

PR #38404 完整报告

sgl-project/sglang

[Deps] Retire the CUDA 12 lane

合并时间: 2026-09-11 07:58

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38404>

执行摘要

- 一句话: 退役 CUDA 12 构建发布通道, 为 PyTorch 2.14 铺路
- 推荐动作: 值得精读, 尤其对负责发布流水线、镜像与依赖管理的工程师: 这是一次「删除为主」的 PR, 真正的决策密度集中在少数几处。建议重点关注三点: 一是 `scripts/update_kernel_whl_index.py` 把 CUDA 版本白名单替换为 `+cuNNN` 正则匹配——这是 fail-closed 的稳健性提升, 杜绝了未来新 CUDA 版本的 wheel 误入 cu130 索引, 值得作为同类脚本的范式; 二是 `_docker-build-and-publish.yml` 中 `manifest` 固定使用 cu130 digest 后 `tag_config.cuda` 字段被静默忽略, 建议合并后排查所有调用方并显式拒绝 (或警告) 非 cu130 的 `cuda` 取值; 三是给 `check_wheel_cuda_version` 补一组针对 `+cu129 / +cu130 / 无标记 / +rocm720` 的单测。对 CUDA 12 用户, 应在 release notes 中明确「v0.5.19 为最后一个 CUDA 12 版本」以及升级到 CUDA 13 的路径, 避免他们从新的报错提示里得不到有效指引。

功能与动机

PR body 给出的动机非常直接: 「Drops the CUDA 12.9 wheels and images, per the advance notice — prerequisite for the PyTorch 2.14 upgrade. Already-published `-cu129 / -cu12` image tags are unaffected. v0.5.19 is the last release with a CUDA 12 lane.」即: 其一, PyTorch 2.14 不再为退役 cu12 通道使用的 cu129 索引发布 CUDA 12 wheel (`ci_install_dependency.sh` 的注释原文: "PyTorch 2.14 publishes no CUDA 12 wheels for the cu129 index the retired cu12 lane used"), 继续保留 cu12 通道会直接卡住升级; 其二, 同时维护 cu129/cu130 两套镜像、wheel 与 CI 分支的边际成本已高于收益; 其三, 该动作已按内部 Slack 公告 (body 中的 advance notice 链接, 非公开可访问) 提前通知用户。关联 Issue 为空, 说明这是一次由维护团队主动发起的通道退役, 而非外部缺陷驱动。

实现拆解

1. 移除 CUDA 12 的发布矩阵 (workflows 层): `release-whl-kernel.yml` 删掉 `build-cu129-matrix` 与 `release-cu129` 两个 job (-105 行) 并从 `target` 选项中去掉 `cu129`; `release-whl-deepep.yml` 删除 `build-cu129-matrix`、删除整个 `target` 输入, 并把 `release-whl / release-pypi` 的 `needs` 与 `if` 条件从「按 target 分支判断两个矩阵」简化为只依赖 `build-cu130-matrix.result == 'success'`; `release-whl-deepgemm.yml` 同样删除 `eu129` 矩阵与 `release-cu129`。结果是新增 wheel 只产出 CUDA 13 一条通道。

