PR #23927 完整报告

sgl-project/sglang

[AMD] Replace fp8 mla with fp8 mha kernel for diffusion model aiter backend

合并时间：2026-06-10 05:46

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23927

执行摘要

- 一句话：用 FP8 MHA 内核替换 MLA 内核，提速 13.8%
- 推荐动作：值得精读，特别是对 AMD GPU 上的扩散模型优化感兴趣的人。设计决策亮点：使用清晰的门控函数隔离平台特定路径，移除 @torch.compiler.disable 以恢复 torch.compile 兼容性。但需关注测试覆盖缺口，建议在后续 PR 中添加形状参数化测试以覆盖 FP8 路径。

功能与动机

原始的 FP8 prefill 路径在 AITer 后端中临时使用了 MLA prefill ASM 内核来在 MI350 系列上运行 FP8 注意力。切换到正确的 FP8 FMHA ASM 内核（fmha_fwd_hd128_fp8_gfx950）移除了额外的填充、头部可除尽约束和整个元数据构建脚手架。FMHA 路径原生支持 MHA，q/k/v head_dim == 128，适用于 Wan 2.2 和其他扩散模型的自注意力和交叉注意力形状。

实现拆解

1. 移除 MLA 预填充路径：删除 _can_use_mla_prefill、_build_mla_prefill_metadata 和 _mla_prefill_ps_attention 三个函数，以及相关的常量（_MLA_PREFILL_QK_HEAD_DIM、_MLA_PREFILL_V_HEAD_DIM、_MLA_PREFILL_HEAD_TILE），约 250 行代码。
2. 添加 FMHA FP8 预填充路径：
   - 定义 _FMHA_FP8_HEAD_DIM = 128 常量。
   - 实现 _can_use_fmha_fp8_prefill 门控函数，检查 arch、MHA 模式以及 q/k/v head_dim 均为 128。
   - 实现 _fmha_fp8_prefill_attention 函数，直接调用 aiter.flash_attn_fp8_pertensor_func，内部辅助 _ensure_fp8_descale 转换 scale 为 per-tensor descale 形状。
3. 更新 AITerImpl.forward 入口：移除 @torch.compiler.disable 装饰器；当 _use_fp8_attn 且形状满足条件时，调用新函数代替旧的 MLA 路径；不满足时回退到原始的 flash_attn_func。
4. 清理依赖：移除不再需要的 typing.Optional 导入和 F.pad 调用。
5. 测试与配套：未新增测试文件；现有 AMD CI 验证默认路径无回归，但新 FP8 路径未被 CI 覆盖，需要后续补充。

关键文件：

- python/sglang/multimodal_gen/runtime/layers/attention/backends/aiter.py（模块 扩散模型；类别 source；类型 core-logic；符号 _can_use_mla_prefill, _build_mla_prefill_metadata, _mla_prefill_ps_attention, _can_use_fmha_fp8_prefill）：核心变更文件，替换整个 FP8 预填充路径，移除 250+ 行 MLA 适配代码，新增约 70 行 FMHA 实现。

关键符号：_can_use_fmha_fp8_prefill, _fmha_fp8_prefill_attention, _ensure_fp8_descale, AITerImpl.forward

