PR #47419 完整报告

vllm-project/vllm

[ROCm] Enable DeepSeek-V4 DSpark speculative decoding on AMD (MI350X / MI355X, gfx950)

合并时间：2026-07-10 23:22

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47419

执行摘要

- 一句话：AMD ROCm 启用 DeepSeek-V4 DSpark 推测解码
- 推荐动作：值得精读，特别是对跨平台模型移植和推测解码实现的工程师。重点关注 model.py 的 aux hidden state 收集设计和 rocm.py 的缓冲区修复策略。建议在合并后尽快补充 amd/dspark.py 的单元测试，并跟进 reviewer 对配置验证的质疑。

功能与动机

DSpark 推测解码之前仅在 NVIDIA GPU 上受支持，vllm/models/deepseek_v4/__init__.py 在 ROCm 分支上将 DSparkDeepseekV4ForCausalLM 设为 None，导致 AMD GPU 上使用 --speculative-config '{"method":"dspark",...}' 时模型加载失败。本 PR 消除了该平台限制，让 AMD GPU 用户也能获得 DSpark 带来的推理加速。

实现拆解

1. 新增草案模型模块(vllm/models/deepseek_v4/amd/dspark.py)：创建约 500 行的 ROCm DSpark 草案模型实现。它从 NVIDIA 版本移植，但改用 AMD 的 DeepseekV4DecoderLayer（aiter/triton 注意力 + MHC CustomOp 路径），通过 HCHeadOp 分发器路由 MHC 头，并移除了 NVIDIA 特有的 mega-MoE 权重路径。
2. 扩展目标模型接口(vllm/models/deepseek_v4/amd/model.py)：为目标模型 DeepseekV4Model 添加 EagleModelMixin 接口，在 forward 方法中收集 aux_hidden_state_layers 指定的目标层（DSpark 默认使用第 58、59、60 层）的重建隐藏状态。对 fused 路径手动调用 hc_post，对 unfused 路径直接使用 hidden_states，并通过 mean(dim=1) 降维传递给草案模型。同时优化了 fused 路径下 hc_post 重复计算的问题。
3. 修复 SWA 缓冲区大小(vllm/models/deepseek_v4/amd/rocm.py)：DeepseekV4ROCMAiterSparseSWAMetadataBuilder 的持久 ragged 索引缓冲区原固定使用 window_size，DSpark 的非因果路径需要更大宽度。将分配大小改为 max(window_size, noncausal_index_width)，并将 _copy_ragged_to_graph_buffers 的 max_entries 参数改为实际索引宽度，避免运行时越界崩溃。
4. 配置验证增强(vllm/config/speculative.py)：在 SpeculativeConfig.__post_init__ 中添加 if method == "dspark" 分支，检查 num_speculative_tokens 是否小于 dspark_block_size，若是则抛出清晰的 ValueError，防止用户配置错误导致输出乱码。

5. 注册与测试适配(vllm/models/deepseek_v4/__init__.py、tests/models/test_registry.py)：在 ROCm 分支中将 DSparkDeepseekV4ForCausalLM 从 None 改为从 .amd.dspark 导入；在注册测试中将跳过条件从 not is_cuda() 扩展为 not (is_cuda() or is_rocm())，使 ROCm 平台能通过模型导入测试。

