PR #42758 完整报告

vllm-project/vllm

Enable perf_token_group_quant/_C_stable_libtorch for ROCm

合并时间：2026-06-03 14:23

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/42758

执行摘要

- 一句话：启用 ROCm 的 per-token-group 量化内核
- 推荐动作：值得精读，特别是 cmake/hipify.py 的路径处理改进和内核的 warp 适配。设计上使用 is_cuda_alike 统一平台检查的做法值得借鉴。但建议在 MI300X 等目标 GPU 上进行充分的回归测试和精度对比。

功能与动机

ROCm 平台此前缺乏对 per-token-group 量化的原生支持，导致 FP8 量化在 AMD GPU 上无法利用高效的 CUDA 内核，只能回退到 Triton 实现或更慢的 fallback。此 PR 旨在通过启用 _C_stable_libtorch 编译目标并将核心量化内核迁移至 HIP，使 ROCm 用户也能获得与 CUDA 相当的量化性能。

实现拆解

1. 构建系统与 ABI 稳定目标：更新 CMakeLists.txt 和 csrc/libtorch_stable/torch_bindings.cpp，移除量化算子周围的 #ifndef USE_ROCM 守卫，使得算子定义和实现在 HIP 下也可编译。同时将 permute_cols 等仅 CUDA 的操作保留守卫。
2. 内核适配：在 per_token_group_quant.cu 中，将 CUDA 特有的 __shfl_xor_sync 等 warp 原语替换为 HIP 兼容形式，并适配 64 线程束（warp）大小。实现 GroupReduceMax 的设备函数以支持 64 warp 的归约。
3. HIPify 增强：在 cmake/hipify.py 中新增 _expected_hip_build_path 函数，用于计算预期的 .hip 文件路径，以处理 PyTorch HIPify 跳过无变化文件写入时 CMake 期望 .hip 文件的问题。同时设置 header_include_dirs=["."]，让 hipify 能解析 csrc/ 相对路径的 #include 头文件。
4. Python 层统一与融合 pass：在 fp8_utils.py 和 int8_utils.py 中，将平台检查从 current_platform.is_cuda() 替换为 current_platform.is_cuda_alike()，使快速路径覆盖 ROCm。同时在 rms_quant_fusion.py 中移除 if current_platform.is_cuda(): 守卫，组量化融合模式默认在所有 CUDA 类平台（包括 ROCm）注册。更新测试文件，添加 ROCm 相关的参数化和标记。

关键文件：

- cmake/hipify.py（模块 构建脚本；类别 source；类型 core-logic；符号 _expected_hip_build_path）：核心构建脚本，新增 _expected_hip_build_path 函数，修复

hipify 对相对路径头文件的处理和 .hip 文件生成，是 ROCm 编译成功的关键。

- vllm/compilation/passes/fusion/rms_quant_fusion.py（模块 编译优化；类别 source；类型 core-logic）：融合 pass 的关键调整：移除 is_cuda() 守卫，使组量化融合模式在 ROCm 上注册，同时引入 get_fp8_min_max 统一 FP8 边界值获取。
- vllm/model_executor/layers/quantization/utils/fp8_utils.py（模块 量化工具；类别 source；类型 data-contract；符号 per_token_group_quant_fp8, per_token_group_quant_fp8_packed_for_deepgemm）：平台检查替换 from is_cuda() to is_cuda_alike()，扩展快速路径到 ROCm；更新 per_token_group_quant_fp8_packed_for_deepgemm 的 assert 以允许 ROCm。
- csrc/libtorch_stable/torch_bindings.cpp（模块 稳定 ABI；类别 source；类型 core-logic）：移除量化算子定义和实现周围的 USE_ROCM 守卫，使它们在 HIP 下也可用；重新排序 permute_cols 守卫使其仅适用于 CUDA。
- csrc/libtorch_stable/quantization/w8a8/fp8/per_token_group_quant.cu（模块 量化内核；类别 source；类型 dependency-wiring；符号 per_token_group_quant_8bit, GroupReduceMax）：内核适配的核心文件：将 CUDA warp shuffle 替换为 HIP 版本，支持 64 线程束大小，修改 GroupReduceMax 以兼容 ROCm。
- tests/kernels/quantization/test_per_token_group_quant.py（模块 测试；类别 test；类型 test-coverage）：测试覆盖扩展：添加 ROCm 标记和参数化，确保 ROCm 上量化测试通过。

关键符号：_expected_hip_build_path, per_token_group_quant_fp8, per_token_group_quant_fp8_packed_for_deepgemm

