PR #43746 完整报告

vllm-project/vllm

[Model Refactoring] Remove torch compile dependency in DSv4

合并时间：2026-05-28 22:26

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43746

执行摘要

- 一句话：移除 DS V4 对 torch.compile 的依赖，改用可中断 CUDA 图
- 推荐动作：值得精读。本 PR 展示了如何通过手动融合 kernel 和利用 breakable CUDA graph 替换 torch.compile，是 vLLM 编译栈演进的重要一步。特别关注 fused_mtp_input_rmsnorm.py 中的 kernel 设计以及 config.py 中的自动启用策略。

功能与动机

DeepSeek V4 模型原来依赖 torch.compile 进行图编译以支持 CUDA 图。torch.compile 带来了启动开销和兼容性问题。通过使用 breakable CUDA graph（#42304），可以在不依赖 torch.compile 的情况下获得分段式 CUDA 图的性能收益，简化编译栈，并减少编译时间和内存消耗。

实现拆解

1. 创建融合 MTP 输入 RMSNorm 的 Triton kernel：新增 vllm/models/deepseek_v4/common/ops/fused_mtp_input_rmsnorm.py，将位置掩码、enorm、hnorm 合并为一个 kernel，减少内核启动次数。
2. 修改 amd 和 nvidia 的 mtp.py：替换原来的顺序操作为融合 kernel，并删除 @support_torch_compile 装饰器和相关导入；同时将 shared head RMSNorm 也替换为融合 kernel。
3. 从 amd 和 nvidia 的 model.py 中删除 torch.compile 依赖：移除 @support_torch_compile 装饰器及 from vllm.compilation.decorators import support_torch_compile。
4. 配置自动启用 breakable CUDA graph：在 vllm/config/vllm.py 的 __post_init__ 中，当模型架构是 DS V4 时自动设置 VLLM_USE_BREAKABLE_CUDAGRAPH=1，除非用户已显式赋值。
5. 确保 drafter 也使用 breakable CUDA graph：在 vllm/v1/worker/gpu_model_runner.py 的 load_model 方法中，除了主模型外，也对 speculative drafter 包裹 BreakableCUDAGraphWrapper。 本 PR 未包含专门的测试文件变更，但依赖已有的模型测试覆盖。

关键文件：

- vllm/models/deepseek_v4/common/ops/fused_mtp_input_rmsnorm.py（模块 融合算子；类别 source；类型 infrastructure；符号 _rmsnorm_row, _fused_mtp_input_rmsnorm_kernel, _mtp_shared_head_rmsnorm_kernel, mtp_shared_head_rmsnorm）：新增融合 MTP 输入 RMSNorm 的 Triton kernel，是移除 torch.compile 依赖的核心替换。
- vllm/models/deepseek_v4/amd/mtp.py（模块 AMD 模型；类别 source；类型 data-contract）：AMD 平台的 MTP 模型，替换前向过程使用融合 kernel，删除 torch.compile 依赖。
- vllm/models/deepseek_v4/nvidia/mtp.py（模块 NVIDIA 模型；类别 source；类型 data-contract）：NVIDIA 平台的 MTP 模型，与 AMD 版本同步修改，替换前向过程使用融合 kernel。
- vllm/config/vllm.py（模块 配置；类别 source；类型 core-logic）：配置层自动启用 breakable CUDA graph，对 DS V4 模型无缝生效。
- vllm/v1/worker/gpu_model_runner.py（模块 推理引擎；类别 source；类型 data-contract）：确保 speculative drafter 也使用 breakable CUDA graph wrapper，保持一致性。

关键符号：fused_mtp_input_rmsnorm, mtp_shared_head_rmsnorm, _fused_mtp_input_rmsnorm_kernel, _rmsnorm_row, DeepSeekV4MultiTokenPredictorLayer.forward, DeepSeekV4MTP.init

关键源码片段

vllm/models/deepseek_v4/common/ops/fused_mtp_input_rmsnorm.py

新增融合 MTP 输入 RMSNorm 的 Triton kernel，是移除 torch.compile 依赖的核心替换。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
"""Fused MTP-input RMSNorm: enorm (with mask-zero at position 0) + hnorm.

Replaces the eager sequence at the top of the MTP draft forward:
  inputs_embeds = torch.where(positions.unsqueeze(-1) == 0, 0, inputs_embeds)
  inputs_embeds = self.enorm(inputs_embeds)
  previous_hidden_states = previous_hidden_states.view(-1, hc_mult, H)
  previous_hidden_states = self.hnorm(previous_hidden_states)
which lowers to ~6 small kernels on the breakable-cudagraph path.
"""

import torch
from vllm.triton_utils import tl, triton


@triton.jit
def _rmsnorm_row(
    x, w_ptr, out_row_ptr, block, mask, eps, HIDDEN: tl.constexpr,
):
```

```
    # 单行 RMSNorm: x -> (x / rms) * weight
    x = x.to(tl.float32)
    variance = tl.sum(x * x, axis=0) / HIDDEN
    rrms = tl.rsqrt(variance + eps)
    w = tl.load(w_ptr + block, mask=mask, other=0.0).to(tl.float32)
    y = x * rrms * w
    tl.store(out_row_ptr + block, y.to(out_row_ptr.dtype.element_ty), mask=mask)


