PR #24348 完整报告

sgl-project/sglang

Add release workflow for sgl-deep-gemm wheels

合并时间：2026-05-04 15:58

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24348

执行摘要

- 一句话：新增 sgl-deep-gemm wheel 发布工作流
- 推荐动作：建议精读：对于负责 CI/CD 或 wheel 发布的成员，本 PR 提供了完整的跨架构、跨 CUDA 版本的 wheel 发布范例。特别值得关注的设计决策包括：
  - 利用 local version tag 区分 cu129/cu130，并在上传 PyPI 时剥离，符合 PyPI 规范。
  - 通过 pip wheel unpack/repack 修改平台 tag，而非重新编译，确保与 manylinux 兼容。
  - 建立双发布渠道（PyPI + GH Release），平衡便利性和版本管理。 对于其他 Python 绑定的发布流程可参考此模式。

功能与动机

此前 sgl-deep-gemm 缺少自动化发布流程，依赖手动构建上传。本 PR 通过标准化 workflow 降低发布出错概率，确保 cu129/cu130 双版本、双架构的 wheel 分发一致性。参考 PR body：'Add .github/workflows/release-whl-deepgemm.yml to build and publish sgl-deep-gemm wheels.'

实现拆解

1. Workflow 定义(.github/workflows/release-whl-deepgemm.yml)：通过 workflow_dispatch 接收 version、target（all/cu129/cu130）、branch 参数。构建矩阵为 [cu129, cu130] × [x86_64, aarch64]，使用自托管 runner（x64-kernel-build-node / arm-kernel-build-node）。并发控制确保同一时间只有一个运行。
2. Docker 构建环境(docker/sgl-deep-gemm.Dockerfile)：基于 pytorch/manylinux2_28-builder 或 pytorch/manylinuxaarch64-builder，安装指定版本的 torch（2.11.0）和 tvm-ffi（0.1.9），并创建 libcuda.so 符号链接以解决链接问题。
3. 构建脚本(scripts/build_sgl_deep_gemm.sh)：参数化 Python 版本、CUDA 版本、DeepGEMM 源码路径和架构。动态选择基础镜像标签，在 Docker 内执行构建，输出默认 py3-none-any 的 wheel。
4. 重命名脚本(scripts/rename_sgl_deep_gemm_whl.sh)：通过 pip wheel unpack 解包 wheel，修改 METADATA 中的 Version（追加 +cu<ver>）和 WHEEL 中的 Tag（改为 py3-none-manylinux2014_<arch>），再重新打包，满足多平台兼容性要求。
5. 索引更新脚本(scripts/update_deepgemm_whl_index.py)：为 cu129 版本生成 PEP 503 简单索引 HTML，包含 SHA256 哈希，托管在 sgl-project/whl 的 Release 中，用户可通

过 --extra-index-url https://sgl-project.github.io/whl/cu129/ 安装。

6. 发布分离策略：cu130 wheels 去除 +cu130 local version tag 后上传至 PyPI（使用 SGL_DEEP_GEMM_PYPI_TOKEN）；cu129 wheels 直接作为 GitHub Release 发布到 sgl-project/whl，并更新索引页面。此设计复用已有的 release-whl-kernel.yml 的发布基础设施（runner、token 等）。

关键文件：

- scripts/update_deepgemm_whl_index.py（模块 索引生成；类别 source；类型 core-logic；符号 update_wheel_index, main）：核心索引生成逻辑，是 cu129 wheel 分发的关键组件，决定了用户能否通过 --extra-index-url 安装。
- .github/workflows/release-whl-deepgemm.yml（模块 发布流水线；类别 infra；类型 infrastructure）：工作流入口，定义了构建矩阵、发布策略和所有步骤，是整个发布自动化的编排文件。
- scripts/build_sgl_deep_gemm.sh（模块 构建脚本；类别 other；类型 core-logic）：实际在 Docker 容器中执行构建的脚本，封装了环境配置和构建命令。
- scripts/rename_sgl_deep_gemm_whl.sh（模块 打包处理；类别 other；类型 core-logic）：处理 wheel 平台 tag 和版本名，确保与 manylinux 和 CUDA 版本兼容。
- docker/sgl-deep-gemm.Dockerfile（模块 构建环境；类别 infra；类型 infrastructure）：定义构建环境，包括基础镜像、CUDA 版本、torch/tvm-ffi 依赖，是构建可复现性的基础。

