PR #45720 完整报告

vllm-project/vllm

[Bugfix][ROCm] Fix MiniMax-M3 FP8 KV cache dtype

合并时间：2026-06-17 11:14

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45720

执行摘要

- 一句话：修复 ROCm 上 MiniMax-M3 FP8 KV 缓存 dtype 错误
- 推荐动作：本 PR 值得关注，尤其是跨平台 dtype 决策的提炼方式：引入 current_platform.fp8_dtype() 和 is_fp8_fnuz() 避免了平台硬编码，是类似问题的标准处理模式。同时，新增的测试覆盖了平台参数化，可以作为 dtype 选择场景的参考模板。建议开发者在涉及平台类型差异时采用类似策略。

功能与动机

关联 Issue #45562 报告在 ROCm MI300X 上，使用 --kv-cache-dtype fp8 时 MiniMax-M3 模型生成结果严重错误，GSM8K strict accuracy 从 0.9666 降至 0.0099。根因是 MiniMax-M3 的 sparse attention 后端硬编码 torch.float8_e4m3fn 来重新解释字节型 FP8 KV 缓存，但 ROCm 平台使用 torch.float8_e4m3fnuz，两种编码不同导致 KV 值被错误读取。

实现拆解

1. 在 sparse_attention.py 的 MiniMaxM3SparseImpl.__init__ 中，将硬编码的 dtype 选择改为：若 kv_cache_dtype 包含 'e5m2'，则根据 current_platform.is_fp8_fnuz() 选择 torch.float8_e5m2fnuz 或 torch.float8_e5m2；否则使用 current_platform.fp8_dtype()（对 ROCm 返回 float8_e4m3fnuz，对 CUDA 返回 float8_e4m3fn）。
2. 在 sparse_attn.py 中定义了 _FP8_DTYPES 元组，包含全部四种 FP8 类型（e4m3fn、e4m3fnuz、e5m2、e5m2fnuz），并将内核中判断是否使用 FP8 路径的条件从 (torch.float8_e4m3fn, torch.float8_e5m2) 改为 in _FP8_DTYPES。
3. 新增两个测试函数：test_sparse_impl_uses_platform_fp8_dtype 参数化验证不同 kv_cache_dtype 字符串下构造的 impl 对象能正确返回平台期望的 dtype；test_sparse_kernels_recognize_fp8_dtypes 验证所有四种 FP8 类型都被 _FP8_DTYPES 识别。

关键文件：

- vllm/models/minimax_m3/common/sparse_attention.py（模块 注意力；类别 source；类型 core-logic；符号 MiniMaxM3SparseImpl.init）：核心修复，在 MiniMaxM3SparseImpl.__init__中使用 current_platform.fp8_dtype() 替换硬编码，并处理 E5M2FNUZ 分支，直接影响 KV 缓存的 dtype 选择。

- vllm/models/minimax_m3/common/ops/sparse_attn.py（模块 算子；类别 source；类型 infrastructure；符号 minimax_m3_sparse_attn, minimax_m3_sparse_attn_decode）：定义了 _FP8_DTYPES 元组供内核使用，并修改了 prefill 和 decode 两个函数的 dtype 检查条件，确保所有 FP8 变体都能触发 FP8 转换路径。
- tests/kernels/attention/test_minimax_m3.py（模块 测试；类别 test；类型 test-coverage；符号 test_sparse_impl_uses_platform_fp8_dtype, test_sparse_kernels_recognize_fp8_dtypes）：新增两个参数化测试，验证稀疏 attention 实现的 dtype 选择正确性，以及 _FP8_DTYPES 能否识别所有 FP8 类型。防止回归。

关键符号：MiniMaxM3SparseImpl.init, minimax_m3_sparse_attn, minimax_m3_sparse_attn_decode, test_sparse_impl_uses_platform_fp8_dtype, test_sparse_kernels_recognize_fp8_dtypes

