PR #27091 完整报告

sgl-project/sglang

Unify full→SWA index translation in init_forward_metadata; drop pool caches

合并时间：2026-06-04 07:12

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27091

执行摘要

- 一句话：统一 full→SWA 翻译并移除池缓存
- 推荐动作：建议精读。该 PR 展示了一次精心设计的状态管理重构：将原本散布的多处缓存和失效逻辑统一到一个单一的、在已知时机计算的点上。其中的设计决策（在 graph init 中记录而非在 connect 回调中处理）以及 fallback 机制的实现值得学习。同时，关于 draft-extend 路径的修复过程显示了在 CUDA graph 环境中保持正确性的典型挑战。

功能与动机

PR body 指出 full→SWA 索引翻译先前以两种不一致的方式触发：DSV4 写路径在首个 SWA 层懒翻译并缓存，读路径在 SWAKVPool 中按需翻译并 memoized，并需要通过 invalidate_loc_cache 保持一致性。本 PR 统一设计：在 attention 后端的 metadata init 中计算一次，存储在 metadata 上，然后删除懒缓存和 invalidate_loc_cache 机制，以简化代码并消除缓存一致性风险。

实现拆解

实现步骤

1. 在 attention 后端 metadata init 中计算一次 在 DSV4AttnMetadata 中添加 swa_out_cache_loc 字段，在 init_forward_metadata_in_graph 中通过 translate_loc_from_full_to_swa 计算并缓存，同时转换为 int32 以满足 FlashMLA 等 kernel 要求。
2. 移除旧缓存机制 - 删除 DeepSeekV4TokenToKVPool 中的 get_cached_swa_loc 方法、SGLANG_OPT_CACHE_SWA_TRANSLATION 环境变量 gating、以及 invalidate_loc_cache 方法。 - 删除 SWAKVPool 中基于 (data_ptr, numel) 的缓存和 invalidate_loc_cache 方法。 - 移除所有调用 invalidate_loc_cache 的位置，包括 cuda_graph_runner、piecewise_cuda_graph_runner、breakable_cuda_graph_runner、model_runner、Eagle/MTP draft runners 等。
3. 提供 get_swa_out_cache_loc 帮助函数 在 DSV4 backend 中添加 get_swa_out_cache_loc，优先返回 metadata 上的缓存值；在特定场景（eager IDLE、draft-extend graph runner、长度因 DP padding 变化）时回退到 store-time 实时翻译，确保正确性。

4. 修改消费者 - 模型层 deepseek_v4.py 中的 store_cache 等方法改用 get_swa_out_cache_loc。 - FlashAttention、FlashInfer、Triton 等后端在构建 metadata 时完成 int64→int32 转换。
5. 测试调整 - 新增 TestDSV4SwaOutCacheLocResolution 测试类，覆盖 get_swa_out_cache_loc 的四个分支（无 metadata、缓存命中、形状过期、IDLE 模式）。 - 删除四个手动测试文件，这些文件验证的是被移除的缓存机制。

关键文件：

- python/sglang/srt/layers/attention/deepseek_v4_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 get_swa_out_cache_loc, init_forward_metadata_in_graph, store_cache）：核心变更：添加 swa_out_cache_loc 字段和 get_swa_out_cache_loc 函数，在 init_forward_metadata_in_graph 中计算 SWA 翻译并缓存到 metadata，同时修改 store_cache 等方法使用新接口。
- python/sglang/srt/layers/attention/deepseek_v4_backend_hip_radix.py（模块 注意力后端；类别 source；类型 core-logic；符号 get_swa_out_cache_loc, init_forward_metadata_in_graph, store_cache）：与 deepseek_v4_backend.py 相同的变更，适用于 HIP radix 后端。
- python/sglang/srt/mem_cache/deepseek_v4_memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 get_cached_swa_loc, invalidate_loc_cache, register_mapping）：移除缓存机制：删除 get_cached_swa_loc 方法、SGLANG_OPT_CACHE_SWA_TRANSLATION 条件分支以及 invalidate_loc_cache 方法。
- python/sglang/srt/mem_cache/swa_memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 invalidate_loc_cache, translate_loc_from_full_to_swa）：移除 SWAKVPool 中基于 (data_ptr, numel) 的缓存以及 invalidate_loc_cache 方法。
- test/registered/attention/unittests/dsv4/test_deepseek_v4.py（模块 注意力测试；类别 test；类型 test-coverage；符号 TestDSV4SwaOutCacheLocResolution）：新增 TestDSV4SwaOutCacheLocResolution 测试类，覆盖 get_swa_out_cache_loc 的四个关键分支。
- python/sglang/srt/models/deepseek_v4.py（模块 模型层；类别 source；类型 data-contract）：修改 KV-store 方法，使用 get_swa_out_cache_loc 代替直接访问 forward_batch.out_cache_loc。
- python/sglang/srt/model_executor/cuda_graph_runner.py（模块 模型执行器；类别 source；类型 data-contract）：移除 invalidate_loc_cache 调用。