@triton.jit
def _fused_mtp_input_rmsnorm_kernel(
    inputs_embeds_ptr, positions_ptr, prev_hidden_ptr,
    enorm_weight_ptr, hnorm_weight_ptr,
    enorm_out_ptr, hnorm_out_ptr,
    eps, HIDDEN: tl.constexpr, HC_MULT: tl.constexpr, BLOCK_SIZE: tl.constexpr,
):
    # task 0: enorm; tasks 1..HC_MULT: hnorm
    token_idx = tl.program_id(0).to(tl.int64)
    pid_task = tl.program_id(1)
    block = tl.arange(0, BLOCK_SIZE)
    mask = block < HIDDEN
    if pid_task == 0:
        # enorm path: 加载 inputs_embeds 并在 position==0 处置零
        pos = tl.load(positions_ptr + token_idx)
        keep = pos != 0
        x = tl.load(inputs_embeds_ptr + token_idx * HIDDEN + block, mask=mask, other=0.0)
        x = tl.where(keep, x, 0.0)
        _rmsnorm_row(x, enorm_weight_ptr, enorm_out_ptr + token_idx * HIDDEN, block, mask,
        eps, HIDDEN)
    else:
        # hnorm path: 加载 prev_hidden[token, slot, :]
        slot = pid_task - 1
        row_offset = (token_idx * HC_MULT + slot) * HIDDEN
        x = tl.load(prev_hidden_ptr + row_offset + block, mask=mask, other=0.0)
        _rmsnorm_row(x, hnorm_weight_ptr, hnorm_out_ptr + row_offset, block, mask, eps,
        HIDDEN)


def fused_mtp_input_rmsnorm(
    inputs_embeds, positions, previous_hidden_states,
    enorm_weight, hnorm_weight, eps, hc_mult,
):
    # 融合入口：reshape 输出并启动 kernel
    T = inputs_embeds.shape[0]
    H = inputs_embeds.shape[1]
    assert previous_hidden_states.shape == (T, hc_mult, H)
    enorm_out = torch.empty_like(inputs_embeds)
    hnorm_out = torch.empty_like(previous_hidden_states)
    grid = (T, hc_mult + 1)
```

```
    _fused_mtp_input_rmsnorm_kernel[grid](
        inputs_embeds, positions, previous_hidden_states,
        enorm_weight, hnorm_weight,
        enorm_out, hnorm_out,
        eps, H, hc_mult, triton.next_power_of_2(H),
    )
    return enorm_out, hnorm_out
```

vllm/config/vllm.py

配置层自动启用 breakable CUDA graph，对 DS V4 模型无缝生效。

```
        # DeepSeek V4 的模型类不使用 @support_torch_compile ——
        # breakable cudagraph 是支持的 PIECEWISE 路径。自动启用它
        # 除非用户已通过环境变量显式选择退出。
        if (
            self.model_config is not None
            and "VLLM_USE_BREAKABLE_CUDAGRAPH" not in os.environ
            and any(
                a in ("DeepseekV4ForCausalLM", "DeepSeekV4MTPModel")
                for a in self.model_config.architectures
            )
        ):
            os.environ["VLLM_USE_BREAKABLE_CUDAGRAPH"] = "1"
            logger.info_once(
                "Auto-enabling VLLM_USE_BREAKABLE_CUDAGRAPH=1 for DeepSeek V4. "
                "Set VLLM_USE_BREAKABLE_CUDAGRAPH=0 to opt out."
            )
```

评论区精华

主要讨论集中在是否默认启用 breakable CUDA graph：

- ZJY0516建议："Let's set VLLM_USE_BREAKABLE_CUDAGRAPH=1 by default for ds v4?"
- 作者采纳并在后续提交（e22a1f9）中实现了自动启用。
- 默认启用 breakable CUDA graph (design): 作者采纳并在后续提交中实现了自动启用（e22a1f9）。

风险与影响

- 风险：
  - 破坏 MRV2 兼容性：PR 正文明确指出 MRV2 不支持 breakable CUDA graph，因此该变更会暂时破坏 MRV2 与 DS V4 的兼容性，需要后续修复。
  - breakable CUDA graph 稳定性风险：该特性相对较新，可能在特定硬件或配置下出现未知问题。
  - 手动融合 Kernel 的精度风险：新增的 Triton kernel 虽然保持了数学等价性，但在不同 GPU 架构上可能存在数值差异。

- 环境变量覆盖：自动设置 VLLM_USE_BREAKABLE_CUDAGRAPH 可能意外覆盖用户显式设置的值（如设置 0），但代码仅在环境变量未定义时设置，设计上避免了覆盖。
- 影响：影响范围：仅限于 DeepSeek V4 模型。用户无需手动设置 VLLM_USE_BREAKABLE_CUDAGRAPH，但如果之前依赖 torch.compile 的优化，现在将自动切换到 breakable CUDA graph。性能持平或略有提升。对团队而言，移除了 torch.compile 依赖，简化了编译栈维护，但 breakable CUDA graph 仍处于早期阶段。破坏性：MRV2 用户暂时无法使用 DS V4，需等待后续兼容性修复。
- 风险标记：破坏 MRV2 兼容性，核心编译路径变更，手动融合 Kernel 精度风险，环境变量覆盖潜在问题（已设计避免）

关联脉络

- PR #42304 [Feature] Breakable CUDA graph support: 本 PR 依赖可中断 CUDA 图功能，该 PR 提供了基础支持。