PR #31531 完整报告

sgl-project/sglang

[Refactor] Separate ROCm-specific DeepSeek MHA and MLA forward paths

合并时间：2026-08-09 04:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31531

执行摘要

- 一句话：DeepSeek 注意力前向拆分出独立 ROCm 路径
- 推荐动作：值得精读。该 PR 展示了如何通过枚举 + 调度函数优雅分离平台特例，是处理多后端代码库中平台分支蔓延的典型案例。重点关注 resolve_rocm_forward_method 的映射表设计、ROCm 专用 mixin 如何复用共享 core 逻辑，以及共享函数去私有化的边界。

功能与动机

PR body 明确指出目标："Separate ROCm-specific DeepSeek attention logic from the shared MHA and MLA forward implementations to improve code organization and maintainability." 此外，reviewer Fridge003 在评论中建议不要将 ROCm 逻辑包装为独立函数，而是新增两个专属 attention forward method（MHA_ROCM、MLA_ROCM），以便从 CUDA 变体中彻底清理 AMD 分支。

实现拆解

1. 枚举与调度层扩展：在 forward_methods.py 中新增 MHA_ROCM、MHA_ONE_SHOT_ROCM、MLA_ROCM 三个 AttnForwardMethod 枚举值；在 attention_backend_handler.py 中新增 _ROCM_FORWARD_METHODS 映射表和 resolve_rocm_forward_method()，HIP 平台将通用枚举映射到 ROCm 专用枚举，非 HIP 平台原样返回。
2. ROCm 专用前向实现：新增 forward_mla_rocm.py 与 forward_mha_rocm.py，分别提供 DeepseekMLARocmForwardMixin 与 DeepseekMHARocmForwardMixin，把 AITER/gfx95 相关的融合量化、absorb BMM、fused RoPE+KV cache 写入等逻辑全部收拢到这两个文件，并导出 rocm_absorb_q_bmm()、rocm_absorb_v_bmm()、_fused_rope_cat_and_cache() 等模块级辅助函数。
3. 共享路径净化：forward_mla.py 与 forward_mha.py 删除所有 _use_aiter/_use_aiter_gfx95 分支，_set_mla_kv_buffer/_get_mla_kv_buffer 只保留 CUDA 分支；同时将 _is_mla_dcp_lse_base_on_e、_should_defer_dsa_cp_kv_gather 等去私有化，供 ROCm 模块复用。
4. 模型装配与分发：deepseek_v2.py 的 DeepseekV2AttentionMLA 混入新的 ROCm mixin，dispatch_attn_forward_method() 在返回前调用 resolve_rocm_forward_method()，forward_prepare()/forward_core() 增加对 MHA_ROCM、MHA_ONE_SHOT_ROCM、MLA_ROCM 的分发分支；同时将 forward_mla_fused_rope_rocm.py 中的 mixin 重命名

为 DeepseekMLAFusedRopeRocmForwardMixin 以避免命名冲突。

5. 测试配套：test_deepseek_mla_dispatch.py 新增 TestResolveRocmForwardMethod，验证 HIP 下通用方法被路由到 ROCm 方法、平台专用方法保持不动、非 HIP 平台为恒等映射。

关键文件：

- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla_rocm.py（模块 MLA 前向；类别 source；类型 core-logic；符号 fused_qk_rmsnorm_bf16, rocm_absorb_q_bmm, rocm_absorb_v_bmm, _fused_rope_cat_and_cache）：新增的 ROCm 专用 MLA 前向实现，封装了所有 AITER/gfx95 内核选择，是本次重构的核心产物。
- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mha_rocm.py（模块 MHA 前向；类别 source；类型 core-logic；符号 DeepseekMHARocmForwardMixin, forward_normal_rocm_prepare, forward_normal_one_shot_rocm_prepare, _concat_and_cast_mha_k_rocm）：新增的 ROCm 专用 MHA 前向 prepare，封装 FP8/MXFP4 融合量化与 KV cache 读写差异。
- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla.py（模块 MLA 前向；类别 source；类型 refactor；符号 is_dcp_mla_decode_phase, is_mla_dcp_lse_base_on_e, fused_qk_rmsnorm_bf16, should_defer_dsa_cp_kv_gather）：共享 MLA 前向删除所有 AMD/AITER 分支，恢复为纯 CUDA/ 通用路径，是重构的主要削减对象。
- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mha.py（模块 MHA 前向；类别 source；类型 refactor；符号 resolve_attn_backend, forward_dsa_indexer_for_mha）：共享 MHA 前向删除所有 ROCm 分支，纯 CUDA 路径，同时将辅助函数去私有化。
- python/sglang/srt/models/deepseek_common/attention_backend_handler.py（模块 平台调度；类别 source；类型 core-logic；符号 resolve_rocm_forward_method）：新增 resolve_rocm_forward_method 调度函数，HIP 平台将通用枚举映射到 ROCm 专用枚举，实现平台分派。
- python/sglang/srt/models/deepseek_v2.py（模块 模型装配；类别 source；类型 entrypoint）：接入新 mixin 和新枚举，在 forward_prepare/forward_core 中分发到 ROCm 专用方法。
- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_methods.py（模块 枚举定义；类别 source；类型 data-contract）：新增 AttnForwardMethod 枚举值，定义平台专用前向类型。
- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla_fused_rope_rocm.py（模块 MLA 前向；类别 source；类型 refactor；符号 DeepseekMLARocmForwardMixin, DeepseekMLAFusedRopeRocmForwardMixin）：重命名 mixin 避免与新 MLA ROCm mixin 命名冲突。
- test/registered/unit/models/test_deepseek_mla_dispatch.py（模块 调度测试；类别 test；类型 test-coverage；符号 TestResolveRocmForwardMethod, test_hip_routes_shared_methods_to_rocm, test_hip_leaves_platform_specific_methods_alone, test_non_hip_is_identity）：新增 TestResolveRocmForwardMethod 测试，验证