关键源码片段

python/sglang/multimodal_gen/runtime/layers/attention/backends/aiter.py

核心变更文件，替换整个 FP8 预填充路径，移除 250+ 行 MLA 适配代码，新增约 70 行 FMHA 实现。

```
# fmha_fwd_hd128_fp8_gfx950 ASM kernel.
# Support full MHA with q/k/v head_dim == 128 -- e.g., Wan 2.2 self- and cross-attention.
_FMHA_FP8_HEAD_DIM = 128


def _can_use_fmha_fp8_prefill(
    q_head_dim: int,
    k_head_dim: int,
    v_head_dim: int,
    num_heads: int,
    num_kv_heads: int,
) -> bool:
    """True if MHA q/k/v head_dim==128 on a gfx950-class arch."""
    if not _use_aiter_gfx95:
        return False
    if num_kv_heads != num_heads:
        return False
    return q_head_dim == k_head_dim == v_head_dim == _FMHA_FP8_HEAD_DIM


def _fmha_fp8_prefill_attention(
    q: torch.Tensor,
    k: torch.Tensor,
    v: torch.Tensor,
    softmax_scale: float,
    is_causal: bool,
    q_scale: torch.Tensor,
    k_scale: torch.Tensor,
    v_scale: torch.Tensor,
) -> torch.Tensor:
    """
    FP8 FMHA prefill via aiter.flash_attn_fp8_pertensor_func.
    Expects q, k, v as (batch, seqlen, nheads, 128) FP8, contiguous.
    """
```

```
    def _ensure_fp8_descale(scale: torch.Tensor) -> torch.Tensor:
        """Per-tensor descale as shape (1,) float32 for pertensor func."""
        return scale.to(dtype=torch.float32).reshape(1).contiguous()

    q = q.contiguous()
    k = k.contiguous()
    v = v.contiguous()
    q_descale = _ensure_fp8_descale(q_scale)
    k_descale = _ensure_fp8_descale(k_scale)
    v_descale = _ensure_fp8_descale(v_scale)

    return aiter.flash_attn_fp8_pertensor_func(
        q, k, v,
        q_descale, k_descale, v_descale,
        causal=is_causal,
        softmax_scale=softmax_scale,
        window_size=(-1, -1, 0),
    )


class AITerImpl(AttentionImpl):
    # ...
    # 在 forward 中，当 _use_fp8_attn 且形状满足条件时调用新函数，
    # 否则回退到 bf16 flash_attn_func 或旧路径。
    def forward(self, query, key, value, attn_metadata=None):
        if _use_fp8_attn:
            if query.dtype != _fp8_dtype:
                # cast to fp8
                ...
            if _can_use_fmha_fp8_prefill(
                q_head_dim=query.shape[-1],
                k_head_dim=key.shape[-1],
                v_head_dim=value.shape[-1],
                num_heads=query.shape[-2],
                num_kv_heads=key.shape[-2],
            ):
                return _fmha_fp8_prefill_attention(
                    q_fp8, k_fp8, v_fp8,
                    self.softmax_scale, self.causal,
                    q_scale, k_scale, v_scale,
                )
            else:
                # fallback to original flash_attn_func
                ...
        else:
            # bf16 path
            ...
```

评论区精华

- 依赖外部 PR #21742 以获得正确输出视频 (question): PR 已合并，但依赖关系仍需解决
- 新 FP8 MHA 路径未被 PR CI 测试覆盖 (testing): PR 仍然合并，但后续可能需要添加测试覆盖

风险与影响

- 风险：
  - 测试覆盖不足：新 FP8 MHA 路径未在标准 CI 中运行，仅依赖作者手动测试，回归风险高。
  - 平台绑定：门控函数 _use_aiter_gfx95 使其仅适用于 MI350 系列，在非 gfx950 架构上静默回退到 BF16 路径，不会崩溃但可能性能下降。
  - MHA 假设：_can_use_fmha_fp8_prefill 要求 num_kv_heads == num_heads，GQA 模型无法使用此 FP8 加速，且无 fallback 告警。
  - 外部依赖：依赖 aiter 和 flash_attn_fp8_pertensor_func，该函数的接口或行为变化可能影响功能。
- 影响：
  - 用户：AMD MI350 用户启用 SGLANG_DIFFUSION_AITER_FP8_ATTN=1 后，扩散模型（如 Wan 2.2）的 FP8 attention 获得 13.8% 端到端加速，且支持 Ulysses SP 4 路 40 头模型。
  - 系统：减少内存开销（消除 padding）和元数据构建，降低显存占用。
  - 团队：消除临时 MLA 变通方案的技术债务，简化注意力后端代码结构。
  - 风险标记：缺少测试覆盖，平台特定路径（仅 MI350），依赖外部库（aiter），假设全 MHA（无 GQA）

关联脉络

- PR #21742 Unknown (referenced in comment): 被 jcaraban 指出为 FP8 attention 的依赖，可能影响输出质量