2. 镜像构建与发布收敛: `_docker-build-and-publish.yml` 删除 `cu129` 的 `amd64 / arm64` build step、`digest-cu129` 输出, 以及 manifest 阶段 `X86_CU129 / ARM64_CU129` 的选择分支, `tag_config` 的 schema 描述从 `cu129|cu130` 收窄为 `cu130`; `release-docker-dev.yml` 同步调整调用参数。
3. Dockerfile 中的 CUDA 12 兼容 hack 全部删除: `docker/Dockerfile` 与 `docker/Dockerfile.cu134` 的 `CUIINDEX` case 只保留 `13.0.3 / 13.4.0`; 删除 `sgl-deep-ep==<ver>+cu129` 预装、对 `pyproject.toml` 的 `cuda-python>=13.0 → >=12,<13`、`flashinfer_python[cu13] → [cu12]`、`nvidia-cutlass-dsl[cu13]` 的 sed 改写、`torch 2.13` 强制重装以及 `sgl_deep_gemm-<ver>+cu129` 直链安装; Mooncake、MSCCL++、NCCL、nixl 全部固定为 `cuda13` 变体, `SGL_DEEP_GEMM_VERSION` build arg 也随之删除。
4. CI 依赖脚本收敛为 fail-fast: `scripts/ci/cuda/ci_install_dependency.sh` 在 `configure_environment` 里加入 case `"${CU_STRIP}" in 13*) ;; *) FATAL ...`, 非 `cu13x` 直接退出; 删除 `install_cuda12_deepep_wheel` (CUDA 12 的 DeepEP wheel 只存在于 `sglang` 轮子索引, 需按 local version 预装); `remove_stale_cuda12_nvidia_wheels` 去掉 `CU_MAJOR != 13` 的早退分支; `install_nccl` 无条件执行 `nvidia-nccl-cu13==2.30.7` 覆盖; `install_sglang_kernel` 不再拼 `+cu129`、不再回退到 GitHub Release 直链; `install_extra_deps` 固定 `mooncake-transfer-engine-cuda13 / nixl-cu13`; 内嵌的 `verify_imports` Python 片段不再接收 `$CU_MAJOR` 参数, NCCL 版本与运行时校验变成无条件。
5. 轮子索引生成逻辑重写 (本 PR 最值得看的代码改动): `scripts/update_kernel_whl_index.py` 删除 `SUPPORTED_CUDA_VERSIONS = ["129", "130"]`, 改为 `CUDA_LOCAL_VERSION_PATTERN = re.compile(r"\+cu(\d+)")`, `check_wheel_cuda_version` 直接比对 wheel 自身携带的 `+cuNNN` 标记, 未带标记的视为默认 CUDA 构建; `update_deepgemm_whl_index.py`、`update_nightly_whl_index.py`、`update_deepep_whl_index.py` 把 `SUPPORTED_CUDA_VERSIONS` 文本收窄为 30。
6. 运行时提示、文档与测试配套: `python/sglang/kernels/aot/python/sgl_kernel/load_utils.py` 删除 `torch.version.cuda.startswith("12")` 的 `cu129` 安装提示分支, 统一为 `pip install --upgrade sglang-kernel`; `python/sglang/kernels/jit/utils/arch.py` 的 docstring 由 `cu129` 示例改为 `cu130`; `docs/docs/get-started/install.mdx (+6/-17)` 与 `MiniMax-M3`、`Inkling cookbook snippet` 的注释同步 (`cu12` 标注为 final Hopper build, 不再重建); `test/registered/unit/tools/test_docker_build_metadata_args.py` 随镜像元数据脚本调整 (+7/-7), 但没有任何测试覆盖新的 `check_wheel_cuda_version` 匹配逻辑。

关键文件:

- `scripts/update_kernel_whl_index.py` (模块 轮子索引; 类别 source; 类型 core-logic; 符号 `check_wheel_cuda_version`, `CUDA_LOCAL_VERSION_PATTERN`, `update_wheel_index`, `SUPPORTED_CUDA_VERSIONS`): 本 PR 中信息密度最高的代码改动: CUDA 版本过滤从硬编码白名单改为解析 wheel 自身的 `+cuNNN` 标记, 是 fail-closed 的正确性提升, 且决定了 wheel 索引里会落入哪些产物。
- `scripts/ci/cuda/ci_install_dependency.sh` (模块 依赖安装; 类别 infra; 类型 infrastructure; 符号 `configure_environment`, `install_cuda12_deepep_wheel`,

install_nccl, install_sglang_kernel) : 删改量最大的基础设施文件 (+32/-78) : 引入 CU_VERSION fail-fast 校验, 删除 CUDA 12 专用的 DeepEP wheel 安装函数与全部 cu12/cu13 分支, 是 CI 侧「只剩一条 CUDA 通道」的执行点。

