PR #47128 完整报告

vllm-project/vllm

[ROCm] [PyTorch] Move to stable abi since ROCm upgraded to torch 2.11

合并时间：2026-07-02 18:34

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47128

执行摘要

本 PR 在 ROCm 升级到 PyTorch 2.11 后，移除了之前兼容 2.10 而保留的 fallback 非 stableABI代码，统一使用stableABI，删除冗余文件并简化构建配置，降低了长期维护成本。

功能与动机

ROCm 此前基于 PyTorch 2.10，无法使用 stable ABI，因而在 PR#44648 中引入了回退注册路径。现在 ROCm 已升级到 PyTorch 2.11（PR#45362），可以清理这些临时代码，统一使用 _C_stable_libtorch 扩展，提升代码可维护性和构建一致性。

实现拆解

1. 移除旧扩展的歧义注册：删除 csrc/torch_bindings.cpp 中为 ROCm 保留的 TORCH_LIBRARY_EXPAND 块（get_cuda_view_from_cpu_tensor 和 _cuda_utils）。
2. 将操作转入 stable 扩展：在 csrc/libtorch_stable/torch_bindings.cpp 中移除 #ifndef USE_ROCM 守卫，使 get_cuda_view_from_cpu_tensor 和 cuda_utils 对 ROCm 也注册。
3. 统一头文件声明：在 csrc/libtorch_stable/ops.h 和 csrc/ops.h 中调整声明，使所有平台共享一个声明。
4. 迁移源文件：删除 csrc/cuda_view.cu（已迁移到 libtorch_stable/cuda_view.cu），将 cuda_utils_kernels.cu 移至 stable 扩展编译源列表。
5. 简化 CMake 构建：统一 TORCH_TARGET_VERSION 为 0x020B（PyTorch 2.11），移除之前为 CUDA 和 ROCm 分别设置的冗余宏定义。

csrc/torch_bindings.cpp

该文件是旧扩展注册入口，移除了 ROCm 回退注册块，是清理的核心

```
// Provides torch::Tensor for ops.h
#include <torch/all.h>
#include "ops.h"
#include "core/registration.h"
#include <torch/library.h>
#include <torch/version.h>

// 注意：此前这里有两个 TORCH_LIBRARY_EXPAND 块用于 ROCm 的
// get_cuda_view_from_cpu_tensor 和 cuda_utils 注册，现已全部删除。
// 这些函数现在通过 stable ABI 扩展（_C_stable_libtorch）注册。
```

```
#ifdef USE_ROCM
TORCH_LIBRARY_FRAGMENT(CONCAT(TORCH_EXTENSION_NAME, _custom_ar), custom_ar) {
  // Quick Reduce all-reduce kernels (ROCm-only) — 保留在旧扩展
  custom_ar.def(
      "qr_all_reduce(int fa, Tensor inp, Tensor out, int quant_level, bool "
      "cast_bf2half) -> ()");
  custom_ar.impl("qr_all_reduce", torch::kCUDA, &qr_all_reduce);
  custom_ar.def("init_custom_qr", &init_custom_qr);
  custom_ar.def("qr_destroy", &qr_destroy);
  custom_ar.def("qr_get_handle", &qr_get_handle);
  custom_ar.def("qr_open_handles(int _fa, Tensor[](b!) handles) -> ()");
  custom_ar.impl("qr_open_handles", torch::kCPU, &qr_open_handles);
  custom_ar.def("qr_max_size", &qr_max_size);
}
#endif

REGISTER_EXTENSION(TORCH_EXTENSION_NAME)
```

csrc/libtorch_stable/torch_bindings.cpp

stable 扩展的注册入口，统一了 get_cuda_view_from_cpu_tensor 的注册条件

```
#include "ops.h"
#include "cuda_utils.h"
#include "core/registration.h"
#include <torch/csrc/stable/library.h>

STABLE_TORCH_LIBRARY_FRAGMENT(_C, ops) {
  // ... 其他 op 注册 ...
  // 现在无条件注册，不再被 #ifndef USE_ROCM 守卫
  ops.def("get_cuda_view_from_cpu_tensor(Tensor cpu_tensor) -> Tensor");
  // 其他 CUDA-only 操作仍然受 #ifndef USE_ROCM 守卫
#ifndef USE_ROCM
  // ...
#endif
}

// 之前的 // TODO: Remove this once ROCm upgrade to torch 2.11. // #ifndef USE_ROCM 已移除
STABLE_TORCH_LIBRARY_IMPL(_C, CPU, ops) {
  ops.impl("get_cuda_view_from_cpu_tensor",
           TORCH_BOX(&get_cuda_view_from_cpu_tensor));
}

STABLE_TORCH_LIBRARY_FRAGMENT(_C_cuda_utils, cuda_utils) {
  // 现在无条件定义
  cuda_utils.def("get_device_attribute(int attribute, int device_id) -> int");
  cuda_utils.def(
      "get_max_shared_memory_per_block_device_attribute(int device_id) -> int");
}

STABLE_TORCH_LIBRARY_IMPL(_C_cuda_utils, CompositeExplicitAutograd,
```

```
                                cuda_utils) {
    cuda_utils.impl("get_device_attribute", TORCH_BOX(&get_device_attribute));
    cuda_utils.impl("get_max_shared_memory_per_block_device_attribute",
                    TORCH_BOX(&get_max_shared_memory_per_block_device_attribute));
}
```

csrc/libtorch_stable/ops.h

声明文件，移除了守卫以支持 ROCm

```
// ... AllSpark ops declarations ...

#endif

// 注：此前 get_cuda_view_from_cpu_tensor 声明被 #ifndef USE_ROCM ... #endif 守卫，
// 仅用于 CUDA 路径。现在移除守卫，对 ROCm 也可见。
// CPU tensor -> CUDA UVA view (shared CUDA/ROCm)
torch::stable::Tensor get_cuda_view_from_cpu_tensor(
    torch::stable::Tensor& cpu_tensor);

// Attention kernels (shared CUDA/ROCm)
void merge_attn_states(
    torch::stable::Tensor& output,
    std::optional<torch::stable::Tensor> output_lse,
    // ... 其他参数
);
```

评论区精华

cleonard530 在 review 中指出 csrc/torch_bindings.cpp 中的 TORCH_LIBRARY_EXPAND 块已没有注册内容，建议删除整个作用域。作者 tjtanaa 采纳意见并在后续提交中清理了该代码块。

风险与影响

- 构建风险：若 stable 扩展中的新文件编译失败会阻塞构建，但已有测试覆盖。
- 兼容性风险：统一 TORCH_TARGET_VERSION 为 0x020B 要求 PyTorch >= 2.11，旧版本 ROCm 环境无法使用。
- 影响范围：仅涉及构建系统和底层 ABI 注册，对模型行为和运行时无影响。团队只需要维护一套 stable ABI 路径，降低了长期维护成本。

关联脉络

此 PR 是 PR#45362（ROCm upgrade to torch 2.11）的后续，清理了 PR#44648 引入的 fallback 机制，最终实现了 ROCm 与 CUDA 在 ABI 层级的统一。