关键符号：get_swa_out_cache_loc, invalidate_loc_cache, get_cached_swa_loc, init_forward_metadata_in_graph, store_cache

关键源码片段

python/sglang/srt/layers/attention/deepseek_v4_backend.py

核心变更：添加 swa_out_cache_loc 字段和 get_swa_out_cache_loc 函数，在 init_forward_metadata_in_graph 中计算 SWA 翻译并缓存到 metadata，同时修改 store_cache 等方法使用新接口。

```
def get_swa_out_cache_loc(self, forward_batch: ForwardBatch) -> Optional[torch.Tensor]:
    """返回 DSV4 写入 SWA KV 缓存的目标位置（SWA 索引空间）。

    优先使用 metadata 上的缓存值；若不适用则回退到实时翻译。
    """
    metadata = self.forward_metadata
    if not isinstance(metadata, DSV4Metadata):
        return None
    cached = metadata.core_attn_metadata.swa_out_cache_loc
    if cached is not None:
        # 检查缓存是否仍然有效
        if forward_batch.forward_mode.is_decode_or_idle():
            if forward_batch.forward_mode.is_idle():
                # IDLE 模式没有真实 out_cache_loc，永远回退
                return None
            # 缓存长度必须与当前 batch 的 out_cache_loc 长度匹配
            if (forward_batch.out_cache_loc is not None and
                cached.shape[0] == forward_batch.out_cache_loc.shape[0]):
                return cached
    # 回退：实时翻译并转换为 int32
    if forward_batch.out_cache_loc is not None:
        return self.token_to_kv_pool.translate_loc_from_full_to_swa(
            forward_batch.out_cache_loc
        ).to(torch.int32)
    return None
```

评论区精华

Review 中 chatgpt-codex-connector[bot] 提出两个 P1 问题，指出在 DRAFT_EXTEND CUDA graph bucket 中，init_forward_metadata_out_graph 调用时未传入 out_cache_loc，导致 swa_out_cache_loc 由合成零值翻译而来，后续 SWA 缓存写入错误的位置，从而损坏草稿 KV 缓存。作者后续提交 4c3c8b8 添加了 get_swa_out_cache_loc 帮助函数，通过长度校验和回退逻辑解决了该问题。该问题已在最终版本中修复。

- DRAFT_EXTEND 路径 SWA 缓存位置错误 (correctness): 作者在后续提交 4c3c8b8 中添加 get_swa_out_cache_loc 帮助函数，通过长度校验和回退逻辑解决了该问题。
- HIP radix 后端相同问题 (correctness): 同步修复，与 CUDA 后端一致的修改。

风险与影响

- 风险：
  1. DRAFT_EXTEND 路径回归：如果 get_swa_out_cache_loc 中的回退逻辑不触发，可能导致缓存写入零位置；但单元测试和 e2e 测试已覆盖。
  2. int32 转换遗漏：某些后端 kernel 要求 int32，如果 metadata 构造时未进行转换（如 FA3、flash_mla），会导致 kernel 崩溃；已排查并在各后端转换。
  3. CUDA graph 重放：swa_out_cache_loc 在 graph capture 时记录，但 replay 时 out_cache_loc 可能被 rebind（spec-v2、DP padding），需要确保 metadata 层面的

值更新；通过 assign_fields（不跨 replay 复制）和 _CP_GLOBAL_FIELDS（全局字段）区分。

4. 移除的 invalidate_loc_cache 影响：所有调用点已清理，但若某个外部补丁仍依赖此接口将导致 AttributeError。
5. 测试覆盖：新测试覆盖了回退路径，被删除的测试已由更集成化的测试替代。 - 影响：对用户：无用户可见变更，模型输出和性能不变。对系统：简化了代码，减少了 per-iteration 的缓存管理操作，可能略微提升性能。对团队：需要关注后续若有新的 SWA 缓存需求，需遵循本 PR 的单一翻译点模式。影响范围包括 DSV4 attention backend（CUDA + HIP）、memory pool 层以及多个 CUDA graph runner。 - 风险标记：DSV4 核心路径变更，CUDA graph 重放正确性，DRAFT_EXTEND 路径修复，int32 类型转换风险

关联脉络

- PR #26997 Reland spec v2 tree drafting (eagle topk>1) with page_size==1: 该 PR 引入了 spec v2 多步草稿功能，本 PR 中的 draft-extend 路径修复（传递 real out_cache_loc）正是为了支持 spec v2 的 DP 填充后 rebind 场景。
- PR #26742 [refactor] Unify CUDA graph runner input buffers behind CudaGraphBufferRegistry: 均为 CUDA graph 相关重构，本 PR 中的 assign_fields 和 _CP_GLOBAL_FIELDS 设计与该 PR 的寄存器模式类似，体现了统一管理图执行中 buffer 复用的思路。
- PR #27072 hicache: publish split write-through fragments: HiCache 的写入路径与本 PR 的 SWA 翻译相关，本 PR 移除了 SWA 翻译的缓存后，HiCache 的 write-through 行为保持不变，但相关代码已简化。