HIP 平台路由和身份映射。

关键符号：resolve_rocm_forward_method, forward_absorb_rocm_prepare, forward_absorb_rocm_core, forward_normal_rocm_prepare, forward_normal_one_shot_rocm_prepare, rocm_absorb_q_bmm, rocm_absorb_v_bmm, _fused_rope_cat_and_cache, _concat_and_cast_mha_k_rocm, _set_mla_kv_buffer_rocm, _get_mla_kv_buffer_rocm

关键源码片段

python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla_rocm.py

新增的 ROCm 专用 MLA 前向实现，封装了所有 AITER/gfx95 内核选择，是本次重构的核心产物。

```
def rocm_absorb_q_bmm(
    attn: DeepseekV2AttentionMLA,
    q_nope: torch.Tensor,
    *,
    is_capture_mode: bool,
) -> torch.Tensor:
    # 在 HIP/AITER 上执行 q_nope @ w_kc 的 absorb BMM（pre-transpose 布局）
    # TODO(haishaw): add bmm_fp8 to ROCm
    if _use_aiter_gfx95 and attn.w_kc.dtype == torch.uint8:
        # gfx95 上 w_kc 为 MXFP4 (uint8) 时，走 AITER 的 fused pre-quant 路径
        x = q_nope.transpose(0, 1)
        q_nope_out = torch.empty(
            x.shape[0],
            x.shape[1],
            attn.w_kc.shape[2],
            device=x.device,
            dtype=torch.bfloat16,
        )
        batched_gemm_afp4wfp4_pre_quant(
            x,
            attn.w_kc.transpose(-2, -1),
            attn.w_scale_k.transpose(-2, -1),
            torch.bfloat16,
            q_nope_out,
        )
    else:
        # fp8 路径：gfx950 上总是走 Triton kernel，gfx942 上仅在 CUDA graph 捕获时走
        if (_use_aiter_gfx95 and attn.w_kc.dtype == torch.float8_e4m3fn) or (
            is_capture_mode and attn.w_kc.dtype == torch.float8_e4m3fnuz
        ):
            q_nope_out = batched_gemm_a8w8_a_per_token_group_prequant_w_per_batched_
            tensor_quant(
                X=q_nope,
```

```
                WQ=attn.w_kc.transpose(-1, -2),
                w_scale=attn.w_scale,
                group_size=128,
                YQ=None, # allocate (B, M, N)
                transpose_bm=False, # (B, M, N)
                transpose_bm_in=True, # (M, B, K)
                dtype=torch.bfloat16,
            )
        else:
            # 兜底：直接 bmm，先转 BF16 并应用 scale
            q_nope_out = torch.bmm(
                q_nope.to(torch.bfloat16).transpose(0, 1),
                attn.w_kc.to(torch.bfloat16) * attn.w_scale,
            )
    return q_nope_out
```

python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mha_rocm.py

新增的 ROCm 专用 MHA 前向 prepare，封装 FP8/MXFP4 融合量化与 KV cache 读写差异。

```
def _concat_and_cast_mha_k_rocm(
    self: DeepseekV2AttentionMLA,
    k_nope: torch.Tensor,
    k_pe: torch.Tensor,
):
    # 拼接 k_nope 与 k_pe 成完整的 k，aiter 后端走 Triton 快速路径，否则直接切片赋值
    k_shape = (k_nope.shape[0], self.num_local_heads, self.qk_head_dim)
    k = k_nope.new_empty(*k_shape)
    if self.current_attention_backend == "aiter":
        concat_and_cast_mha_k_triton(k, k_nope, k_pe)
    else:
        k[..., : self.qk_nope_head_dim] = k_nope
        k[..., self.qk_nope_head_dim :] = k_pe
    return k


def _set_mla_kv_buffer_rocm(
    self: DeepseekV2AttentionMLA,
    latent_cache: torch.Tensor,
    kv_a: torch.Tensor,
    k_pe: torch.Tensor,
    forward_batch: ForwardBatch,
):
    # 按平台写入 MLA KV buffer：gfx95 用专用 set_mla_kv_buffer，否则回退 latent_cache 覆盖
    if _use_aiter_gfx95:
        get_token_to_kv_pool().set_mla_kv_buffer(
            self.attn_mha, forward_batch.out_cache_loc, kv_a.unsqueeze(1), k_pe
        )
    else:
```

```
        latent_cache[:, :, : self.kv_lora_rank] = kv_a.unsqueeze(1)
        latent_cache[:, :, self.kv_lora_rank :] = k_pe.clone()
        get_token_to_kv_pool().set_kv_buffer(
            self.attn_mha, forward_batch.out_cache_loc, latent_cache, None
        )
```

