# PR #23146 完整报告

sgl-project/sglang

[AMD] Enable EAGLE speculative decoding for Qwen3.5 FP8 and MXFP4 models with aiter's unified attention

合并时间：2026-05-05 15:09

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23146

## 执行摘要

- 一句话：AMD 启用 EAGLE 统一注意力验证并修复 MXFP4 加载
- 推荐动作：值得精读，尤其关注注意力后端如何适配不同数据类型（MLA/non-MLA）和投机解码布局（ragged vs paged）。设计决策（如保持 radix-cache 分离）体现了模块化思维。建议后续补充单元测试覆盖新路径。

## 功能与动机

之前当 SGLANG_USE_AITER_UNIFIED_ATTN=1 配合 --speculative-algorithm EAGLE 时，非 MLA 且 topk==1 的 EAGLE 缺少 unified_attention 目标验证路径，被迫使用低效的自定义 mask 扩展注意力。此外，Quark 导出的 MXFP4 Qwen3.5 检查点中 MTP 模块为 bf16，但加载器误分配 MXFP4 packed 形状导致加载失败（issue #23113），且 --quantization quark 因未注册而被 CLI 拒绝。

## 实现拆解

1. aiter_backend.py 核心改造：在 AiterAttnBackend 中新增 _build_unified_page_table_from_spec 和 _build_verify_unified_metadata 方法，将 spec_info 的 ragged token 级索引转换为 unified_attention 所需的 2D 块级页表；在 forward_extend 中根据 _use_unified_verify 标志（非 MLA 且 topk==1 生效）将 target_verify 路径路由到 unified_attention 而非 extend_attention_fwd；预计算 qo_indptr_unified_decode 避免逐请求 cumsum。
2. 新增 Triton scatter kernel：在 aiter_unified_attention.py 中实现 scatter_ragged_to_page_table_kernel 和 scatter_req_to_token_to_page_table_kernel，高效完成 ragged 到 page_table 的并行转换，并支持 SWA 槽映射。
3. 滑动窗口支持：在 init_forward_metadata 和 _build_verify_unified_metadata 中传递 swa_page_table，根据 layer.sliding_window_size 设置 window_size 参数。
4. 修复 MTP 量化加载：在 qwen3_5_mtp.py 的 __init__ 中，检测 quark 量化配置的排除层列表，若包含 mtp.* 则跳过量化，确保线性层分配 bf16 形状。
5. 注册 quark 量化选项：在 server_args.py 的 QUANTIZATION_CHOICES 中添加 "quark" 以支持 CLI 输入。测试配套：本次变更未新增测试文件，依赖自有实验验证。

关键文件：

- python/sglang/srt/layers/attention/aiter_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 _build_unified_page_table_from_spec, _build_verify_unified_metadata, forward_extend, init_forward_metadata）：主要改动，新增 EAGLE 统一验证路径和页表转换逻辑，核心变更文件
- python/sglang/srt/layers/attention/triton_ops/aiter_unified_attention.py（模块 注意力 Kernel；类别 infra；类型 infrastructure；符号 scatter_ragged_to_page_table_kernel, scatter_req_to_token_to_page_table_kernel）：新增的 Triton scatter kernel，将 ragged 索引转换为块级页表，是统一验证路径的基础
- python/sglang/srt/models/qwen3_5_mtp.py（模块 模型加载；类别 source；类型 data-contract；符号 init）：修复 Quark 量化 MTP 模块 bf16 加载异常，关联 issue #23113
- python/sglang/srt/server_args.py（模块 配置；类别 source；类型 core-logic）：注册 quark 量化选项使 CLI 可接受 --quantization quark

关键符号：_build_unified_page_table_from_spec, _build_verify_unified_metadata, scatter_ragged_to_page_table_kernel, scatter_req_to_token_to_page_table_kernel, forward_extend

## 关键源码片段

### python/sglang/srt/layers/attention/aiter_backend.py

主要改动，新增 EAGLE 统一验证路径和页表转换逻辑，核心变更文件

```
    def _build_unified_page_table_from_spec(
        self,
        spec_info,
        bs: int,
        dest_buf: Optional[torch.Tensor] = None,
        swa_dest_buf: Optional[torch.Tensor] = None,
    ) -> torch.Tensor:
        """Convert ragged (token-level) kv_indices from spec_info into a 2D
        block-level page_table of shape (bs, max_num_blocks_per_seq).
        unified_attention expects max_seqlen_k = page_table.shape[1] * page_size
        to be a captured constant, so rows are sized to the backend-level
        max_num_blocks_per_seq regardless of seqused_k.
        """
        kv_indptr = spec_info.kv_indptr
        kv_flat = spec_info.kv_indices
        page_size = self.page_size
        max_blocks = (self.max_context_len + page_size - 1) // page_size

        swa_slot_mapping = None
        swa_page_table = None

        if dest_buf is not None:
            # The scatter kernel fills [0, num_blocks) and loads past that use
            # other=0, so the tail is 0-filled. Under graph replay rows > bs
```

```
            # are stale but unified_attention only walks rows [0, bs).
            page_table = dest_buf
        else:
            page_table = torch.zeros(
                bs, max_blocks, dtype=torch.int32, device=self.device
            )

        if self.use_sliding_window_kv_pool:
            swa_slot_mapping = self.token_to_kv_pool.full_to_swa_index_mapping.long()

        if swa_dest_buf is not None:
            swa_page_table = swa_dest_buf
        elif self.use_sliding_window_kv_pool:
            swa_page_table = torch.zeros_like(page_table)

        # Launch scatter kernel to populate page_table from ragged indices
        scatter_ragged_to_page_table_kernel[(bs, max_blocks)](
            kv_flat,
            kv_indptr,
            page_table,
            page_table.stride(0),
            swa_page_table,
            swa_slot_mapping,
            PAGE_SIZE=page_size,
            BLOCK_SIZE=128, # internal block size for Triton
            HAS_SWA=swa_page_table is not None,
        )
        return page_table
```

