PR #37226 完整报告

vllm-project/vllm

[CI] Add PyTorch nightly build and test pipeline

合并时间：2026-04-15 08:13

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/37226

执行摘要

- 一句话：添加 PyTorch nightly 构建与测试流水线，支持 CUDA 13.0 和递归依赖处理。
- 推荐动作：建议 CI 维护者关注此 PR，特别是构建脚本中的安全问题和参数设计，以及依赖处理的递归变更，这些对于确保 nightly 测试的稳定性和安全性至关重要。

功能与动机

根据关联 Issue #310，目标是重新启用 PyTorch nightly 构建和测试流水线，确保 vLLM 与最新 PyTorch 版本的兼容性，及早发现潜在问题。PR body 中提到这是 vllm-project/ci-infra#310 的伴侣 PR，用于添加 vLLM 侧的 Docker 构建脚本和测试配置。

实现拆解

1. 新增 PyTorch nightly 构建脚本：创建 .buildkite/image_build/image_build_torch_nightly.sh，脚本包含 ECR 登录、buildx 设置、镜像存在检查，并使用 CUDA 13.0 基础镜像和特定 torch_cuda_arch_list 构建 Docker 镜像。关键变更：设置 PYTORCH_NIGHTLY=1 和扩展的 CUDA 架构列表。影响：为 nightly 测试提供专用镜像。
2. 修复依赖脚本以递归处理 requirements：修改 use_existing_torch.py 中的 glob 模式，从 requirements/*.txt 改为 requirements/**/*.txt 并启用 recursive=True。原因：此前未处理子目录（如 requirements/test/cuda.in）中的文件，导致 torchvision 版本冲突。影响：确保所有 requirements 文件中的 torch 依赖被正确剥离，兼容 nightly 构建。
3. 更新测试配置以加入 nightly 流水线：在 .buildkite/test_areas/models_language.yaml 和 models_basic.yaml 中，将 torch_nightly: true 移至 mirror: torch_nightly: {}，使得测试步骤在 nightly 流水线中镜像运行。同时升级 ROCm AMD 镜像中的 causal-conv1d 版本从 v1.5.2 到 v1.6.0。影响：基础语言模型和混合模型测试现在会在 PyTorch nightly 环境下执行。
4. 伴随更改：提交历史中包含多个修复提交，如调整 causal-conv1d 版本以确保依赖一致性；无直接测试文件变更，但配置调整影响 CI 执行流程。

关键文件：

- .buildkite/image_build/image_build_torch_nightly.sh（模块 CI 构建脚本；类别 infra；类型 core-logic）：新增构建脚本，核心实现 PyTorch nightly Docker 镜像的构建逻辑，是流水线入口。

- use_existing_torch.py（模块 依赖管理；类别 source；类型 core-logic；符号 main）：修改依赖处理逻辑，递归查找 requirements 子目录文件，确保所有 torch 依赖被剥离以兼容 nightly 构建。
- .buildkite/test_areas/models_language.yaml（模块 测试配置；类别 config；类型 configuration）：更新测试配置，将语言模型测试步骤标记为 torch_nightly 镜像运行，纳入 nightly 流水线。
- .buildkite/test_areas/models_basic.yaml（模块 测试配置；类别 config；类型 configuration）：更新测试配置，将基础模型测试步骤标记为 torch_nightly 镜像运行，纳入 nightly 流水线。

关键符号：main

关键源码片段

.buildkite/image_build/image_build_torch_nightly.sh

新增构建脚本，核心实现 PyTorch nightly Docker 镜像的构建逻辑，是流水线入口。

```
#!/bin/bash
set -euo pipefail

# 构建vLLM测试镜像，安装PyTorch nightly版本。
# 由流水线生成器的"vLLM Against PyTorch Nightly"组调用。

if [[ $# -lt 5 ]]; then
  echo "Usage: $0 <registry> <repo> <commit> <branch> <image_tag>"
  exit 1
fi

REGISTRY=$1
REPO=$2 # 注意：此参数在脚本中未使用，可能存在设计瑕疵
BUILDKITE_COMMIT=$3
BRANCH=$4 # 注意：此参数在脚本中未使用，可能存在设计瑕疵
IMAGE_TAG=$5

# ECR登录
echo "--- :key: ECR login"
aws ecr-public get-login-password --region us-east-1 \
  | docker login --username AWS --password-stdin "$REGISTRY"
aws ecr get-login-password --region us-east-1 \
  | docker login --username AWS --password-stdin 936637512419.dkr.ecr.us-east-1.amazonaws.
  com # 安全风险：硬编码AWS账户ID

# 设置buildx
echo "--- :docker: Setting up buildx"
docker buildx create --name vllm-builder --driver docker-container --use || true
docker buildx inspect --bootstrap
docker buildx ls
```

```
# 检查镜像是否已存在
echo "--- :mag: Checking if image already exists"
if docker manifest inspect "$IMAGE_TAG" >/dev/null 2>&1; then
  echo "Image found: $IMAGE_TAG — skipping build"
  exit 0
fi
echo "Image not found, proceeding with build..."

# Nightly构建使用CUDA 13.0，而常规CI使用CUDA 12.9
NIGHTLY_CUDA_VERSION="13.0.0"
NIGHTLY_BUILD_BASE_IMAGE="nvidia/cuda:${NIGHTLY_CUDA_VERSION}-devel-ubuntu22.04"
NIGHTLY_FINAL_BASE_IMAGE="nvidia/cuda:${NIGHTLY_CUDA_VERSION}-base-ubuntu22.04"

echo "--- :docker: Building torch nightly image (CUDA ${NIGHTLY_CUDA_VERSION})"
docker buildx build --file docker/Dockerfile \
  --build-arg max_jobs=16 \
  --build-arg buildkite_commit="$BUILDKITE_COMMIT" \
  --build-arg USE_SCCACHE=1 \
  --build-arg PYTORCH_NIGHTLY=1 \
  --build-arg CUDA_VERSION="${NIGHTLY_CUDA_VERSION}" \
  --build-arg BUILD_BASE_IMAGE="${NIGHTLY_BUILD_BASE_IMAGE}" \
  --build-arg FINAL_BASE_IMAGE="${NIGHTLY_FINAL_BASE_IMAGE}" \
  --build-arg torch_cuda_arch_list="8.0 8.9 9.0 10.0 12.0" \  #
  注意：在review中提到了大小写问题，这里已修正为小写
  --tag "$IMAGE_TAG" \
  --push \
  --target test \
  --progress plain .

echo "--- :white_check_mark: Torch nightly image build complete: $IMAGE_TAG"
```