关键文件：

- vllm/models/deepseek_v4/amd/dspark.py（模块 草案模型；类别 source；类型 data-contract；符号 DSparkDeepseekV4Model, init, embed_input_ids, combine_hidden_states）：新增的 ROCm DSpark 草案模型实现（~500 行），是 PR 的核心：定义 DSparkDeepseekV4Model 和 DSparkDeepseekV4ForCausalLM，封装半自回归块草案器。
- vllm/models/deepseek_v4/amd/model.py（模块 目标模型；类别 source；类型 data-contract；符号 DeepseekV4Model, DeepseekV4ForCausalLM, forward）：修改目标模型 forward，添加 EAGLE3 aux hidden state 接口，是 DSpark 草案模型获取目标模型中间层隐藏状态的关键依赖。
- vllm/models/deepseek_v4/__init__.py（模块 模型入口；类别 source；类型 entrypoint；符号 DSparkDeepseekV4ForCausalLM）：修改模型入口，在 ROCm 分支导入 DSpark 代替设为 None，使模型注册能正确加载。
- vllm/models/deepseek_v4/amd/rocm.py（模块 SWA 元数据；类别 source；类型 core-logic；符号 DeepseekV4ROCMAiterSparseSWAMetadataBuilder）：修复 DSpark 非因果 SWA 索引导致缓冲区溢出的底层问题，保证指定路径正确运行。
- vllm/config/speculative.py（模块 推测配置；类别 source；类型 core-logic；符号 SpeculativeConfig.post_init）：添加 DSpark 特有验证，防止用户配置错误的 num_speculative_tokens 导致异常。
- tests/models/test_registry.py（模块 注册测试；类别 test；类型 test-coverage；符号 test_registry_imports）：更新注册测试跳过条件，反映 DSpark 现在在 ROCm 上也支持。

关键符号：DSparkDeepseekV4Model.init, DSparkDeepseekV4Model.forward, DSparkDeepseekV4Model.embed_input_ids, DSparkDeepseekV4Model.combine_hidden_states, DSparkDeepseekV4ForCausalLM.init, DeepseekV4Model.forward, DeepseekV4ForCausalLM.init, DeepseekV4ROCMAiterSparseSWAMetadataBuilder.init, SpeculativeConfig.post_init

关键源码片段

vllm/models/deepseek_v4/amd/dspark.py

新增的 ROCm DSpark 草案模型实现（~500 行），是 PR 的核心：定义 DSparkDeepseekV4Model 和 DSparkDeepseekV4ForCausalLM，封装半自回归块草案器。

```
# SPDX-License-Identifier: Apache-2.0
"""DSpark draft model for DeepSeek-V4 on ROCm/AMD (gfx950).

ROCm port of nvidia/dspark.py. 遵循与 amd/mtp.py 相同的移植方案：
• 从 AMD .model 导入 DeepseekV4DecoderLayer（aiter/triton 注意力 + MHC CustomOp
路径）而不是 nvidia 版本；
```