## python/sglang/srt/layers/attention/triton_ops/aiter_unified_attention.py

新增的 Triton scatter kernel，将 ragged 索引转换为块级页表，是统一验证路径的基础

```
import triton
import triton.language as tl


@triton.jit
def scatter_ragged_to_page_table_kernel(
    kv_flat_ptr,
    kv_indptr_ptr,
    dest_ptr,
    dest_stride,
    sw_page_table_ptr,
    swa_slot_mapping_ptr,
    PAGE_SIZE: tl.constexpr,
    BLOCK_SIZE: tl.constexpr,
    HAS_SWA: tl.constexpr,
):
    """Scatter ragged token-level kv_indices into a 2D block-level page table."""
```

```
    pid = tl.program_id(0) # request index
    block_id = tl.program_id(1) # block offset index

    start = tl.load(kv_indptr_ptr + pid).to(tl.int64)
    kv_len = tl.load(kv_indptr_ptr + pid + 1).to(tl.int64) - start
    num_blocks = (kv_len + PAGE_SIZE - 1) // PAGE_SIZE

    offsets = block_id * BLOCK_SIZE + tl.arange(0, BLOCK_SIZE)
    if block_id * BLOCK_SIZE >= num_blocks:
        return
    mask = offsets < num_blocks
    token_idx = offsets.to(tl.int64) * PAGE_SIZE
    vals = tl.load(kv_flat_ptr + start + token_idx, mask=mask, other=0)
    block_vals = vals // PAGE_SIZE
    tl.store(
        dest_ptr + pid.to(tl.int64) * dest_stride + offsets,
        block_vals,
        mask=mask,
    )

    if HAS_SWA:
        sw_vals = tl.load(swa_slot_mapping_ptr + vals)
        block_vals = sw_vals // PAGE_SIZE
        tl.store(
            sw_page_table_ptr + pid.to(tl.int64) * dest_stride + offsets,
            block_vals,
            mask=mask,
        )
```

## 评论区精华

1. gemini-code-assist[bot] 指出 target_verify 的 unified_attention 输出视图应使用 layer.v_head_dim 而非 qk_head_dim，与缓存形状一致。作者已修正。
2. gemini-code-assist[bot] 建议在 unified_attention 调用中尊重 sliding_window_size 而非硬编码 (-1,-1)。作者后续提交添加了滑动窗口支持。
3. mqhc2020 建议将 inline 的 Triton kernel 移到 triton_ops 模块。HaiShaw 也建议后续重构，作者已将 kernel 移至新文件。
4. kkHuang-amd 指出 init_forward_metadata 中 else 分支对 MLA 或非 unified_attention 情况下可能破坏逻辑。HaiShaw 询问后作者已处理（具体未明确但 PR 已合并）。

- 输出视图维度使用 v_head_dim vs qk_head_dim (correctness): 作者接受了建议，在后续提交中修正。
- 启用滑动窗口注意力支持 (correctness): 作者添加了滑动窗口支持（提交 2 和 3）。
- 将内联 Triton kernel 重构到 triton_ops 模块 (design): 作者在第四提交中将 kernel 移至新文件 python/sglang/srt/layers/attention/triton_ops/aiter_unified_attention.py。
- init_forward_metadata 中 else 分支对 MLA 和非 unified_attention 的潜在破坏 (correctness): 作者可能已处理（PR 合并），但具体响应不明确。

## 风险与影响

- 风险：该 PR 引入多项核心逻辑变更： 1) 环境变量 SGLANG_AITER_UNIFIED_VERIFY 控制新路径，默认开启，但未设置时可能回退到旧路径，需确保兼容性； 2) 新增的 Triton kernel 在非 AMD 平台可能未定义，通过 try/except 导入 aiter，但 scatter kernel 始终编译，可能引入符号冲突； 3) Qwen3.5 MTP 量化检测逻辑依赖 quant_config.get_name() 和 exclude_layers 属性，若 Quark 配置格式变化可能失效； 4) 缺少单元测试，仅依赖手动 GSM8K 验证，回归风险存在； 5) 预计算 qo_indptr_unified_decode 假设 q_len==1，若未来支持长序列可能会出错。
  - 影响：用户影响：AMD gfx950/MI355X 用户可体验 Qwen3.5 397B FP8/MXFP4 模型启用 EAGLE 投机解码（需设置环境变量并 --disable-radix-cache），吞吐提升约 10 倍（根据输出吞吐指标），精度保持 94%+。系统影响：改动限制在注意力后端和量化配置，未波及调度器或其他模块。团队影响：维护者需关注相关环境变量和新 kernel 维护。
  - 风险标记：缺少测试覆盖，环境变量依赖，平台兼容性风险

## 关联脉络

- PR #23113 [Bug] [rocm] qwen 3.5 mtp fp4 broken: 关联的 bug issue，本 PR 修复了 MTP 加载问题
- PR #23461 [AMD] Add kimi-k2.5 eagle3 support with unified attention.: 相关的 draft-decode 修复 PR，本 PR 保持范围外但依赖