- python/sglang/kernels/aot/python/sgl_kernel/load_utils.py (模块 内核加载; 类别 source; 类型 core-logic; 符号 _load_architecture_specific_ops) : 运行期用户可见的提示文本改动: 删除 CUDA 12 专属的 cu129 索引安装建议, 统一为升级 PyPI wheel, 直接改变 CUDA 12 用户遇到加载失败时看到的指引。
- .github/workflows/_docker-build-and-publish.yml (模块 镜像构建; 类别 infra; 类型 infrastructure; 符号 build-cu129, digest-cu129, X86_CU129, ARM64_CU129) : reusable workflow 的 CUDA 维度收敛点: 删除 cu129 的 amd64/arm64 构建步骤与 digest 输出, manifest 阶段固定消费 cu130 digest, tag_config 的 cuda 字段随之失效。
- docker/Dockerfile (模块 容器镜像; 类别 infra; 类型 infrastructure; 符号 CUDA_VERSION, CUIINDEX, SGL_DEEP_GEMM_VERSION) : 删除 CUDA 12.6.3 / 12.9.2 的 CUIINDEX 分支、sgl-deep-ep+cu129 预装、pyproject 依赖改写与 sgl_deep_gemm+cu129 直链安装, 是镜像层退化的主要来源 (-77 行)。
- docker/Dockerfile.cu134 (模块 容器镜像; 类别 infra; 类型 infrastructure; 符号 CUDA_VERSION, CUIINDEX, NCCL_VERSION) : 与主 Dockerfile 同构的清理, 同时保留 13.4.0 通道, 说明退役只针对 CUDA 12 而非所有旧版本。
- .github/workflows/release-whl-kernel.yml (模块 发布流水线; 类别 infra; 类型 infrastructure; 符号 build-cu129-matrix, release-cu129, target) : 一次性删除 105 行 : build-cu129-matrix、release-cu129 与 target 选项中的 cu129, 直接决定 sglang-kernel wheel 不再产出 CUDA 12 变体。
- .github/workflows/release-whl-deepep.yml (模块 发布流水线; 类别 infra; 类型 infrastructure; 符号 build-cu129-matrix, target, release-whl, release-pypi) : 除删除 build-cu129-matrix 外, 还把 release-whl / release-pypi 的 needs 与 if 条件简化为只依赖 build-cu130-matrix, 是发布语义变化最明显的一处。
- test/registered/unit/tools/test_docker_build_metadata_args.py (模块 元数据校验; 类别 test; 类型 test-coverage) : 唯一的测试配套改动, 随镜像元数据脚本调整 (+7/-7) ; 但没有为新的 check_wheel_cuda_version 增加任何覆盖, 是测试缺口所在。
- docs/docs/get-started/install.mdx (模块 安装文档; 类别 docs; 类型 documentation) : 面向用户反映 CUDA 12 退役的安装文档改动 (+6/-17) , 是用户感知这次断点的主要入口。

关键符号: check_wheel_cuda_version, CUDA_LOCAL_VERSION_PATTERN, update_wheel_index, configure_environment, install_nccl, install_sglang_kernel, install_extra_deps, verify_imports, install_cuda12_deepep_wheel (删除) , build-cu129-matrix (删除 job) , release-cu129 (删除 job) , _load_architecture_specific_ops

关键源码片段

[scripts/update_kernel_whl_index.py](#)