• 通过 HCHeadOp CustomOp 分发器路由 MHC 头，并在 use_fused_mhc == False（aiter 路径，decoder layer 已内联 hc_post）时关闭 trailing mhc_post；
• 移除 mega-MoE 权重路径（amd/model.py 中不存在）。
其余部分为纯 torch/Triton，与 nvidia 实现一致。
"""

```
from collections.abc import Iterable
import regex as re
import torch
import torch.nn as nn
from vllm.config import VllmConfig, get_current_vllm_config
from vllm.model_executor.layers.mhc import HCHeadOp
from vllm.model_executor.models.qwen3_dspark import DSparkMarkovHead
from .model import DeepseekV4DecoderLayer

logger = init_logger(__name__)
# MoE 专家 scale 后缀随专家权重 dtype 变化（fp4 用 .weight_scale，block-fp8 用 .weight_scale_inv）
_EXPERT_SCALE_RE = re.compile(r"\.experts\.\d+\.w[123]\.scale$")

class DSparkDeepseekV4Model(nn.Module):
    """DSpark 草案模型：接收 target 模型多个中间层的平均隐藏状态，
    经过合并映射、多个 decoder layer 和 Markov 头输出半自回归块。"""
    def __init__(self, *, vllm_config: VllmConfig, prefix: str = "") -> None:
        super().__init__()
        assert vllm_config.speculative_config is not None
        config = vllm_config.speculative_config.draft_model_config.hf_config
        self.config = config
        self.hidden_size = config.hidden_size
        self.hc_mult = config.hc_mult
        self.num_hidden_layers = config.num_hidden_layers
        # DSpark 使用的 target 模型层索引，如 [58, 59, 60]
        self.target_layer_ids = tuple(config.dspark_target_layer_ids)

        # 草案模型的 decoder layer 数量，通常为 3，从 checkpoint 配置读取
        self.num_dspark_layers = getattr(config, "n_mtp_layers", None) or 3

        # 共享目标模型的 embedding（由 speculator 加载工具别名引用）
        self.embed_tokens = VocabParallelEmbedding(
            config.vocab_size, config.hidden_size,
            prefix=maybe_prefix(prefix, "embed_tokens"),
        )

        # 将多个 target 层的隐藏状态拼接后映射到 hidden_size（main_proj + main_norm）
        self.main_proj = ReplicatedLinear(
            config.hidden_size * len(self.target_layer_ids),
            config.hidden_size, bias=False, return_bias=False,
            quant_config=vllm_config.quant_config,
            prefix=maybe_prefix(prefix, "main_proj"),
```

```
        )
        self.main_norm = RMSNorm(config.hidden_size, eps=config.rms_norm_eps)

        # 创建 n_dspark_layers 个 AMD 版的 decoder layer
        current_vllm_config = get_current_vllm_config()
        self.layers = nn.ModuleList([
            DeepseekV4DecoderLayer(
                current_vllm_config,
                prefix=maybe_prefix(prefix, f"layers.{self.num_hidden_layers + i}"),
            )
            for i in range(self.num_dspark_layers)
        ])

        # 最终 norm + hyper-computation (HC) 头：参数从 mtp.* 权重加载
        self.norm = RMSNorm(config.hidden_size, eps=config.rms_norm_eps)
        hc_dim = self.hc_mult * config.hidden_size
        self.hc_head_fn = nn.Parameter(
            torch.empty(self.hc_mult, hc_dim, dtype=torch.float32), requires_grad=False)
        self.hc_head_base = nn.Parameter(
            torch.empty(self.hc_mult, dtype=torch.float32), requires_grad=False)
        self.hc_head_scale = nn.Parameter(
            torch.empty(1, dtype=torch.float32), requires_grad=False)

        # Markov 头：生成块内多个 draft tokens
        draft_vocab_size = getattr(config, "draft_vocab_size", None) or config.vocab_size
        # ... 后续 Markov 头初始化
```

vllm/models/deepseek_v4/amd/model.py

修改目标模型 forward，添加 EAGLE3 aux hidden state 接口，是 DSpark 草案模型获取目标模型中间层隐藏状态的关键依赖。

```
# DeepseekV4Model.forward 中的关键修改：收集目标层的 aux hidden states
# 该部分插入在 decoder layer 循环体内
aux_hidden_states: list[torch.Tensor] = []
final_aux_recon: torch.Tensor | None = None

for idx, layer in enumerate(
    islice(self.layers, self.start_layer, self.end_layer),
    start=self.start_layer,
):
    hidden_states, residual, post_mix, res_mix = layer(
        hidden_states, positions, input_ids, post_mix, res_mix, residual
    )

    # 检查当前层是否在 aux_hidden_state_layers 中（由 DSpark 配置的 target 层）
    if (idx + 1) in self.aux_hidden_state_layers:
        # fused 路径（use_fused_mhc）：layer 内部未调用 hc_post，需手动重建
        if layer.use_fused_mhc:
            aux_recon = layer.hc_post(hidden_states, residual, post_mix, res_mix)
```

```
                final_aux_recon = aux_recon # 缓存最后一层的结果，避免循环后重复计算
            else:
                # unfused (aiter) 路径：layer 已经应用了 hc_post，hidden_states 即为重建状态
                aux_recon = hidden_states
            # 对隐藏状态做 hc_mult 维度平均，得到 [T, hidden_size] 的表示
            aux_hidden_states.append(aux_recon.mean(dim=1))

    # 循环结束后，对最后一层应用 hc_post（如果 fused 且未在 aux 中缓存）
    if layer is not None and layer.use_fused_mhc:
        if (final_aux_recon is not None
                and self.end_layer in self.aux_hidden_state_layers):
            hidden_states = final_aux_recon
        else:
            hidden_states = layer.hc_post(hidden_states, residual, post_mix, res_mix)

    # 后续 norm 和 head 处理，若有 aux_hidden_states 则一同返回
    if len(aux_hidden_states) > 0:
        return hidden_states, aux_hidden_states
    return hidden_states
```

