PR #28106 完整报告

sgl-project/sglang

[attn backend] Make seq_lens_cpu optional in trtllm_mha backend

合并时间：2026-06-18 07:24

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28106

执行摘要

- 一句话：trtllm_mha 后端移除 CPU seq_lens 同步，提升推测解码性能
- 推荐动作：值得精读：该 PR 清晰地展示了一个通过设备端计算替代主机端同步的经典优化案例。设计决策（静态上界、设备端保护、Python 包装器）和 review 中的讨论（eager 路径同步移除）都很有参考价值。对于理解 SGLang 注意力后端的同步模型和推测解码优化很有帮助。

功能与动机

trtllm_mha 后端原先设置 needs_cpu_seq_lens=True，导致在 spec-v2 中每个 decode 步骤都需要 CUDA 同步，因为 seq_lens 仅在 GPU 工作完成后才最终确定，CPU 必须等待后才能设置下一步的元数据并启动核函数。这阻碍了 CPU 与 GPU 异步执行，造成 GPU 空闲。

实现拆解

1. 新增设备端页表构建内核：在 python/sglang/srt/layers/attention/triton_ops/trtllm_mha_page_table.py 中实现 Triton 内核 create_trtllm_mha_kv_indices_triton，直接利用 GPU 上的 seq_lens 为每个请求生成块 ID，避免主机端 max() 同步。内核的网格大小基于静态上界 max_num_pages，但每个程序根据实际长度自行保护，使得工作量由真实长度决定。
2. 修改后端主文件：在 trtllm_mha_backend.py 中将 needs_cpu_seq_lens 设为 False，在 __init__ 中计算 max_num_pages。新增 _fill_page_table_device 方法调用 build_trtllm_mha_page_table 包装内核，同时处理 SWA 模型的页表翻译。
3. 消除 eager 路径同步：根据 review 讨论，在 init_forward_metadata 方法中也移除了原先的 seq_lens.max().item() 同步，改用设备端构建，统一了 CUDA graph 和 eager 模式的行为。
4. 新增自动化测试：新增 test/registered/attention/test_trtllm_mha_page_table.py，通过对比主机端参考实现（带 max 同步）与设备端内核的输出，验证页表构建的正确性，包括带 SWA 的场景。测试覆盖多种配置组合，并注册为 CI 测试。

关键文件：

- python/sglang/srt/layers/attention/trtllm_mha_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 needs_cpu_seq_lens, _fill_page_table_device, _maybe_translate_swa, _copy_swa_page_table）：核心后端文件，类属性

needs_cpu_seq_lens = False 取消同步，新增 _fill_page_table_device 方法调用设备端内核，并在 init_forward_metadata 中移除同步。

- test/registered/attention/test_trtllm_mha_page_table.py（模块 页表测试；类别 test；类型 test-coverage；符号 _build_page_table_reference, _build_page_table_kernel, TestTrtllmMhaPageTable, _run_case）：新增测试文件，验证设备端页表构建与主机端参考实现一致，确保正确性。
- python/sglang/srt/layers/attention/triton_ops/trtllm_mha_page_table.py（模块 页表内核；类别 infra；类型 infrastructure；符号 get_num_mha_kv_index_blocks, create_trtllm_mha_kv_indices_triton, build_trtllm_mha_page_table）：新增 Triton 内核基础设施，负责设备端页表构建。

关键符号：_fill_page_table_device, build_trtllm_mha_page_table, create_trtllm_mha_kv_indices_triton, _build_page_table_reference, _build_page_table_kernel, test_matches_reference_gather, test_swa_matches_reference

关键源码片段

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

核心后端文件，类属性 needs_cpu_seq_lens = False 取消同步，新增 _fill_page_table_device 方法调用设备端内核，并在 init_forward_metadata 中移除同步。

```
class TRTLLMHAAttnBackend(FlashInferAttnBackend):
    # 设为 False 表示不再需要 CPU 上的 seq_lens，页表在设备侧构建
    needs_cpu_seq_lens: bool = False

    def __init__(
        self,
        model_runner: ModelRunner,
        skip_prefill: bool = False,
        kv_indptr_buf: Optional[torch.Tensor] = None,
        kv_last_page_len_buf: Optional[torch.Tensor] = None,
        speculative_step_id: int = 0,
    ):
        ...
        # 静态上界：max_context_len 除以 page_size 向上取整，用于页面表缓冲区宽度
        self.max_num_pages = (
            self.max_context_len + self.page_size - 1
        ) // self.page_size
        ...

    def _fill_page_table_device(
        self,
        metadata: TRTLLMMHAMetadata,
        req_pool_indices: torch.Tensor,
        cache_seqlens: torch.Tensor,
    ):
        """从设备侧 per-request 的 KV 长度构建页表（无同步）。
```

```
        调用 `build_trtllm_mha_page_table` 内核，该内核直接从 GPU 的
        `cache_seqlens` 读取每个请求的实际 token 数，为每个请求写入
        对应 block id 到 `metadata.page_table` ；对于 SWA 模型，还会
        通过 `full_to_swa` 查找表同时计算 SWA 页表。
        """
        build_trtllm_mha_page_table(
            req_to_token=self.req_to_token,
            req_pool_indices=req_pool_indices,
            cache_seqlens=cache_seqlens,
            page_table=metadata.page_table,
            page_size=self.page_size,
            swa_page_table=metadata.swa_page_table,
            full_to_swa=self._swa_kv_pool.get_full_to_swa() if self._swa_kv_pool else None,
        )
```