关键源码片段

vllm/models/minimax_m3/common/sparse_attention.py

核心修复，在 MiniMaxM3SparseImpl.__init__中使用 current_platform.fp8_dtype() 替换硬编码，并处理 E5M2FNUZ 分支，直接影响 KV 缓存的 dtype 选择。

```
class MiniMaxM3SparseImpl(AttentionImplBase):
    def __init__(self, ..., kv_cache_dtype: str = "auto", ...):
        self.kv_cache_dtype = kv_cache_dtype
        self.use_fp8_kv = is_quantized_kv_cache(kv_cache_dtype)
        # 原来是硬编码：
        # torch.float8_e5m2 if "e5m2" in kv_cache_dtype else torch.float8_e4m3fn
        # 现在改为动态平台感知：
        if "e5m2" in kv_cache_dtype:
            self.kv_cache_fp8_dtype = (
                torch.float8_e5m2fnuz
                if current_platform.is_fp8_fnuz()
                else torch.float8_e5m2
            )
        else:
            # 对 ROCm 返回 float8_e4m3fnuz，对 CUDA 返回 float8_e4m3fn
            self.kv_cache_fp8_dtype = current_platform.fp8_dtype()
        self.topk_blocks = topk_blocks
        self.block_size = sparse_block_size
```

vllm/models/minimax_m3/common/ops/sparse_attn.py

定义了 _FP8_DTYPES 元组供内核使用，并修改了 prefill 和 decode 两个函数的 dtype 检查条件，确保所有 FP8 变体都能触发 FP8 转换路径。

```
# 定义全部 FP8 类型集合，供内核检查使用
_FP8_DTYPES = (
    torch.float8_e4m3fn,
    torch.float8_e4m3fnuz,
    torch.float8_e5m2,
```

```
    torch.float8_e5m2fnuz,
)

def minimax_m3_sparse_attn(..., kv_cache, ...):
    # 之前只检查 (torch.float8_e4m3fn, torch.float8_e5m2)
    # 现在改为检查全集
    use_fp8 = kv_cache.dtype in _FP8_DTYPES
    ...

def minimax_m3_sparse_attn_decode(..., kv_cache, ...):
    use_fp8 = kv_cache.dtype in _FP8_DTYPES
    ...
```

评论区精华

PR 没有公开的 review 评论线程，但 reviewer @hongxiayang 参与了 co-authored commit（3e9faf5），在 E5M2 分支中增加了对 float8_e5m2fnuz 的判断。该修改使得 ROCm 的 E5M2 也能正确选择 FNUZ 变体，完善了原实现的平台覆盖。

- 暂无高价值评论线程

风险与影响

- 风险：本次修改仅影响 MiniMax-M3 模型，且只在 ROCm 平台上原有行为有差异。核心风险在于：如果未来新增其他 FP8 变体（如 e4m3fnuz 的另一种形式），需要同步更新 _FP8_DTYPES 和 MiniMaxM3SparseImpl。目前测试覆盖了所有已知的 FP8 dtype，降低了回归风险。性能上无影响，仅改变了 dtype 选择逻辑。
- 影响：修复了 ROCm 平台（主要是 MI300X、MI325X）使用 MiniMax-M3 模型并开启 FP8 KV 缓存时的极端精度问题（GSM8K 从 0.01 恢复至 0.956）。用户无需修改配置即可获得正常精度。对 CUDA 平台无影响，因为 current_platform.fp8_dtype() 返回 float8_e4m3fn 与之前硬编码一致。系统其他组件不受影响。
- 风险标记：跨平台 dtype 兼容性，核心路径变更，仅影响 ROCm

关联脉络

- PR #45563 [Bugfix][ROCm] Fix MiniMax-M3 FP8 KV cache dtype（替代版本）：本 PR 是 #45563 的替代版本，当 #45563 的 base 分支被合并移除后重新基于 main 提交。修复逻辑相同，且包含了 #45563 的 review 反馈（E5M2FNUZ 处理）。