use_existing_torch.py

修改依赖处理逻辑，递归查找 requirements 子目录文件，确保所有 torch 依赖被剥离以兼容 nightly 构建。

```
def main(argv):
    parser = argparse.ArgumentParser(
        description="Strip torch lib requirements to use installed version."
    )
    parser.add_argument(
        "--prefix",
        action="store_true",
        help="Strip prefix matches only (default: False)",
    )
    args = parser.parse_args(argv)

    # 使用递归glob模式查找requirements目录下的所有.txt和.in文件，包括子目录
    for file in (
        *glob.glob("requirements/**/*.txt", recursive=True),
```

```
        *glob.glob("requirements/**/*.in", recursive=True),
        "pyproject.toml",
    ):
        with open(file) as f:
            lines = f.readlines()
        if "torch" in "".join(lines).lower():
            with open(file, "w") as f:
                for line in lines:
                    # 根据参数决定是否仅匹配前缀或包含"torch"字符串
                    if (
                        args.prefix
                        and not line.lower().strip().startswith(TORCH_LIB_PREFIXES)
                        or not args.prefix
                        and "torch" not in line.lower()
                    ):
                        f.write(line)
                    else:
                        print(f">>> removed from {file}:", line.strip())
```

评论区精华

- 安全风险：gemini-code-assist[bot] 指出构建脚本中硬编码了 AWS 账户 ID（936637512419.dkr.ecr.us-east-1.amazonaws.com），存在安全风险，建议参数化使用环境变量。
- 构建参数问题：同一评论者提到 REPO 和 BRANCH 参数未使用，TORCH_CUDA_ARCH_LIST 大小写不匹配应改为 torch_cuda_arch_list，且 FI_TORCH_CUDA_ARCH_LIST 未在 Dockerfile 中定义。
- 决策与状态：评论中提供了建议，但 PR 最终被合并，可能这些问题被接受或暂未解决。从 commit 历史看，最后一个提交专注于修复 use_existing_torch.py，而构建脚本的问题未在提交中显示是否处理。
  - 硬编码 AWS 账户 ID 安全风险 (security): 未在提交中显示是否解决，PR 被合并可能接受风险或暂缓处理。
  - 未使用脚本参数和构建参数问题 (design): 可能未在本次 PR 中解决，但 PR 仍被合并。

风险与影响

- 风险：- 安全风险：硬编码 AWS 账户 ID 可能暴露内部基础设施，增加安全威胁。
- 构建失败风险：构建参数不匹配（如 TORCH_CUDA_ARCH_LIST vs torch_cuda_arch_list）可能导致镜像构建使用默认值，而非预期的 CUDA 架构，影响性能或兼容性。
- 维护复杂性：新增脚本和配置增加了 CI 流水线的维护负担，尤其是 nightly 构建可能因 PyTorch 不稳定而频繁失败。
- 依赖冲突风险：尽管修复了递归 glob，但若其他子目录中有未处理的依赖文件，仍可能导致版本冲突。
- 影响：- 对用户：无直接用户影响，这是内部 CI 改进。

- 对系统：扩展了测试矩阵，使用 PyTorch nightly 可以提前检测与未来 PyTorch 版本的兼容性问题，提升软件质量。
- 对团队：开发团队能更早获得反馈，但需要处理 nightly 构建可能的失败，增加了 CI 运维工作量。
- 风险标记：硬编码敏感信息，构建参数不匹配，未使用参数

关联脉络

- PR #310 [CI] Re-enable PyTorch nightly build and test in nightly CI/CD: 伴侣 PR，在 ci-infra 仓库中定义流水线生成逻辑，与此 PR 协同工作以启用 nightly 测试。