test/registered/attention/test_trtllm_mha_page_table.py

新增测试文件，验证设备端页表构建与主机端参考实现一致，确保正确性。

```
def _build_page_table_kernel(
    req_to_token: torch.Tensor,
    req_pool_indices: torch.Tensor,
    cache_seqlens: torch.Tensor,
    page_size: int,
    max_num_pages: int,
    full_to_swa: Optional[torch.Tensor] = None,
) -> tuple[torch.Tensor, Optional[torch.Tensor]]:
    """设备端页表构建：调用 Triton 内核，输出尺寸由 max_num_pages 确定。"""
    dev = req_to_token.device
    bs = req_pool_indices.shape[0]
    page_table = torch.zeros((bs, max_num_pages), dtype=torch.int32, device=dev)
    swa_page_table = (
        torch.zeros((bs, max_num_pages), dtype=torch.int32, device=dev)
        if full_to_swa is not None else None
    )
    build_trtllm_mha_page_table(
        req_to_token=req_to_token,
        req_pool_indices=req_pool_indices,
        cache_seqlens=cache_seqlens,
        page_table=page_table,
        page_size=page_size,
        swa_page_table=swa_page_table,
        full_to_swa=full_to_swa,
    )
    return page_table, swa_page_table

class TestTrtllmMhaPageTable(CustomTestCase):
    def _run_case(self, max_context_len, page_size, num_reqs, bs, swa=False):
        # 随机生成输入，对比内核与参考实现
        pt_kernel, swa_kernel = _build_page_table_kernel(...)
```

```
    pt_ref, swa_ref = _build_page_table_reference(...)
    for i in range(bs):
        npages = (int(cache_seqlens[i].item()) + page_size - 1) // page_size
        # 仅对比该请求实际使用的列（前 npages 列）
        self.assertTrue(torch.equal(pt_kernel[i, :npages], pt_ref[i, :npages]))
        if swa:
            self.assertTrue(torch.equal(swa_kernel[i, :npages], swa_ref[i, :npages]))
```

评论区精华

- 类型提示：jasonjk-park 建议为测试函数添加类型提示（未明确解决）。
- 函数命名：jasonjk-park 指出 _build_page_table_kernel 可能与 Triton 内核混淆，作者将其重命名为 _build_page_table_reference / _build_page_table_kernel 以区分（已解决）。
- 直接调用 vs 包装器：jasonjk-park 质疑为何不提供 Python 包装器，作者先引用 MLA 后端的风格，后续添加了 build_trtllm_mha_page_table 包装函数（已解决）。
- 设备同步：merrymercy 询问 *eager* 路径是否仍触发同步，作者说明仅在 CUDA graph 中移除，随后 merrymercy 要求也移除 *eager* 路径的同步，作者遵从并移除（已解决）。
- 测试函数类型提示 (style): 未明确解决，但 PR 已合并，类型提示未添加。
- 函数命名混淆 (style): 作者将原函数重命名为 _build_page_table_reference / _build_page_table_kernel 以区分。
- 直接调用 Triton 内核 vs Python 包装器 (design): 添加了 build_trtllm_mha_page_table Python 包装器。
- Eager 路径的设备同步移除 (performance): 作者移除 *eager* 路径的同步，统一了 CUDA *graph* 和 *eager* 的行为。

风险与影响

- 风险：
  1. 新 Triton 内核的硬件兼容性：该内核仅用于 Blackwell (sm100) 架构，因为 trtllm_mha 后端本身限制。非 sm100 设备不会受影响。
  2. 非推测解码使用场景：修改影响了所有使用 trtllm_mha 后端的请求，不限于推测解码。虽然所有测试通过，但缺少非推测场景的专项测试。
  3. SWA 页表翻译正确性：当使用滑动窗口注意力时，内核通过 full_to_swa 表进行翻译，新测试覆盖了该路径，但生产环境中的极端值可能未覆盖。
  4. *eager* 路径同步移除：原先 *eager* 路径依赖 seq_lens.max().item()，现在完全移除，如果存在未知的依赖此 host 最大值的分支，可能引入问题。 - 影响：用户影响：使用 trtllm_mha 注意力后端且启用推测解码（如 EAGLE3）的用户将获得显著的性能提升，TPOT 降低 10-15%，吞吐量提升类似。对于不使用推测解码的用户，该变更无功能影响，但同样移除了多余的同步，可能带来微小改善。系统影响：减少了 GPU 与 CPU 间的同步点，允许 CPU 更早地启动后续步骤，提高 GPU 利用率和整体吞吐。团队影响：该模式与 trtllm_mla 和 triton 后端保持一致，统一了不同后端的架构模式，降低了维护复杂度。
- 风险标记：新 Triton 内核依赖 , *eager* 路径同步移除 , Blackwell 架构定制

关联脉络

- 暂无明显关联 PR