本 PR 中信息密度最高的代码改动：CUDA 版本过滤从硬编码白名单改为解析 wheel 自身的 +cuNNN 标记，是 fail-closed 的正确性提升，且决定了 wheel 索引里会落入哪些产物。

```
# scripts/update_kernel_whl_index.py

DEFAULT_CUDA_VERSION = "130"
# CUDA wheel 自带的本地版本标记，例如 sglang_kernel-0.4.6.post1+cu130-...whl
CUDA_LOCAL_VERSION_PATTERN = re.compile(r"\+cu(\d+)")

def check_wheel_cuda_version(path_name, target_cuda_version):
    # 非 CUDA 后端直接跳过：ROCm/MUSA 把后端编码在 local version 里（如 +rocm720），
    # XPU 则使用独立包名 sglang_kernel_xpu-*。
    if re.search(r"\+(rocm|musa)", path_name) or path_name.startswith(
        "sglang_kernel_xpu-"
    ):
        return False

    # 关键改动：不再遍历 SUPPORTED_CUDA_VERSIONS 的已知版本列表，而是读取 wheel
    # 自身的 +cuNNN 标记。这样即便某个 wheel 是用本脚本从未见过的 CUDA 版本构建的，
    # 也会被直接拒绝，而不是错误地落进目标索引（旧逻辑对默认版本采用「反查白名单」）。
    match = CUDA_LOCAL_VERSION_PATTERN.search(path_name)
    if match is not None:
        return match.group(1) == target_cuda_version

    # 没有标记的 wheel 即默认 CUDA 版本的构建，例如
    # sglang_kernel-0.4.0-cp310-abi3-manylinux2014_x86_64.whl
    # （PyPI 拒绝 local version，因此上传到 PyPI 时会剥离 +cuNNN 标记）。
    return target_cuda_version == DEFAULT_CUDA_VERSION

def update_wheel_index(cuda_version=DEFAULT_CUDA_VERSION, rocm_version=None):
    index_dir = pathlib.Path(f"sgl-whl/cu{cuda_version}/sglang-kernel")
    index_dir.mkdir(exist_ok=True, parents=True)
    base_url = "https://github.com/sgl-project/whl/releases/download"

    # 调用方从此只需传 cu130；check_wheel_cuda_version 负责把 +cu129 等历史
    # 产物挡在索引之外，避免退役通道的 wheel 继续出现在新索引里。
    for path in sorted(pathlib.Path("python/sglang/kernels/aot/dist").glob("*.whl")):
        if not check_wheel_cuda_version(path.name, cuda_version):
            continue
        with open(path, "rb") as f:
            sha256 = hashlib.sha256(f.read()).hexdigest()
        ver = re.findall(
            r"sglang_kernel-([0-9.]+(?:\.[0-9]+)?(?:\+cu[0-9]+)?-)", path.name
        )[0]
        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')
```

scripts/ci/cuda/ci_install_dependency.sh

删改量最大的基础设施文件 (+32/-78)：引入 CU_VERSION fail-fast 校验，删除 CUDA 12 专用的 DeepEP wheel 安装函数与全部 cu12/cu13 分支，是 CI 侧「只剩一条 CUDA 通道」的执行点。

```
# scripts/ci/cuda/ci_install_dependency.sh (退役 CUDA 12 后的形态)
```

```
configure_environment() {
    # CU_VERSION 决定 PyTorch 索引地址、FlashInfer JIT cache 索引以及 sglang wheel 索引。
    # 现在只剩 CUDA 13 通道：PyTorch 2.14 不再为 cu12 通道所用的 cu129 索引发布
    # CUDA 12 wheel，因此这里用 fail-fast 取代过去的 CU_MAJOR 双分支。
    CU_VERSION="${CU_VERSION:-cu130}"
    CU_STRIP="${CU_VERSION#cu}"
    case "${CU_STRIP}" in
        13*) ;; # 仅放行 cu13x
        *) echo "FATAL: unsupported CU_VERSION=${CU_VERSION}; only CUDA 13 is supported";
        exit 1 ;;
    esac
}

# install_cuda12_deepep_wheel 已整段删除：该函数原本从 pyproject.toml 抓取 sgl-deep-ep
# 版本，再装上 "sgl-deep-ep==${version}+${CU_VERSION}" 这个只存在于 sglang 轮子索引的
# local version wheel，以便满足 pyproject 的公共版本 pin。CUDA 13 直接用 PyPI wheel。

install_nccl() {
    # PyTorch 会锁到 2.29.7，所以本次覆盖必须放在所有解析 Python 依赖的命令
    # (含 Imms-eval) 之后执行；原先按 CU_MAJOR 判断的 else 分支已删除。
    $PIP_CMD install "nvidia-nccl-cu13==2.30.7" --force-reinstall --no-deps $PIP_INSTALL_SUFFIX
    mark_step_done "${FUNCNAME[0]}"
}

install_sglang_kernel() {
    # 不再拼接 +cu129 本地版本，也不再回退到 GitHub Release 直链下载
    # sgl_deep_gemm-<ver>+cu129-py3-none-manylinux2014_${uname -m}.whl。
    SGL_DEEP_GEMM_VERSION=$(grep -Po -m1 '(?<=sgl-deep-gemm==)[0-9A-Za-z\.\-]+' python/
    pyproject.toml)
    if installed_wheel_ok sgl-deep-gemm "${SGL_DEEP_GEMM_VERSION}"; then
        echo "sgl-deep-gemm==${SGL_DEEP_GEMM_VERSION} already installed, keeping it"
    else
        $PIP_CMD install "sgl-deep-gemm==${SGL_DEEP_GEMM_VERSION}" --force-reinstall $PIP_
        INSTALL_SUFFIX
    fi
}

install_extra_deps() {
    MOONCAKE_VERSION="0.3.13"
    NIXL_VERSION="1.3.0"
    # 四个变量不再随 CU_MAJOR 切换：CUDA 12 的 mooncake-transfer-engine 与
```

```
# nixl-cu12 / nvidia-cuda-nvrtc-cu12 组合已随通道一起删除。
MOONCAKE_PKG="mooncake-transfer-engine-cuda13==${MOONCAKE_VERSION}"
MOONCAKE_STALE_PKG="mooncake-transfer-engine" # 需要强制卸载的旧 CUDA 12 变体
NIXL_BIN_NAME="nixl-cu13"
EXTRA_NVIDIA_SPECS="nvidia-cuda-nvrtc"
}
```