python/sglang/srt/models/deepseek_common/attention_backend_handler.py

新增 resolve_rocm_forward_method 调度函数，HIP 平台将通用枚举映射到 ROCm 专用枚举，实现平台分派。

```
# ROCm 运行专用的 MHA/MLA 实现（forward_mha_rocm.py / forward_mla_rocm.py），
# 因此共享 CUDA 路径不再携带 AMD 分支。后端 handler 仍然返回通用方法，
# 平台替换在这里统一完成。
# MHA_CHUNKED_KV 没有 ROCm 入口，因为其累积步骤依赖仅 CUDA 的 merge_state_v2 内核。
_ROCM_FORWARD_METHODS = {
    AttnForwardMethod.MHA: AttnForwardMethod.MHA_ROCM,
    AttnForwardMethod.MHA_ONE_SHOT: AttnForwardMethod.MHA_ONE_SHOT_ROCM,
    AttnForwardMethod.MLA: AttnForwardMethod.MLA_ROCM,
}


def resolve_rocm_forward_method(method: AttnForwardMethod) -> AttnForwardMethod:
    # 非 HIP 平台不做任何替换，保持原方法
    if not _is_hip:
        return method
    return _ROCM_FORWARD_METHODS.get(method, method)
```

评论区精华

Reviewer Fridge003 提出的核心建议是采用独立 forward method 的方式替代独立函数："Adding two new attention forward methods dedicated for rocm (for example, MHA_ROCM, MLA_ROCM)"，作者采纳并落地。此外，关于 _is_hip 是否涵盖 AITER 情况，Fridge003 询问是否需要额外加 _is_aiter 判断，作者 dpeng2333 澄清 _is_hip 已经覆盖所有 AITER 场景（_use_aiter = get_bool_env_var("SGLANG_USE_AITER") and _is_hip），无需额外条件。另有测试子用例删除的快速修正，作者均已处理。

- Mixin 命名与职责划分 (design): 作者接受并完成重命名，避免 fused rope 实现与新的 ROCm MLA 实现混淆。
- 测试子用例精简 (testing): 作者回复 Done，已删除冗余子测试。
- _is_hip 覆盖范围确认 (question): dpeng2333 澄清："_is_hip already covers all AITER cases, _use_aiter = get_bool_env_var(\"SGLANG_USE_AITER\") and _is_hip"，无需额外判断。

风险与影响

- 风险：本次重构涉及 DeepSeek 注意力核心路径，主要风险集中在： 1) deepseek_v2.py 中新增枚举分发分支，若 resolve_rocm_forward_method 映射遗漏（如

MHA_CHUNKED_KV 无 ROCm 入口），可能导致 HIP 平台运行时走错方法；2)forward_mla.py/forward_mha.py 删除 AMD 分支后，若存在未被移除的隐藏依赖（如 _skip_rope_for_dsa_tilelang_fused 等符号的引用），会导致 ROCm 或 CUDA 路径回归； 3) DeepseekMLARocmForwardMixin 重命名影响外部导入，需确认所有引用点已同步更新； 4) 共享函数去私有化（如 is_dcp_mla_decode_phase）改变了模块公共 API，可能影响其他调用方。

- 影响：直接影响 AMD/ROCm 平台上所有 DeepSeek 系列模型（V3、V3.2 等）的 MHA/MLA 前向推理路径，但行为保持不变。对 CUDA 用户无行为影响，反而因共享路径更简洁而降低维护成本。对 SGLang 开发者，代码结构更清晰，平台相关内核选择被隔离，后续新增 ROCm 优化无需触碰共享代码。同时，非 AMD 构建不再 import aiter，减少依赖面。
- 风险标记：核心路径重构，平台行为回归风险，mixin 重命名影响外部引用，新增枚举需同步下游，缺少性能基准

关联脉络

- PR #33888 config: delete the dead get_server_args() bindings across the repo: 同为 DeepSeek 注意力路径重构系列，修改了 deepseek_v2.py 与 attention_forward_methods 下的多个文件，与本 PR 有直接文件交集。
- PR #33889 moe: the shared-experts-fusion decision is a per-runner value the loader installs: 同样涉及 deepseek_v2.py 与模型装配逻辑，属于同一轮 DeepSeek 相关代码整理。
- PR #33981 [AMD] Add K3 verified mla kernel for DSpark on triton backend: AMD 平台 MLA 内核相关工作，与本 PR 的 ROCm MLA 路径同属 AMD 适配方向。