernel 实际不读取该值。
3. 冷路径主机端最大值辅助：抽取 _host_max_seq_len 函数，优先使用 forward_batch.seq_lens_cpu（若非 None），否则调用 .cpu() 或回退至 max_context_len。在 topk>1、draft-extend、prefill-aware SWA decode 等路径中替换

原 seq_lens_cpu.max().item() 调用。

4. 路径逐一适配：依次调整 prefill-aware SWA decode、SWA extend、scheduler_metadata 等路径，确保 seq_lens_cpu 可能为 None 时不会访问。
5. 测试增强：在 test_trtllm_mha_page_table.py 中新增 _run_self_guard_case 和 test_writes_bounded_by_cache_seqlens，用 sentinel 填充静态缓冲区，运行 kernel 后验证每个请求的已用列正确且未用列保留 sentinel，证明 kernel 不越界写入。

关键文件：

- python/sglang/srt/layers/attention/flashattention_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 _host_max_seq_len）：核心变更文件：新增设备端页面表构建逻辑、_host_max_seq_len 辅助函数、needs_cpu_seq_lens 条件判断，修改 init_forward_metadata 各热 / 冷路径。
- test/registered/attention/test_trtllm_mha_page_table.py（模块 页表构建；类别 test；类型 test-coverage；符号 _run_self_guard_case, test_writes_bounded_by_cache_seqlens）：测试覆盖：新增 write-bounded invariant 测试，确保设备端 kernel 不会写入超出 cache_seqlens 的列，为 GPU-only 路径提供安全保证。

关键符号：FlashAttentionBackend.init, init_forward_metadata, _host_max_seq_len

关键源码片段

python/sglang/srt/layers/attention/flashattention_backend.py

核心变更文件：新增设备端页面表构建逻辑、_host_max_seq_len 辅助函数、needs_cpu_seq_lens 条件判断，修改 init_forward_metadata 各热 / 冷路径。

```
# 辅助函数：返回主机端最大 seq len，用于冷路径（topk>1、draft-extend 等）
# 优先使用 forward_batch.seq_lens_cpu（若已由 upstream 发布），否则 fallback 到 max_context_
len
@staticmethod
def _host_max_seq_len(forward_batch: ForwardBatch, max_context_len: int) -> int:
    if forward_batch.seq_lens_cpu is not None:
        return forward_batch.seq_lens_cpu.max().item()
    # seq_lens_cpu 为 None（GPU-only 路径）时，回退到静态上界
    # 调用者需确保该路径下 kernel 不会读取超出实际序列的位置
    return max_context_len

# FlashAttentionBackend.__init__ 中的关键片段（简化）
def __init__(self, ...):
    ...
    # 静态上界：页表 buffer 宽度固定为最大可能页数
    self.max_num_pages = (self.max_context_len + self.page_size - 1) // self.page_size
    # 仅 dflash 绕过 CPU mirror；EAGLE/MTP 保持原样
    self.needs_cpu_seq_lens = not SpeculativeAlgorithm.from_string(
        model_runner.server_args.speculative_algorithm
    ).is_dflash()
    ...
```

```
# init_forward_metadata 中的热路径（decode, topk=1），使用设备端构建
if forward_batch.forward_mode.is_decode_or_idle():
    if not self.needs_cpu_seq_lens and forward_batch.topk <= 1:
        # 设备端构建：call build_trtllm_mha_page_table, max_seq_len_k 设为静态值
        build_trtllm_mha_page_table(
            req_to_token=self.req_to_token,
            req_pool_indices=forward_batch.req_pool_indices,
            cache_seqlens=forward_batch.seq_lens.to(torch.int32),
            page_table=page_table_buffer,
            page_size=self.page_size,
        )
        metadata.max_seq_len_k = self.max_context_len
    else:
        # 冷路径：使用 _host_max_seq_len 获取主机最大值
        metadata.max_seq_len_k = (
            self._host_max_seq_len(forward_batch, self.max_context_len)
            + (self.speculative_step_id + 1)
        )
```

评论区精华

PR 无 reviewer 评论，作者自行合并。关键设计决策体现在 commit 历史中：

- 最初实现设备端构建后，逐步提取 _host_max_seq_len 辅助函数以处理冷路径（commit 569935ca）。
- 最终将 needs_cpu_seq_lens 作用域限制为仅 dflash（commit 25f4857a），避免影响 EAGLE/MTP/standalone，降低风险。PR body 明确了后续将借由 #29589 将同步消除推广至所有 spec 算法。
- 设备端页表构建范围限制：仅 dflash 启用 GPU-only 路径 (design): 限制到 dflash，后续通过 #29589 推广至所有 spec 算法。

风险与影响

- 风险：
  1. 作用域判断风险：needs_cpu_seq_lens = not is_dflash() 依赖 SpeculativeAlgorithm.from_string 正确识别 dflash。若未来算法枚举变化或配置错误，可能导致非 dflash 意外跳过 CPU mirror 而引发错误。
  2. 冷路径回退隐式假设：_host_max_seq_len 在 seq_lens_cpu 为 None 时回退到 max_context_len。此回退假设 kernel 能通过 page_table 自保护，但若后续代码变更使得 max_seq_len_k 被用于实际大小，可能超出真实序列长度。目前仅 fa3 kernel 不读该值，但需保持一致。
  3. 测试覆盖不足：新增测试仅覆盖设备端 kernel 的 write-bounded invariant，未覆盖启用 GPU-only 路径后的端到端推理正确性（如采样结果一致性）。
  4. 显存占用：静态 max_num_pages 固定分配最大可能页面表，可能略高于运行时所需，但该 buffer 在 CUDA graph 中可复用，影响可接受。 - 影响：用户侧：dflash 用户每次 decode 迭代消除约 370us 同步开销，显著提升前向 occupancy 和吞吐量。非

dflash 用户无影响。系统侧：fa3/falcon 后端新增一个 Triton kernel 调用，引入少量固定大小的页面表 buffer（max_context_len / page_size 行）。整体改动小，回归风险低。团队侧：为后续 #29589（EAGLE/MTP 同步消除）提供了参考范式和测试基础设施。

- 风险标记：核心路径变更，依赖后续 PR 扩展，冷路径回退隐式假设

关联脉络

- PR #29232 [Spec] Replace shared-infra dflash special-cases with capabilities (WAR barrier + seq_lens_cpu): 前序重构，统一了 dflash 的能力模型，为本 PR 的页表优化奠定基础（提供 FutureMap 等基础设施）。