评论区精华

本 PR 的公开讨论非常稀薄，唯一的 review 是 Fridge003 的 APPROVED 空评论，`review_comments` 为 0，没有任何行内技术交锋。因此需要从 PR body 的状态槽位里读取「未被讨论但客观存在」的信号：

PR body 的 `pr-states` 区块显示 Latest PR Test (Base)、Latest PR Test (Extra)、Latest PR Test (AMD ROCm 10) 三个 run 全部为  (Run #34440714447 / #34440714159 / #34440714312) 。

兼容承诺：「Already-published `-cu129` / `-cu12` image tags are unaffected. v0.5.19 is the last release with a CUDA 12 lane.」——这是对 CUDA 12 用户的唯一书面保证，且没有在 review 中被进一步确认或质疑。

代码层面的设计取舍（白名单 → 正则标签匹配）由作者单方面做出，未经过讨论；`_docker-build-and-publish.yml` 中 `tag_config` 传入的 `cuda` 字段被静默忽略、manifest 固定使用 `cu130 digest` 这一行为变化，也没有在任何评论中被提出。

- 空 body 的 APPROVED 是否覆盖失败中的 PR CI (other): 维护者以「已提前公告 + 通道退役属既定计划」为前提直接放行，CI 失败未在 PR 中被讨论或复跑说明，因此这些红灯究竟是被删除 job 造成的预期噪音还是真实回归，无法从本 PR 记录中判断。
- 已发布 `cu129` / `cu12` 镜像与 wheel 的兼容承诺 (documentation): 不删除历史产物、只停止重建，旧 tag 保持可拉取，用户短期不会遇到镜像拉取失败；但「最后一个 CUDA 12 版本」的结论只在 PR body 中书面给出，未落到 release notes 或安装文档的显式声明里。
- 索引匹配策略从版本白名单改为 wheel 标签解析 (design): 属于 fail-closed 的正确性提升，同时移除了未来新增 CUDA 版本时需要同步更新的白名单维护负担；代价是行为改变（未知版本 wheel 从可能通过变为必然拒绝）缺乏单测覆盖。

风险与影响

- 风险：

1. `tag_config` 的静默语义漂移：`_docker-build-and-publish.yml` 的 manifest job 删除了 `CUDA=$(...jq -r '.cuda')` 分支，改为无条件使用 `X86_CU130` / `ARM64_CU130`。若仍有调用方沿用 `{"cuda":"cu129","tags":[...]}` 的旧配置，不会报错，而是把这些 tag 名打到 CUDA 13 镜像上，形成「标签说 cu12、内容是 cu13」的错配。当前仓库内 `release-docker-dev.yml` 已同步，但这是 reusable workflow，仓库外或历史调用方存在残留风险。

1. 新的索引匹配逻辑缺少测试：`check_wheel_cuda_version` 从「白名单里反查」变成「读取 wheel 自身 `+cuNNN`」，是 fail-closed 的正向改进，但没有任何单测覆盖（仓库有

