PR #24129 完整报告

sgl-project/sglang

fix(aiter): drop FP8 KV upcast; use native FP8 path in paged_attentio···

合并时间：2026-05-08 17:47

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24129

执行摘要

- 一句话：移除 Aiter 后端 FP8 KV upcast，使用原生 FP8 路径
- 推荐动作：值得精读，尤其关注 FP8 推理优化路径的开发者。核心设计决策是消除隐式 upcast，利用内核原生 FP8 支持，这是 FP8 推理的最佳实践。需确认 aiter 内核的缩放因子处理是否与当前实现一致。

功能与动机

Aiter 后端中 FP8 KV 缓存原本会在每层注意力计算前 upcast 到更高精度（如 bf16/fp16），增加了不必要的显存带宽和计算开销，成为延迟和吞吐量的主要瓶颈。硬件和内核本身支持原生 FP8，因此应直接利用 paged_attention_ragged 中的 FP8 路径，端到端避免隐式类型转换。

实现拆解

1. 新增 _get_aiter_paged_ragged_kv_cache_dtype 方法：在 aiter_backend.py 的 AiterAttentionBackend 类中新增了一个方法，根据 self.kv_cache_dtype 返回 "fp8_e4m3"（当为 fp8_dtype 时）或 "auto"（非 FP8 时）。这是本次变更的核心函数。
2. 移除FP8KVupcast代码：在forward_decode的else分支（非unified_attention路径）中，删除了原先将 FP8 类型 KV 缓存转换为 self.input_dtype 的代码段（k_cache = k_cache.to(self.input_dtype) 等）。
3. 修改 paged_attention_ragged 调用：将传入的 kv_cache_dtype 参数从硬编码的 "auto" 替换为调用 _get_aiter_paged_ragged_kv_cache_dtype() 返回的值，从而启用原生 FP8 路径。
4. 依赖关系：此变更依赖 Aiter 侧 paged_attention_ragged 内核的 API，该内核已支持 "fp8_e4m3" 字符串并执行 in-kernel 反量化。

关键文件：

- python/sglang/srt/layers/attention/aiter_backend.py（模块 注意力层；类别 source；类型 core-logic；符号 _get_aiter_paged_ragged_kv_cache_dtype）：唯一变更文件。新增 _get_aiter_paged_ragged_kv_cache_dtype 方法并修改了 FP8 KV upcast 逻辑。

关键符号：_get_aiter_paged_ragged_kv_cache_dtype

关键源码片段

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

唯一变更文件。新增 _get_aiter_paged_ragged_kv_cache_dtype 方法并修改了 FP8 KV upcast 逻辑。

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

class AiterAttentionBackend:
    def _get_aiter_paged_ragged_kv_cache_dtype(self) -> str:
        """
        返回 paged_attention_ragged 期待的 kv_cache_dtype 字符串。

        行为变更：不再将 FP8 KV 上转型到激活数据类型的显存表示。
        分页 K/V 保留原生 FP8 存储，传递 "fp8_e4m3" 让内核在读取时反量化
        （通过 k_scale / v_scale），而非将缓存扩容为 bf16/fp16 的 "auto"。

        aiter 只接受 "auto" / "fp8" / "fp8_e4m3"（不支持 fp8_e5m2）。
        HIP 上的 configure_kv_cache_dtype 会将 CLI 中的 fp8_e5m2 和 fp8_e4m3 都映射到
        fp8_dtype；因此当 self.kv_cache_dtype 为 fp8_dtype 时返回 "fp8_e4m3"，
        否则返回 "auto"。
        """
        # 非 FP8 情况：直接返回 "auto"，由内核自动推断类型
        if self.kv_cache_dtype != fp8_dtype:
            return "auto"
        # FP8 情况：返回 "fp8_e4m3" 触发内核原生 FP8 反量化路径
        return "fp8_e4m3"

    def forward_decode(self, ...):
        ...
        else:
            # 之前：将 FP8 KV 缓存原地转为 input_dtype，产生冗余拷贝
            # if self.kv_cache_dtype == fp8_dtype:
            # k_cache = k_cache.to(self.input_dtype)
            # v_cache = v_cache.to(self.input_dtype)

            # 现在：直接使用原生 FP8 路径，传入字符串让内核处理反量化
            aiter_kv_str = self._get_aiter_paged_ragged_kv_cache_dtype()
            paged_attention_ragged(
                o.view(-1, layer.tp_q_head_num, layer.v_head_dim),
                self.workspace_buffer,
                q.view(-1, layer.tp_q_head_num, layer.qk_head_dim),
                k_cache.view(-1, 1, layer.tp_k_head_num, layer.qk_head_dim),
                v_cache.view(-1, 1, layer.tp_v_head_num, layer.v_head_dim),
                self.scale,
                self.forward_metadata.kv_indptr,
                self.forward_metadata.kv_indices,
                self.kv_last_page_len,
                1,
                self.max_num_partitions,
```

```
            None,
            aiter_kv_str, # 之前是 "auto"
            "NHD",
            self.logits_soft_cap,
            self.k_scale,
            self.v_scale,
        )
```

评论区精华

审核中，gemini-code-assist 提出了一个关键的正确性问题：当使用 aiter_kv_str == "fp8_e4m3" 时，paged_attention_ragged 需要实际的缩放因子（layer-specific k_descale / v_descale），但当前调用传入的是默认值为 1.0 的 self.k_scale / self.v_scale，可能导致 FP8 KV 缓存非单位缩放时输出错误。然而，作者 fanxingran 两次确认“all checks are OK”，表明当前实现通过测试，缩放因子问题可能在实际运行中不成立（因为缩放因子统一或内核处理方式）。审核者 HaiShaw 要求 AMD 方确认 API 变更兼容性，但未得到直接回复。最终 HaiShaw 批准了 PR，表明团队内部已达成共识。

- FP8 缩放因子传递正确性 (correctness): 作者 fanxingran 两次确认“all checks are OK”，表明实际运行无误。审核者 HaiShaw 要求 AMD 方确认 API 兼容性，但未收到回复后直接批准了 PR。这暗示当前缩放因子传递在现有场景下是安全的。

风险与影响

- 风险：主要风险在于缩放因子的正确传递：如果 Aiter 内核期望 layer-specific 的 k_descale / v_descale 而非全局 self.k_scale / self.v_scale，则可能导致 FP8 KV 缓存下注意力计算错误。此外，变更仅修改了 paged_attention_ragged 的 decode 路径，unified_attention 分支保持不变；若未来统一路径，需确认一致性。当前测试覆盖不足（未修改测试文件），回归风险依赖 CI 中的已有测试。
- 影响：影响范围仅限于 Aiter 注意力后端的 decode 阶段（paged_attention_ragged 路径）。期望收益是 FP8 模式下降低显存带宽和计算开销，提升延迟和吞吐量。对于非 FP8 模式，行为不变（"auto" 路径）。用户无需修改配置或代码。
- 风险标记：核心路径变更，缺少测试覆盖，缩放因子潜在问题

关联脉络

- PR #24686 Remove unnecessary bf16 assert in rotate_activation: 同为 FP8 相关优化，移除不必要的类型限制，体现团队在 FP8 支持上的持续改进。
- PR #24271 [KDA] Optimize prefill kernels with diagonal and recompute fuse: 同为注意力核的性能优化，体现了对注意力路径持续的 Kernel 级优化趋势。