PR #41516 完整报告

vllm-project/vllm

[Build] Build bundled DeepGEMM `_C` per-Python so the wheel imports on every CPython

合并时间：2026-05-12 22:27

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41516

执行摘要

- 一句话：按 Python 版本构建 DeepGEMM _C 扩展并打包进 wheel，支持多 Python 导入
- 推荐动作：该 PR 解决了关键的跨 Python 兼容性问题，并建立了可维护的构建体系。值得合入 v0.21 版本。建议后续关注 CMake 头文件依赖扫描的改进，并考虑推动上游或使用 nanobind 简化构建。

功能与动机

修复 issue #41476 和 #41512 描述的回归：DeepGEMM 的 _C 是 pybind11 模块，编译产生的 .so 文件只适用于构建时的 Python 版本，而之前 wheel 只包含一个 .so 文件，导致用户在非构建 Python 版本上安装 vLLM 时无法导入 DeepGEMM。本 PR 采用按 Python 版本分别编译并打包的方式规避了这一限制。

实现拆解

1. 新增 tools/build_deepgemm_C.py：针对一个目标 Python 版本编译 _C.so。利用构建环境的 torch，目标 Python 只需提供头文件和 SOABI，无需安装 torch。
2. 新增 tools/setup_deepgemm_pythons.sh：基于 pyproject.toml 中的 requires-python 字段自动推导需要支持的 Python 版本列表，利用 uv 创建裸的虚拟环境（仅包含 Python 解释器，不安装 torch）。
3. 修改 cmake/external_projects/deepgemm.cmake：将原先使用 Python_add_library 的单一构建方式改为基于 DEEPGEMM_PYTHON_INTERPRETERS 循环的 add_custom_command，对每个 Python 调用 build_deepgemm_C.py 生成对应的 .so，并通过 cmake --install 打包到 vllm/third_party/deep_gemm/。
4. 修改 docker/Dockerfile：在 wheel 构建阶段调用 setup_deepgemm_pythons.sh 生成解释器列表，并通过环境变量传入 CMake。
5. 新增 tools/check_wheel_deepgemm.py：在 CI 的 H100 DeepGEMM 测试步骤中验证 wheel 是否包含所有必需 Python 版本的 .so，若缺失则报错退出。
6. 修改 .buildkite/test_areas/kernels.yaml：在 H100 DeepGEMM 测试步骤中添加 wheel 验证命令，并更新源码依赖列表以包含新脚本。

关键文件：

- tools/build_deepgemm_C.py（模块 编译脚本；类别 source；类型 core-logic）：新增的编译脚本，负责针对单个目标 Python 版本编译 _C.so，是整个多 Python 构建的核心。

- tools/check_wheel_deepgemm.py（模块 验证脚本；类别 source；类型 core-logic；符号 required_pythons）：新增的 wheel 验证脚本，在 CI 中确保 wheel 包含所有必需 Python 版本的 .so，避免回归。
- cmake/external_projects/deepgemm.cmake（模块 CMake 配置；类别 config；类型 core-logic）：CMake 构建文件重构，由单一 Python_add_library 改为 per-Python add_custom_command 循环，是多 Python 构建的关键基础设施。
- tools/setup_deepgemm_pythons.sh（模块 环境准备；类别 other；类型 core-logic）：新增的环境准备脚本，利用 uv 快速创建裸 Python 虚拟环境（不含 torch），自动推导版本矩阵。
- docker/Dockerfile（模块 Docker 配置；类别 infra；类型 infrastructure）：Docker 构建环境修改，在 wheel 构建阶段调用 setup_deepgemm_pythons.sh 并通过环境变量传递解释器列表。
- .buildkite/test_areas/kernels.yaml（模块 CI 配置；类别 config；类型 configuration）：CI 配置更新，在 H100 DeepGEMM 测试步骤中添加 wheel 验证命令，并更新文件依赖列表。

关键符号：required_pythons

关键源码片段

tools/build_deepgemm_C.py

新增的编译脚本，负责针对单个目标 Python 版本编译 _C.so，是整个多 Python 构建的核心。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
"""Build DeepGEMM's `_C` pybind11 extension for a target Python.

Driven from `cmake/external_projects/deepgemm.cmake`. The driver is the
build interpreter (which has torch); the *target* Python is only used for
its header path and SOABI. This avoids needing torch installed in N venvs
to produce N matching `.so` files.

