PR #40871 完整报告

vllm-project/vllm

[New Model][ROCm] Add AMD support for DeepSeek V4

合并时间：2026-05-05 23:55

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/40871

执行摘要

- 一句话：为 DeepSeek V4 模型添加 AMD ROCm 支持
- 推荐动作：值得精读，尤其是 ROCm 平台适配的分支策略（早期返回 vs 内部分支）和 MoE 后端选择逻辑（AITER vs Triton-unfused）。建议关注未来对 AITER 代码重复、Mooncake 断言精度等问题的后续修复。

功能与动机

PR 描述明确指出："This PR adds support of DeepSeek V4 for AMD." 旨在扩展 vLLM 对 AMD ROCm 平台的支持，使 DeepSeek V4 模型能够在 AMD GPU 上部署。

实现拆解

1. 注意力层适配（deepseek_v4_attention.py）：导入 ROCm 专用操作（rocm_inv_rope_einsum, rocm_sparse_attn_prefill, rocm_forward_decode_fallback），在 forward 中提前返回 ROCm 路径，避免后续 FP8 量化；在 attn_gemm_parallel_execute 中处理 aux_streams 为 None 的情况。
2. MoE 后端优化（oracle/mxfp4.py）：添加 AITER 作为 AITER_MXFP4_BF16 的别名；在 _get_priority_backends 中为 ROCm 平台返回 [AITER_MXFP4_BF16]；为 DeepSeek V4 路由模式设置优先列表 [TRITON_UNFUSED, AITER_MXFP4_BF16] 以保证精度；添加 convert_weight_to_mxfp4_moe_kernel_format 的 AITER 分支，执行权重混洗和格式转换。
3. 混合缓存层参考实现（mhc.py）：在 mhc_pre 和 mhc_post 中增加 if current_platform.is_rocm(): 分支，使用纯 PyTorch 算子替换 tilelang 内核；新增 _hc_head_fused_reference 函数作为 hc_head_fuse_tilelang 的 PyTorch 回退。
4. 稀疏注意力索引器（sparse_attn_indexer.py）：在 forward_hip 中移除对 skip_k_cache_insert 的阻止断言，增加不依赖 AITER 的原生路径（rocm_aiter_sparse_attn_indexer_native），并调整异常处理。
5. 量化线性层兼容性（aiter.py, fp8_utils.py）：在 apply_block_scaled_mm 中检测 E8M0 格式并上转换为 FP32；在 w8a8_triton_block_scaled_mm 中增加 ROCm 的 E8M0 转 FP32 处理。
6. 多流并行禁用（deepseek_v4.py, deepseek_v4_attention.py）：在 ROCm 上将 aux_stream_list 设置为 None，避免挂起问题。

7. 测试与文档：在 tests/ 中添加相应测试（如 test_mhc.py），更新 supported_models.md 等文档。

关键文件：

- vllm/model_executor/layers/deepseek_v4_attention.py（模块 注意力层；类别 source；类型 core-logic）：注意力层核心修改，添加 ROCm 导入和分支逻辑，控制注意力计算路径
- vllm/model_executor/layers/mhc.py（模块 混合缓存；类别 source；类型 core-logic；符号 _hc_head_fused_reference）：混合缓存层添加 ROCm 参考实现和 hc_head_fuse 纯 PyTorch 回退
- vllm/model_executor/layers/fused_moe/oracle/mxfp4.py（模块 MoE 调度；类别 source；类型 core-logic）：MoE 后端选择策略适配 ROCm，添加 AITER 权重转换和精度优化
- vllm/v1/attention/ops/rocm_aiter_mla_sparse.py（模块 稀疏注意力；类别 infra；类型 infrastructure；符号 _topk_indices_torch, rocm_aiter_sparse_attn_indexer, rocm_aiter_sparse_attn_indexer_native, _decode_e8m0_scales）：新增大量 ROCm 专用稀疏注意力操作，包括 TOPK 索引、decode 回退、inv_rope_einsum 等
- vllm/model_executor/layers/sparse_attn_indexer.py（模块 索引器；类别 source；类型 core-logic）：调整 forward_hip 方法，支持 skip_k_cache_insert 和使用原生索引器

关键符号：_hc_head_fused_reference, rocm_inv_rope_einsum, rocm_sparse_attn_prefill, rocm_forward_decode_fallback, rocm_aiter_sparse_attn_indexer_native, _topk_indices_torch