评论区精华

1. hc_post 重复计算（已解决）

> dllehr-amd（reviewer）：*"you are calling this twice if we are fused. Can you reuse this one over there?"* larryli2-amd（作者）：*"cached the final layer's hc_post output as final_aux_recon and reuse it after the loop, so it's no longer computed twice when the last layer is also an aux layer."*

2. 测试条件优化建议（已解决）

> AndreasKaratzas（reviewer）：*"nit: This could have been is_cuda_alike(). But if there are no further changes then just leave it as is."* larryli2-amd（作者）：*"Thanks for the review! I'll leave it as it is for now."*

3. 配置验证准确性（未解决）

> benchislett（reviewer）：*"What is this referencing? I think this is false. Could this be an unrelated bug in your implementation? What's stopping us from using the effective block size during verification??"* larryli2-amd（作者）：*"I'm on marriage leave. I'll be back at work on August 3."*（未给出实质性答复，需后续跟进）

- hc_post 在 fused 路径下重复计算 (performance): 作者将最终 aux layer 的 hc_post 结果缓存为 final_aux_recon，并在其后复用，避免了重复计算。
- test_registry.py 条件优化建议 (style): 作者回复保持现状。
- DSpark 配置验证的准确性 (correctness): 作者正在婚假，未给出实质性回复，需后续澄清。

风险与影响

- 风险：
  1. 目标模型 forward 改动风险：DeepseekV4Model.forward 新增 aux hidden states 收集逻辑，可能影响现有 EAGLE3/DFlash 功能及 fused/unfused 分支的行为。需确保 aux_hidden_state_layers 为空时路径与原行为完全一致。
  2. 新增代码缺少独立测试：amd/dspark.py 约 500 行，但无单元测试，仅依赖集成测试，长期维护和回归风险较高。
  3. MHC 路径差异：AMD 的 use_fused_mhc 默认为 False（aiter 路径），与 NVIDIA 的 fused 路径存在行为差异（hc_post 是否内联到 decoder layer）。DSpark 草案模型在此分支下的行为可能不如 NVIDIA 稳定。
  4. 验证条件争议：num_speculative_tokens >= dspark_block_size 的硬限制受到 reviewer 质疑，可能过于严格或基于不准确的假设。若确实允许更小的值，该验证会错误阻止用户，且错误信息可能产生误导。 - 影响：用户：AMD GPU（MI350X/MI355X）用户可使用 DSpark 以 ~1.59× 加速 DeepSeek-V4-Pro 推理，无需额外配置。系统：无新依赖项。新增的 amd/dspark.py 由 AMD 子包管理，不改变 NVIDIA/XPU 路径。团队：AMD 团队需持续同步 NVIDIA 侧 DSpark 的后续变更（如 kernel 升级），保持两个实现的 API 一致性。
- 风险标记：目标模型 forward 改动（核心路径），新增 ~500 行代码无独立测试，MHC 路径差异可能影响行为，验证边界可能过严

关联脉络

- PR #48044 [ROCm] Fused Shared Expert Support for AMD Quark DeepSeek-V4 Model Checkpoints: 同为 AMD 上的 DeepSeek-V4 模型支持，为本次 DSpark 移植提供了基础（amd/model.py 等）。
- PR #48993 [Core][DSV4] Compact MXFP4 indexer KV cache and packed group overlays: DeepSeek-V4 的性能优化，与本 PR 同属 V4 模型在存储和推理效率上的持续改进。