PR #27978 完整报告

sgl-project/sglang

[AMD] Cache unified_kv swa_loc once per step instead of per layer

合并时间：2026-06-12 14:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27978

执行摘要

- 一句话：统一 KV 缓存 SWA 位置计算每步仅一次
- 推荐动作：值得阅读，展示了如何通过简单的缓存减少热路径上的重复计算以获得可观的性能收益。设计模式（元数据缓存 + 回退）可推广到其他层间共享的计算。

功能与动机

在 unified_kv 解码路径中，swa_loc（SWA 环写入目标，req_slot * ring + pos % ring）原本在每个层的 KV store 中重复计算。该值与 layer_id 无关，仅依赖于当前前向的请求槽位和 token 位置，因此在解码热路径上重复计算是冗余的。

实现拆解

1. 新增元数据字段 unified_swa_loc：在 DSV4AttnMetadata 数据类中新增 unified_swa_loc: Optional[torch.Tensor] 字段，用于缓存每步计算的 SWA 位置，并添加到 assign_fields 中以确保在 CUDA graph 重放时被正确重新计算。
2. 单次计算并缓存：在 _attach_unified_kv_decode_streams 方法中，利用已有的 req_pool_indices 和 positions_casual 一次计算出 unified_swa_loc，赋值给 core.unified_swa_loc。
3. 提供缓存读取方法 get_unified_swa_loc：在 DeepseekV4HipRadixBackend 中新增该方法，优先返回缓存值（形状匹配且非 idle 状态），否则回退到实时计算（处理 prefill、extend、idle 或批重填充导致的形状不匹配）。
4. 简化模型层调用：在 MQALayer._forward_prepare 中移除原先每层计算 swa_loc 的代码，改为统一调用 attn_backend.get_unified_swa_loc(forward_batch)。

关键文件：

- 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_unified_swa_loc）：核心实现文件：新增 unified_swa_loc 元数据字段、在 _attach_unified_kv_decode_streams 中单次计算、提供 get_unified_swa_loc 方法并包含回退逻辑。
- python/sglang/srt/models/deepseek_v4.py（模块 模型层；类别 source；类型 data-contract）：调用方：在 _forward_prepare 中替换了原先每层计算 swa_loc 的代码为统一调用 attn_backend.get_unified_swa_loc。

关键符号：get_unified_swa_loc, _attach_unified_kv_decode_streams, _forward_prepare

关键源码片段

python/sglang/srt/layers/attention/deepseek_v4_backend_hip_radix.py

核心实现文件：新增 unified_swa_loc 元数据字段、在 _attach_unified_kv_decode_streams 中单次计算、提供 get_unified_swa_loc 方法并包含回退逻辑。

```
# python/sglang/srt/layers/attention/deepseek_v4_backend_hip_radix.py

@dataclass
class DSV4AttnMetadata:
    # ... 其他字段 ...

    # unified_kv: per-forward prebuilt ragged decode index
    # SWA ring write target (req_slot*ring + pos%ring), computed once per
    # forward in _attach_unified_kv_decode_streams, read by every layer's store.
    unified_swa_loc: Optional[torch.Tensor] = None # 新增字段
    # ... 其他 unified 索引字段 ...

    def copy_(self, other: "DSV4AttnMetadata") -> None:
        # ... 复制逻辑 ...
        assign_fields = [
            "swa_out_cache_loc",
            "unified_swa_loc", # 确保在 CUDA graph 重放时被重新计算
            "c1_flashmla_metadata",
            "c4_flashmla_metadata",
            "c128_flashmla_metadata",
        ]
        # ...

    def _attach_unified_kv_decode_streams(self, ...):
        # ... 构建索引流 ...
        # SWA ring write target, same value for every layer this forward.
        # Decode: N tokens == N reqs, positions already aligned (no repeat).
        req_slot = req_pool_indices[:N].to(torch.int64)
        core.unified_swa_loc = (
            req_slot * pool.unified_swa_ring_size +
            core.positions_casual.to(torch.int64) % pool.unified_swa_ring_size
        ).to(torch.int32)

    def get_unified_swa_loc(self, forward_batch: ForwardBatch) -> torch.Tensor:
        """SWA ring write target for unified_kv, shared by all layers.

        Fast path: 使用缓存值; Fallback: 实时计算以处理 prefill/extend/idle 等场景.
        """
        positions = forward_batch.positions
        core = getattr(self.forward_metadata, "core_attn_metadata", None)
        cached = core.unified_swa_loc if core is not None else None
```

```
if (cached is not None and not forward_batch.forward_mode.is_idle()
        and cached.shape[0] == positions.shape[0]):
    return cached # 形状匹配且非 idle，直接返回缓存
# 回退：实时计算
ring = self.token_to_kv_pool.unified_swa_ring_size
req_slot = forward_batch.req_pool_indices.to(torch.int64)
if req_slot.shape[0] != positions.shape[0]:
    req_slot = req_slot.repeat_interleave(
        positions.shape[0] // req_slot.shape[0]
    )
return (req_slot * ring + positions.to(torch.int64) % ring).to(torch.int32)
```

评论区精华

无 review 讨论。PR 由 HaiShaw 批准合并。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。主要变更集中在 AMD 后端的统一 KV 路径上，且有回退机制（形状不匹配或非 *decode* 模式时重新计算）。但缺少专门的单元测试覆盖（仅依赖现有的精度和性能测试）。
- 影响：影响范围限定于 AMD HIP Radix 后端下的 DeepSeek V4 模型，在启用 unified_kv 的解码路径上获得约 2.5% 的吞吐量提升。对 NVIDIA 后端无影响。
- 风险标记：仅 AMD 后端，缺少单元测试

关联脉络

- 暂无明显关联 PR