关键源码片段

cmake/hipify.py

核心构建脚本，新增 _expected_hip_build_path 函数，修复 hipify 对相对路径头文件的处理和 .hip 文件生成，是 ROCm 编译成功的关键。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project

import argparse
import os
import shutil

from torch.utils.hipify.hipify_python import get_hip_file_path, hipify


def _expected_hip_build_path(source_abs: str, output_directory: str) -> str:
    '''计算 HIPify 输出路径，与 PyTorch 内部命名一致。

    当 PyTorch 跳过无变化文件的写入时，hipified_path 仍指向 .cu 文件，
    而 CMake 需要 .hip 文件。此函数生成 CMake 期望的路径。
    '''
    rel = os.path.relpath(source_abs, output_directory)
```

```
    return os.path.abspath(
        os.path.join(
            output_directory, get_hip_file_path(rel, is_pytorch_extension=True)
        )
    )


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    # ( 参数解析省略 )
    args = parser.parse_args()

    includes = [os.path.join(args.project_dir, '*')]
    extra_files = [os.path.abspath(s) for s in args.sources]

    shutil.copytree(args.project_dir, args.output_dir, dirs_exist_ok=True)

    hipify_result = hipify(
        project_directory=args.project_dir,
        output_directory=args.output_dir,
        # 添加头文件搜索路径，使 hipify 能解析 csrc/ 下的相对路径 #include
        header_include_dirs=['.'],
        includes=includes,
        extra_files=extra_files,
        show_detailed=True,
        is_pytorch_extension=True,
        hipify_extra_files_only=True,
    )

    hipified_sources = []
    for source in args.sources:
        s_abs = os.path.abspath(source)
        if s_abs in hipify_result and hipify_result[s_abs].hipified_path is not None:
            path = hipify_result[s_abs].hipified_path
            # 处理 PyTorch 跳过写入的情况
            if s_abs.endswith('.cu') and path.endswith('.cu'):
                dest = _expected_hip_build_path(s_abs, args.output_dir)
                if os.path.normpath(path) != os.path.normpath(dest):
                    os.makedirs(os.path.dirname(dest), exist_ok=True)
                    shutil.copy2(path, dest)
                hipified_s_abs = dest
            else:
                hipified_s_abs = path
        else:
            hipified_s_abs = s_abs
        hipified_sources.append(hipified_s_abs)

    # 输出 hipified 文件路径给 CMake
    print('\\n'.join(hipified_sources))
```

评论区精华

Review 中围绕平台抽象和 ABI 稳定迁移展开讨论：

- Harry-Chen 询问是否需要在融合 pass 内保留 is_cuda_alike 守卫，charlifu 回应控制逻辑应放在 pass 外部，且动态量化模式已无守卫，因此移除 is_cuda 守卫是合理的。
- janeyx99 对 AMD 参与此工作表示兴奋，并确认 ABI 稳定部分 LGTM；同时询问内核中 FP8 类型分发是否覆盖 e5m2，charlifu 解释 VLLM_STABLE_DISPATCH_FP8_TYPES 仅分发 e4m3 和 e4m3fnuz。
- 关于 rocm.py 中 import warning 的强度，charlifu 指出相同代码已在上游 PR 中，因此决定移除重复部分。
- 团队内部协调方面，提到了与 #39513 的潜在重叠，但确认无重复。
- 融合 pass 中平台守卫的必要性 (design): 同意移除，不添加守卫。
- FP8 类型分发范围 (correctness): 确认范围，无变更。
- ROCM 平台中 import warning 的强度 (design): 移除重复的 import 尝试，保留现有 warning 强度。
- 内部工作协调 (other): 无重复，各自独立。

风险与影响

- 风险：主要风险包括：(1) hipify.py 的头文件路径处理可能在某些项目结构下失败，导致头文件未被转换或编译错误；(2) warp size 64 的 shuffle 实现可能与 CUDA 有微小差异，影响量化结果数值； (3) 移除融合 pass 的平台守卫后，组量化模式可能在非 CUDA/ROCm 平台上注册并引发崩溃； (4) per_token_group_quant_packed 的 assert 改为 is_cuda_alike 后，可能在非支持的 ROCm 设备上被调用失败；(5) 量化精度一致性需在多种 AMD GPU 上验证，特别是 MI250 和 MI300X。
- 影响：对用户：ROCm 用户现在可以利用高效的 per-token-group 量化，提升 FP8 推理性能；CUDA 用户不受影响。对系统：构建增加 _C_stable_libtorch 目标，略微增加构建时间；运行时无明显变化。对团队：需维护 HIP 内核与 CUDA 内核的同步，确保未来功能扩展兼容。对测试：组量化测试在 ROCm 上全面覆盖，融合 pass 测试也通过调整后通过。
- 风险标记：HIPify 路径处理，warp size 差异，平台守卫移除，量化精度一致性

关联脉络

- 暂无明显关联 PR