`test/registered/unit/tools/` 目录，却只改了 docker 元数据测试）。边界情形（无标记 wheel 归入默认 130、`+rocm720 / sglang_kernel_xpu-` 排除、未知 cu 版本被拒绝）只能靠发布流程事后验证。

2. 报错提示降级：`load_utils.py` 删掉了 CUDA 12 专属的 `--index-url https://docs.sglang.ai/whl/cu129/` 提示后，处于 CUDA 12 环境的用户加载 `common_ops` 失败时只会被引导执行 `pip install --upgrade sglang-kernel`，而该 wheel 在 CUDA 12 驱动上无法工作，用户会陷入「按提示操作仍失败且无下一步线索」的状态。
3. 发布 job 条件收窄：`release-whl-deepep.yml` 的 `release-whl / release-pypi` 从 `always() && ...` 的显式组合判断改为纯 `needs.build-cu130-matrix.result == 'success'`。语义上更清晰，但当上游 job 被 skip（例如 if 判定仓库不匹配）时，下游 release job 的触发行为与过去不同，需要确认手动重跑 / 部分发布场景未受影响。
4. CUDA 12 用户的硬断点：`cu129` wheel 与镜像不再重建，`v0.5.19` 成为最后一个含 CUDA 12 通道的版本；由于 PyTorch 2.14 本身也不再发布 CUDA 12 wheel，这条路径事实上不可逆。风险主要在用户侧感知与支持成本，而非代码正确性。
 - 影响：用户侧：仅支持 CUDA 12 驱动栈的机器（例如停留在 12.x 的 Hopper 集群）无法再获得新的 sglang wheel 与镜像，只能继续使用既有 tar 出的镜像 tag 或 $\leq v0.5.19$ 的版本；`docs/docs/get-started/install.mdx` 的改动使这一断点在安装文档上可见。已经发布的 `-cu129 / -cu12` tag 保持可拉取，短期不构成服务中断。系统与 CI 侧：发布矩阵从两条 CUDA 通道收敛为一条，`_docker-build-and-publish.yml` 少了两个镜像构建 step（AMD64/ARM64 各一个 `cu129`），Dockerfile 少了整段 CUDA 12 改写与重装逻辑，构建时间与失败面同时下降；`ci_install_dependency.sh` 增加 `fail-fast` 后，任何仍传 `CU_VERSION=cu129` 的 runner 会立刻以 FATAL 退出而不是带着错误依赖继续跑。团队侧：为 PyTorch 2.14 升级扫清前置障碍，此后 CUDA 相关改动只需维护 13.x 一条路径，长期维护成本下降；代价是发布流程与镜像脚本的大面积删除缺乏自动化验证，回归主要依赖下一次真实 release 演练。
 - 风险标记：发布链路大范围删除，CUDA 12 用户兼容性断点，PR CI 状态为失败，索引逻辑缺少测试覆盖，`tag_config.cuda` 字段静默失效，错误提示指引退化

关联脉络

- PR #38767 [AMD][CI] Retire the ROCm 7.0 kernel wheel: 同类「退役旧 wheel/ 镜像通道」的运维动作，且同样改动 `.github/workflows/release-whl-kernel.yml`，两者共同把发布矩阵收敛为更少的后端变体；可对比 `tag/` 索引退役时的文档同步方式。
- PR #38842 Revert "[CI] Temporarily disable GB300 tests": 同样涉及 `release-whl-deepgemm.yml` 与 NVIDIA CI 工作流，说明该文件正处在密集调整期；本 PR 删除 `cu129` `deepgemm` 矩阵，与该 PR 的恢复动作互为上下文。
- PR #38881 [CI] Drop the GPTQ dynamic-config test for the deleted non-Marlin kernel: 同一批「清理已失效通道 / 测试」的收尾 PR，都在移除仓库中不再有对应产物的 CI 配置，体现维护者正在系统性回收历史包袱。
- PR #38830 [JIT] Port the expert-pack MXFP4 kernels to `load_jit` and fix their launch limits: 同属 `python/sglang/kernels` JIT 基础设施；本 PR 更新了 `sglang/kernels/jit/utils/arch.py` 中关于 CUDA 版本选择的文档说明，两者共同定义 JIT 侧

对 CUDA 版本的假设。