Usage: python build_deepgemm_C.py <DEEPGEMM_SRC_DIR> <OUTPUT_DIR> <TARGET_PY>
"""

import json
import os
import subprocess
import sys
from pathlib import Path

import torch
from torch.utils import cpp_extension

# 校验参数数量
if len(sys.argv) != 4:
    sys.exit(f"usage: {sys.argv[0]} <SRC> <OUT> <TARGET_PY>")
```

```
src = Path(sys.argv[1]).resolve()
out = Path(sys.argv[2]).resolve()
target_py = sys.argv[3]
out.mkdir(parents=True, exist_ok=True)

# 查询目标 Python 的 EXT_SUFFIX 和 INCLUDEPY，仅需解释器，无需 torch
info = json.loads(
    subprocess.check_output(
        [
            target_py,
            "-c",
            "import sysconfig, json; "
            "print(json.dumps({k: sysconfig.get_config_var(k) "
            "for k in ('EXT_SUFFIX', 'INCLUDEPY')}))",
        ]
    ).decode()
)

cuda_home = cpp_extension.CUDA_HOME
if cuda_home is None:
    sys.exit("CUDA_HOME not found; cannot build DeepGEMM _C")

# 构造头文件包含路径，包含 CCCL 等 DeepGEMM 依赖
includes = [
    info["INCLUDEPY"],
    f"{cuda_home}/include",
    f"{cuda_home}/include/cccl",
    str(src / "csrc"),
    str(src / "deep_gemm/include"),
    str(src / "third-party/cutlass/include"),
    str(src / "third-party/cutlass/tools/util/include"),
    str(src / "third-party/fmt/include"),
    *cpp_extension.include_paths(device_type="cuda"),
]

# 使用 $CXX 或默认 g++ 编译，生成带目标 Python SOABI 的 .so 文件
cmd = [
    os.environ.get("CXX", "g++"),
    "-shared",
    "-fPIC",
    "-std=c++20",
    "-O3",
    "-g0",
    "-Wno-psabi",
    "-Wno-deprecated-declarations",
    "-DTORCH_API_INCLUDE_EXTENSION_H",
    "-DTORCH_EXTENSION_NAME=_C",
    f"-D_GLIBCXX_USE_CXX11_ABI={int(torch.compiled_with_cxx11_abi())}",
    *(f"-I{p}" for p in includes),
```

```
        str(src / "csrc/python_api.cpp"),
        *(f"-L{p}" for p in cpp_extension.library_paths(device_type="cuda")),
        f"-L{cuda_home}/lib64",
        "-ltorch",
        "-ltorch_python",
        "-ltorch_cpu",
        "-ltorch_cuda",
        "-lc10",
        "-lc10_cuda",
        "-lcudart",
        "-lnvrtc",
        "-o",
        str(out / f"_C{info['EXT_SUFFIX']}"),
    ]
    print("[build_deepgemm_C] " + " ".join(cmd), flush=True)
    subprocess.check_call(cmd)
```

tools/check_wheel_deepgemm.py

新增的 wheel 验证脚本，在 CI 中确保 wheel 包含所有必需 Python 版本的 .so，避免回归。

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

    """Assert the installed vLLM has a `_C.cpython-X.Y-*.so` for every CPython
    covered by `requires-python`. Fails closed if a Python's `.so` is missing
    from the wheel — i.e. the regression that surfaced in #41476/#41512.

    Run from a CI test job after vLLM is installed, e.g. the H100 deepgemm
    kernel tests in .buildkite/test_areas/kernels.yaml.
    """

    import importlib.util
    import os
    import sys
    from pathlib import Path

    import regex as re
    import tomllib

    SO_RE = re.compile(r"^_C\.cpython-(\d)(\d+)-")

    # 从 pyproject.toml 中读取 requires-python 字段，动态推导需要支持的 Python 版本列表
    def required_pythons() -> list[str]:
        pyproject = Path(__file__).resolve().parent.parent / "pyproject.toml"
        spec = tomllib.loads(pyproject.read_text())["project"]["requires-python"]
        m = re.match(r">=3\.(\d+),<3\.(\d+)", spec)
        if not m:
            sys.exit(f"unexpected requires-python format: {spec!r}")
        return [f"3.{v}" for v in range(int(m[1]), int(m[2]))]
```

```
# 定位 vllm.third_party.deep_gemm 包目录，通过 importlib 避免触发 deep_gemm 的 import
spec = importlib.util.find_spec("vllm.third_party.deep_gemm")
if spec is None or spec.origin is None:
    sys.exit("vllm.third_party.deep_gemm not importable; is vllm installed?")
pkg_dir = Path(spec.origin).parent

# 找到目录下所有 _C.cpython-X.Y-*.so 文件，提取 Python 版本号
found = {f"{m[1]}.{m[2]}" for f in os.listdir(pkg_dir) if (m := SO_RE.match(f))}
required = required_pythons()
missing = [v for v in required if v not in found]
print(f"deepgemm _C: found {sorted(found)}, required {required}, missing {missing}")
sys.exit(1 if missing else 0)
```

