PR #43717 完整报告

vllm-project/vllm

[9/n] Migrate attention and cache kernels to torch stable ABI (continued)

合并时间：2026-05-29 12:44

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/43717

执行摘要

此 PR 是 vLLM 向 libtorch stable ABI 迁移的第九阶段，成功将 paged_attention_v1/v2 以及全部 12 个缓存操作从非稳定的 _C 扩展迁移到 _C_stable_libtorch。迁移后 _C.abi3.so 中不稳定操作计数下降 6 个，稳定库覆盖增加 1 个 shim，进一步减少了库对 PyTorch 内部 API 的依赖。review 中修复了一处因 dispatch 类型放宽导致的安全隐患，并确立了后续头文件迁移的方向。

功能与动机

此 PR 属于将 vLLM 迁移到 libtorch stable ABI 的持续工作（延续 #38871），目标是减少对 PyTorch 内部不稳定 API 的依赖，确保库在不同 PyTorch 版本间的二进制兼容性。具体迁移了 attention（paged_attention_v1/v2）和 cache（swap_blocks、reshape_and_cache、concat_and_cache_mla 等共 12 个操作）的核心内核。

实现拆解

1. 迁移注意力内核：将 csrc/attention/ 下的 paged_attention_v1.cu、paged_attention_v2.cu、attention_kernels.cuh 移动到 csrc/libtorch_stable/attention/，更新 API 调用为 torch stable 版本（如 torch::stable::Tensor、torch::stable::DeviceGuard、get_current_cuda_stream()）。在 csrc/libtorch_stable/ops.h 中新增 paged_attention_v1/v2 的稳定声明。
2. 迁移缓存内核：将 csrc/ 下的 cache_kernels.cu、cache_kernels_fused.cu、nvfp4_kv_cache_kernels.cu 移动到 csrc/libtorch_stable/，同样适配稳定 API，并在 csrc/libtorch_stable/ops.h 中添加 12 个缓存操作的声明。
3. 注册稳定操作：在 csrc/libtorch_stable/torch_bindings.cpp 中使用 STABLE_TORCH_LIBRARY_FRAGMENT 定义操作 schema，使用 STABLE_TORCH_LIBRARY_IMPL 注册 CUDA 实现。从 csrc/torch_bindings.cpp 中删除旧的 TORCH_LIBRARY_EXPAND 注册块，并移除 #include "cache.h" 改为 <torch/all.h>。
4. 清理非稳定声明：从 csrc/ops.h 中删除 paged_attention_v1/v2 的旧声明。在 csrc/quantization/w8a8/fp8/nvidia/quant_utils.cuh 和 amd/quant_utils.cuh 中添加 <torch/headeronly/core/ScalarType.h> 包含，使 DISPATCH_BY_KV_CACHE_DTYPE 宏能在稳定代码中使用。
5. 调整头文件路径：更新移动后文件的 #include 路径，并依据 review 建议将相关稳定专用的头文件（如 attention_kernels.cuh）也移入 libtorch_stable/attention/。

测试方面，运行了现有的 tests/kernels/attention/test_attention.py 和 tests/kernels/attention/test_cache.py 以保证功能正确。

csrc/libtorch_stable/torch_bindings.cpp

在稳定扩展中添加 attention 和 cache 操作的 schema 定义与实现注册，是迁移的核心文件之一。

```
// csrc/libtorch_stable/torch_bindings.cpp 中新增的稳定注册部分

// 1. Attention 操作的 schema 定义（在之前的片段之后）
STABLE_TORCH_LIBRARY_FRAGMENT(_C, ops) {
  ops.def(
      "paged_attention_v1("
      "    Tensor! out, Tensor query, Tensor key_cache,"
      "    Tensor value_cache, int num_kv_heads, float scale,"
      "    Tensor block_tables, Tensor seq_lens, int block_size,"
      "    int max_seq_len, Tensor? alibi_slopes,"
      "    str kv_cache_dtype, Tensor k_scale, Tensor v_scale,"
      "    int tp_rank, int blocksparse_local_blocks,"
      "    int blocksparse_vert_stride, int blocksparse_block_size,"
      "    int blocksparse_head_sliding_step) -> ()");

  ops.def(
      "paged_attention_v2("
      "    Tensor! out, Tensor! exp_sums, Tensor! max_logits,"
      "    Tensor! tmp_out, Tensor query, Tensor key_cache,"
      "    Tensor value_cache, int num_kv_heads, float scale,"
      "    Tensor block_tables, Tensor seq_lens, int block_size,"
      "    int max_seq_len, Tensor? alibi_slopes,"
      "    str kv_cache_dtype, Tensor k_scale, Tensor v_scale,"
      "    int tp_rank, int blocksparse_local_blocks,"
      "    int blocksparse_vert_stride, int blocksparse_block_size,"
      "    int blocksparse_head_sliding_step) -> ()");
}

// 2. CUDA 实现注册（在已有 CUDA impl 块之后）
STABLE_TORCH_LIBRARY_IMPL(_C, CUDA, ops) {
  // ... 已有 impl (permute_cols, 量化 , GGML 等 ) ...

  ops.impl("paged_attention_v1", TORCH_BOX(&paged_attention_v1));
  ops.impl("paged_attention_v2", TORCH_BOX(&paged_attention_v2));
}

// 3. Cache ops 的完整稳定库定义
STABLE_TORCH_LIBRARY_FRAGMENT(_C_cache_ops, ops) {
  ops.def("swap_blocks(Tensor src, Tensor! dst, int block_size_in_bytes, Tensor block_mapping) -
  > ()");
  ops.def("swap_blocks_batch(Tensor src_ptrs, Tensor dst_ptrs, Tensor sizes, bool is_src_
  access_order_any=False) -> ()");
```

```
  ops.def("reshape_and_cache(Tensor key, Tensor value, Tensor! key_cache, Tensor! value_
  cache, Tensor slot_mapping, str kv_cache_dtype, Tensor k_scale, Tensor v_scale) -> ()");
  ops.def("reshape_and_cache_flash(...) -> ()");
  ops.def("concat_and_cache_mla(...) -> ()");
  ops.def("concat_and_cache_mla_rope_fused(...) -> ()");
  // ... 其他缓存操作 ...
}
```