关键源码片段

vllm/model_executor/layers/mhc.py

混合缓存层添加 ROCm 参考实现和 hc_head_fuse 纯 PyTorch 回退

```
def mhc_pre(
    residual: torch.Tensor,
    fn: torch.Tensor,
    hc_scale: torch.Tensor,
    hc_base: torch.Tensor,
    rms_eps: float = 1e-6,
    hc_eps: float = 1e-6,
    hc_post_mult_value: float = 1.0,
    sinkhorn_repeat: int = 1,
) -> tuple[torch.Tensor, torch.Tensor, torch.Tensor]:
    # ... 形状验证省略 ...
    residual_flat = residual.view(-1, hc_mult, hidden_size)
    num_tokens = residual_flat.shape[0]

    # ROCm 路径：使用纯 PyTorch 实现，代替 tilelang 内核
    if current_platform.is_rocm():
        # 将输入展开为 2D 并转为 FP32
        x = residual_flat.view(num_tokens, hc_mult * hidden_size).to(torch.float32)
        # 计算线性混合并应用 RMS 归一化
```

```
        mixes = torch.matmul(x, fn.t())
        sqrsum = x.square().sum(dim=-1, keepdim=True)
        mixes = mixes * torch.rsqrt(sqrsum / (hc_mult * hidden_size) + rms_eps)

        # 分离出 pre、post、comb 分量
        pre_logits = mixes[:, :hc_mult] * hc_scale[0] + hc_base[:hc_mult]
        pre_mix = torch.sigmoid(pre_logits) + hc_eps

        post_logits = (mixes[:, hc_mult:2 * hc_mult] * hc_scale[1]
                       + hc_base[hc_mult:2 * hc_mult])
        post_mix = torch.sigmoid(post_logits) * hc_post_mult_value

        comb_logits = (mixes[:, 2 * hc_mult:].view(num_tokens, hc_mult, hc_mult)
                       * hc_scale[2] + hc_base[2 * hc_mult:].view(1, hc_mult, hc_mult))
        comb_mix = torch.softmax(comb_logits, dim=-1) + hc_eps
        comb_mix = comb_mix / (comb_mix.sum(dim=-2, keepdim=True) + hc_eps)
        for _ in range(sinkhorn_repeat - 1):
            comb_mix = comb_mix / (comb_mix.sum(dim=-1, keepdim=True) + hc_eps)
            comb_mix = comb_mix / (comb_mix.sum(dim=-2, keepdim=True) + hc_eps)

        # 加权求和得到 layer_input
        layer_input = torch.sum(
            pre_mix.unsqueeze(-1) * residual_flat.to(torch.float32), dim=1
        ).to(torch.bfloat16)

        return (
            post_mix.view(*outer_shape, hc_mult, 1),
            comb_mix.view(*outer_shape, hc_mult, hc_mult),
            layer_input.view(*outer_shape, hidden_size),
        )

    # CUDA tilelang 路径（保持不变）
    # ...
```

评论区精华

- 注意力层分支策略：zyongye 建议减少 deepseek_v4_attention.py 中的分支数量，将 ROCm 执行路径提前分出。whx-sjtu 回应已重构，仅在关键操作分支，Compressor 模块不更改。
- MoE 后端精度：tjtanaa 和 whx-sjtu 发现 Triton fused 路径在 ROCm 上有精度问题，故优先使用 Triton-unfused 和 AITER 后端。最终采用优先列表 [TRITON_UNFUSED, AITER_MXFP4_BF16] 确保精度。
- 多流并行禁用：ZJY0516 询问 aux_stream_list 为何设为 None；whx-sjtu 解释类型定义支持 None，且在 ROCm 上存在挂起问题，后续修复。
- 全局导入保护：tjtanaa 指出 from vllm.platforms.rocm import _ON_GFX942 无条件导入会在非 ROCm 平台出错。whx-sjtu 修复为函数内局部导入。

- AITER 代码重复：gemini-code-assist 指出 mxfp4.py 中 AITER 后端转换块在三处重复，应提取共用函数。未在本次 PR 中解决。
- Mooncake 断言问题：gemini-code-assist 指出 assert self.block_size > kernel_block_size 可能阻止用户设置较小 block_size 时的自动增大，需修复。
- Monkey-patching：gemini-code-assist 指出 gpt_oss_triton_kernels_moe.py 中 monkey-patching SparseMatrix.__post_init__.__globals__ 脆弱，建议更稳健方案。未在本次 PR 中改变。
  - 注意力层 ROCm 分支策略 (design): whx-sjtu 回应已重构注意力部分，当前仅在关键操作分支，Compressor 模块不更改。
  - ROCm 上 MoE 后端精度问题 (correctness): 采用优先列表 [TRITON_UNFUSED, AITER_MXFP4_BF16] 确保精度。
  - ROCm 上 aux_stream_list 设置为 None (performance): 保持 None，后续修复多流问题后再启用。
  - 全局导入 _ON_GFX942 需保护 (correctness): whx-sjtu 修复为在函数内局部导入或加条件。

风险与影响

- 风险：
  - 维护成本风险：deepseek_v4_attention.py 中新增 ROCm 分支（BF16 参考路径），与 CUDA 的 FP8 路径形成两套逻辑，增加长期维护负担。
  - 性能风险：ROCm 上使用的 BF16 参考路径可能不如 CUDA 的 FP8 路径高效，影响推理吞吐。
  - 代码重复风险：mxfp4.py 中 AITER 权重转换代码重复三份，可能导致后续修改不一致。
  - 跨平台兼容性风险：全局导入 _ON_GFX942 虽已修复，但类似的平台特定代码可能遗漏条件保护，导致非 ROCm 平台崩溃。
  - 量化精度风险：ROCm 上 E8M0 格式需转换为 FP32，可能因精度损失影响模型输出质量。
  - 影响：用户影响：AMD 平台用户现在可以运行 DeepSeek V4 模型（Flash 和 Pro 版本），无需 CUDA。系统影响：代码库新增约 940 行 ROCm 平台专用代码，模块间引入平台分支，增加架构复杂度。维护影响：团队需要维护两套注意力执行路径，后续量化内核的更新需同步考虑 ROCm 兼容性。性能影响：当前 ROCm 路径精度已验证（GSM8K 准确率约 95%），但性能可能低于 CUDA 路径，有待后续优化。
- 风险标记：跨平台分支增加维护负担，ROCm 参考路径性能风险，AITER 后端代码重复，全局导入条件缺失，量化精度依赖

关联脉络

- 暂无明显关联 PR