PR #45176 完整报告

vllm-project/vllm

[11a/n] Migrate Marlin kernels to torch stable ABI

合并时间：2026-06-12 12:22

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/45176

执行摘要

本次 PR 将 Marlin 量化内核从 legacy _C 扩展迁移到 _C_stable_libtorch，实现 torch stable ABI 兼容。同时修复了 marlin_int4_fp8_preprocess 中默认 CUDA stream 导致的潜在竞态条件。整体影响面可控，测试通过。

功能与动机

继续 #26946 提出的 libtorch stable ABI 迁移。Marlin 内核是目前最后一组依赖不稳定 ABI 的量化核函数，将其迁移后可以消除对 torch::Tensor 等不稳定接口的依赖，降低 PyTorch 版本升级的兼容性成本。

实现拆解

1. 文件搬迁：将 csrc/quantization/marlin/ 下所有 .cu, .h, .py 文件移至 csrc/libtorch_stable/quantization/marlin/。
2. API 替换：替换所有 torch::Tensor 为 torch::stable::Tensor，TORCH_CHECK 为 STD_TORCH_CHECK，并改用 stable 头文件。
3. 算子注册切换：在 csrc/libtorch_stable/torch_bindings.cpp 新增 STABLE_TORCH_LIBRARY_FRAGMENT 注册，在 csrc/torch_bindings.cpp 删除对应注册。
4. Stream 修复：在 marlin_int4_fp8_preprocess.cu 中，使用 get_current_cuda_stream() 获取当前 stream 而非默认 stream。
5. 构建适配：更新 CMakeLists.txt，将新文件加入 _C_stable_libtorch 目标。
6. 测试验证：通过 tests/models/quantization/test_gptq_marlin.py 确保功能正常。

csrc/libtorch_stable/torch_bindings.cpp

在 stable libtorch 扩展中注册 Marlin 算子。

```
// csrc/libtorch_stable/torch_bindings.cpp
// 在 stable libtorch 扩展中注册 Marlin 算子

STABLE_TORCH_LIBRARY_FRAGMENT(_C, ops) {
  // ... 其他已有注册 ...

  // Marlin GEMM 主算子
  ops.def(
```

```
        "marlin_gemm(Tensor a, Tensor? c_or_none, Tensor b_q_weight, "
        "Tensor? b_bias_or_none,Tensor b_scales, "
        "Tensor? a_scales, Tensor? global_scale, Tensor? b_zeros_or_none, "
        "Tensor? g_idx_or_none, Tensor? perm_or_none, Tensor workspace, "
        "int b_type_id, SymInt size_m, SymInt size_n, SymInt size_k, "
        "bool is_k_full, bool use_atomic_add, bool use_fp32_reduce, "
        "bool is_zp_float) -> Tensor");

    // GPTQ 权重重排
    ops.def(
        "gptq_marlin_repack(Tensor b_q_weight, Tensor perm, "
        "SymInt size_k, SymInt size_n, int num_bits, bool is_a_8bit) -> Tensor");

    // AWQ 权重重排
    ops.def(
        "awq_marlin_repack(Tensor b_q_weight, SymInt size_k, "
        "SymInt size_n, int num_bits, bool is_a_8bit) -> Tensor");

    // int4 到 fp8 的预处理
    ops.def(
        "marlin_int4_fp8_preprocess(Tensor qweight, "
        "Tensor? qzeros_or_none, bool inplace) -> Tensor");
}
```

csrc/libtorch_stable/quantization/marlin/marlin_int4_fp8_preprocess.cu

新增的 stable ABI 内核文件，包含 stream 修复。

```
// csrc/libtorch_stable/quantization/marlin/marlin_int4_fp8_preprocess.cu
// 使用 torch stable ABI 重写的 int4 -> fp8 预处理函数。
// 关键改进：通过 get_current_cuda_stream 获取当前 stream，避免竞态条件。

#include "marlin.cuh"
#include <torch/csrc/stable/accelerator.h>
#include <torch/csrc/stable/library.h>
// ... 其他 include

torch::stable::Tensor marlin_int4_fp8_preprocess(
    torch::stable::Tensor& qweight,
    std::optional<torch::stable::Tensor> qzeros_or_none,
    bool inplace) {
  // 使用 stable 宏检查输入
  STD_TORCH_CHECK(qweight.is_cuda(), "qweight is not on GPU");
  STD_TORCH_CHECK(qweight.scalar_type() == torch::headeronly::ScalarType::Int,
                  "qweight.dtype != torch.int32");

  const int32_t device_index = qweight.get_device_index();
  torch::stable::accelerator::DeviceGuard device_guard(device_index);
  // 获取当前 CUDA stream，而非默认 stream
  const cudaStream_t stream = get_current_cuda_stream(device_index);
```

```
  torch::stable::Tensor output =
      inplace ? qweight : torch::stable::empty_like(qweight);

  if (!qzeros_or_none.has_value()) {
    // 处理无零点 (non-zp) 格式
    int blocks = qweight.numel() * 8 / 256;
    marlin_int4_fp8_preprocess_kernel_without_zp<<<blocks, 32, 0, stream>>>(
        reinterpret_cast<const int32_t*>(qweight.const_data_ptr()),
        reinterpret_cast<int32_t*>(output.mutable_data_ptr()));
  } else {
    // 处理 AWQ 格式（略）
  }
  return output;
}
```

评论区精华

- janeyx99对 stream 修复提出疑问："stream addition looks new, why is it needed now?" cleonard530 解释这是避免竞态条件的刻意改进。
- janeyx99指出 marlin.cu 包含非 stable 头文件 core/registration.h 和一行死代码，已确认移除。
- janeyx99询问 CMakeLists 中条件是否新加，cleonard530 说明是原条件迁移。
- janeyx99留意到 awq_marlin_repack 检查顺序变化，但未要求修改。

风险与影响

风险：本次迁移为等价替换，接口语义不变，且通过现有测试。主要风险是构建配置遗漏，但已由 review 确认。stream 修复降低了并发 bug 可能性。

影响：对用户无功能感知；对系统消除不稳定 ABI 依赖；对团队推进迁移进度。

关联脉络

该 PR 是 stable ABI 迁移系列的一部分，基于 #44565（已合并）的工作。整个系列完成后将可彻底移除 legacy _C 扩展。关联 Issue: #26946。