csrc/libtorch_stable/ops.h

在此头文件中添加了 attention 和 cache 操作的稳定 ABI 函数声明，是稳定扩展的接口定义。

```
// csrc/libtorch_stable/ops.h 中在原有声明末尾新增的内容

// PagedAttention 稳定声明
void paged_attention_v1(
    torch::stable::Tensor& out, torch::stable::Tensor& query,
    torch::stable::Tensor& key_cache, torch::stable::Tensor& value_cache,
    int64_t num_kv_heads, double scale, torch::stable::Tensor& block_tables,
    torch::stable::Tensor& seq_lens, int64_t block_size, int64_t max_seq_len,
    const std::optional<torch::stable::Tensor>& alibi_slopes,
    const std::string& kv_cache_dtype, torch::stable::Tensor& k_scale,
    torch::stable::Tensor& v_scale, const int64_t tp_rank,
    const int64_t blocksparse_local_blocks,
    const int64_t blocksparse_vert_stride, const int64_t blocksparse_block_size,
    const int64_t blocksparse_head_sliding_step);

void paged_attention_v2(
    torch::stable::Tensor& out, torch::stable::Tensor& exp_sums,
    torch::stable::Tensor& max_logits, torch::stable::Tensor& tmp_out,
    torch::stable::Tensor& query, torch::stable::Tensor& key_cache,
    torch::stable::Tensor& value_cache, int64_t num_kv_heads, double scale,
    torch::stable::Tensor& block_tables, torch::stable::Tensor& seq_lens,
    int64_t block_size, int64_t max_seq_len,
    const std::optional<torch::stable::Tensor>& alibi_slopes,
    const std::string& kv_cache_dtype, torch::stable::Tensor& k_scale,
    torch::stable::Tensor& v_scale, const int64_t tp_rank,
    const int64_t blocksparse_local_blocks,
    const int64_t blocksparse_vert_stride, const int64_t blocksparse_block_size,
    const int64_t blocksparse_head_sliding_step);

// Cache 操作稳定声明
void swap_blocks(torch::stable::Tensor& src, torch::stable::Tensor& dst,
                 int64_t block_size_in_bytes,
                 const torch::stable::Tensor& block_mapping);
void reshape_and_cache(torch::stable::Tensor& key, torch::stable::Tensor& value,
                       torch::stable::Tensor& key_cache,
                       torch::stable::Tensor& value_cache,
                       torch::stable::Tensor& slot_mapping,
                       const std::string& kv_cache_dtype,
```

```
                    torch::stable::Tensor& k_scale,
                    torch::stable::Tensor& v_scale);
// 其他类似声明省略
```

评论区精华

> depthfirst-app[bot]：concat_mla_q 的 dispatch 从 HALF_TYPES 改成 FLOATING_TYPES 会导致 Half 输入时输出损坏，应使用 HALF_TYPES。 cleonard530：已修复，改回 HALF_TYPES 并添加类型检查。 Harry-Chen：建议将稳定专用的头文件也移入 libtorch_stable，并考虑更激进的整体迁移。 cleonard530：本 PR 移动了部分，后续在 #44013 继续讨论。 janeyx99：quant_utils.cuh 是否已经完全 stable? cleonard530：尚未完全，因为它使用 c10::Float8_* 而非 stable 头。 Harry-Chen：nvfp4_kv_cache_kernels.cu 也应移入 libtorch_stable。 cleonard530：已移动。

风险与影响

风险

- 类型调度风险：concat_mla_q 的 dispatch 类型已修复，但类似模式可能存在于其他迁移代码中。
- 未完全迁移的头文件：quant_utils.cuh 仍包含不稳定依赖，被 stable 代码间接引入会破坏 ABI 稳定性。
- 包含路径错误：移动后使用 ../ 相对路径，未来目录重构可能导致编译失败。
- 平台专用代码：nvfp4_kv_cache 仅 Blackwell 支持，条件编译需要验证。
- 测试覆盖不足：仅有注意力测试，缓存操作缺乏独立新增测试。

影响

- 用户：无功能变化，但库在不同 PyTorch 版本间更稳定。
- 系统：_C.abi3.so 不稳定操作从 99 降至 93。
- 团队：本次迁移确立了可复用的模式，降低后续迁移成本。

关联脉络

- 此 PR 是 [#38871] 的延续（phase 9），继续之前未完成的注意力与缓存内核迁移。
- review 中讨论了更彻底的头文件迁移方案，已在 #44013 中继续。
- 完整的 ABI 迁移计划将逐步覆盖所有 CUDA 内核，最终实现 _C.abi3.so 零不稳定操作。