评论区精华

核心讨论包括：

- wheel 大小：ZJY0516 询问 wheel 膨胀程度，mgoin 回应每个 .so 约 1.4 MB，总计约 6 MB，认为可接受。
- nanobind 替代：Harry-Chen 建议迁移到 nanobind（支持 Python 稳定 API）以避免版本相关问题，并创建上游 issue。mgoin 表示上游不接受 PR，且需要维护 fork，当前方案足够。
- 编译时间：Harry-Chen 担心多次编译的时间开销，mgoin 解释每次编译仅需一分钟，因为核心 kernel 是 JIT 的。
- 空环境变量处理：*claude[bot]* 指出 CMake 的 if(DEFINED ENV{...}) 对空字符串为真，可能导致跳过构建。后续提交修复为 if(NOT "$ENV{...}" STREQUAL "")。
- python vs python3：*claude[bot]* 发现 CI 镜像中只有 python3，后修复为 python3。
- 头文件依赖跟踪：*claude[bot]* 指出 add_custom_command 不隐式扫描头文件，头文件变更不会触发重编。作者认为 CI 通常洁净构建，影响有限。
  - wheel 尺寸影响 (performance): 接受约 6 MB 的尺寸增加，认为可接受。
  - 使用 nanobind 替代 pybind11 (design): 暂不切换，维持 pybind11 的 per-Python 构建方案。
  - 编译时间顾虑 (performance): 编译时间可接受（4 次，各约 1 分钟）。
  - CMake 空环境变量处理 (correctness): 修复为 if(NOT "$ENV{DEEPGEMM_PYTHON_INTERPRETERS}" STREQUAL "")，确保空字符串时回退到构建解释器。
  - CI 中 python vs python3 (testing): 将命令改为 python3 ./tools/check_wheel_deepgemm.py。

风险与影响

- 风险：
  1. wheel 大小增加约 6 MB，但可接受。
  2. 编译时间增加约 4-5 分钟，CI 中可接受，但可能影响本地构建测试。

3. CMake 增量构建时，头文件变更不触发 _C.so 重编（add_custom_command 不自动扫描头文件）。CI 通常洁净构建，但本地开发需注意。
4. 版本矩阵同步：pyproject.toml 中的 requires-python 变更需与脚本同步，但已通过自动推导缓解。
5. 新增 CI 验证步骤，若验证失败会阻止发布，保障质量。 - 影响：用户：vLLM wheel 现可在多个 Python 版本上正确导入 DeepGEMM，之前因 .so 缺失导致的导入错误已修复。构建系统：构建过程更复杂，需额外依赖 uv 用于创建 venv，但运行时无额外依赖。需要维护额外的构建脚本和 CI 步骤。团队：需要理解并维护多 Python 构建逻辑，但已有自动推导减少手动维护。与上游 DeepGEMM 的集成方式仍存在改进空间（如迁移到 nanobind）。

- 风险标记：wheel 尺寸增加 6 MB，增量编译头文件变更不触发生成，新增约 4 次编译，自动推导版本矩阵

关联脉络

- PR #41476 [Bugfix] Use fake-abi3 trick for DeepGEMM _C cross-version: 本 PR 之前尝试的方案，试图使用 fake-abi3 技巧使单个 .so 跨版本工作，但因 pybind11 不支持而被 revert。
- PR #41512 [Revert] Revert #41476: DeepGEMM _C fake-abi3: 将 #41476 的变更回退，导致回归发生，从而促使本 PR 采取 per-Python 构建方案。