关键符号：update_wheel_index, main

关键源码片段

scripts/update_deepgemm_whl_index.py

核心索引生成逻辑，是 cu129 wheel 分发的关键组件，决定了用户能否通过 --extra-index-url 安装。

```
# scripts/update_deepgemm_whl_index.py
# 为 sgl-deep-gemm 的 cu129 wheels 生成 PEP 503 简单索引 HTML，
# 用户可通过 --extra-index-url https://sgl-project.github.io/whl/cu129/ 安装。
import argparse
import hashlib
import pathlib
import re

SUPPORTED_CUDA_VERSIONS = ["129", "130"]

def update_wheel_index(cuda_version: str, wheel_dir: str) -> None:
    """遍历 wheel 目录，筛选匹配 +cu<version> 的 .whl 文件，
       计算 SHA256，生成带哈希的索引条目并写入 index.html。"""
    index_dir = pathlib.Path(f"sgl-whl/cu{cuda_version}/sgl-deep-gemm")
    index_dir.mkdir(exist_ok=True, parents=True)
    base_url = "https://github.com/sgl-project/whl/releases/download"

    suffix = f"+cu{cuda_version}"
```

```
    for path in sorted(pathlib.Path(wheel_dir).glob("*.whl")):
        if suffix not in path.name:
            continue # 跳过不匹配当前 CUDA 版本的 wheel
        with open(path, "rb") as f:
            sha256 = hashlib.sha256(f.read()).hexdigest()
        # 提取版本号：sgl_deep_gemm-0.1.0+cu129-py3-none-manylinux2014_x86_64.whl
        match = re.match(r"sgl_deep_gemm-([0-9][^-+]*)(?:\+cu[0-9]+)?-", path.name)
        if not match:
            continue
        ver = match.group(1)
        full_url = f"{base_url}/v{ver}/{path.name}#sha256={sha256}"
        with (index_dir / "index.html").open("a") as f:
            f.write(f'<a href="{full_url}">{path.name}</a><br>\n')

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("--cuda", type=str, required=True, choices=SUPPORTED_CUDA_
    VERSIONS)
    parser.add_argument("--wheel-dir", type=str, default="dist")
    args = parser.parse_args()
    update_wheel_index(args.cuda, args.wheel_dir)

if __name__ == "__main__":
    main()
```

评论区精华

所有 review 评论均由作者 @Fridge003 在合并前提出并解决，主要涉及：

- 文件组织：要求将 Dockerfile 移入 docker/，构建和重命名脚本移入 scripts/，保持仓库结构一致。
- 构建参数化：为 Dockerfile 添加 TORCH_VER 和 TVM_FFI_VER ARG，支持 future 版本覆盖。
- 工作流 UI 友好性：将 workflow 显示名改为 'Release sgl-deep-gemm'，在 target/branch 输入描述中标明默认值，提升 workflow_dispatch 可用性。 无明显未解决争议。
- 文件位置调整 (other)：已完成移动。
- Dockerfile ARG 参数化 (design)：已添加 ARG。
- Workflow UI 友好性 (design)：已修改。

风险与影响

- 风险：
  1. 外部依赖稳定性：构建依赖 sgl-project/DeepGEMM 仓库，若该仓库的分支（release-0426）变更或出现兼容性问题，工作流将失败。

2. 自托管 Runner 环境：使用非标准 self-hosted runner，缺少隔离和环境快照，可能因磁盘空间或残留缓存导致构建失败。
3. 秘密管理：SGL_DEEP_GEMM_PYPI_TOKEN 和 GH_PAT_FOR_WHL_RELEASE 作为明文 secret 使用，若泄露可致供应链投毒。需确认为严格受限的 token。
4. 版本冲突：cu130 wheels 上传至 PyPI 时丢掉 +cu130 版本段，同一 index 上只有一个版本；用户若未指定索引可能意外安装错版。设计上需确保发布时仅推送 cu130 到 PyPI。
5. 无自动化测试：当前工作流仅有手动触发验证计划（见 PR body Test plan），未集成 CI 回归测试，无法在合并前自动验证构建成功。

- 影响：
  - 用户侧：DeepGEMM 用户现在可以通过 pip install sgl-deep-gemm（cu130）或通过 cu129 索引安装 cu129 版本，降低使用门槛。
  - 团队侧：发布流程完全自动化，从手动执行 shell 脚本变为一键 workflow dispatch，减少人为错误。
  - 系统侧：复用已有 runner 和发布仓库（sgl-project/whl），基础设施扩展成本低。 整体影响为中等，主要惠及内部发布者和下游 DeepGEMM 用户。
- 风险标记：外部依赖，自托管 Runner，缺少 CI 测试，PyPI 版本冲突，凭证安全

关联脉络

- 暂无明显关联 PR