PR #34041 完整报告

sgl-project/sglang

Docker: install DeepEP from release wheels

合并时间：2026-08-08 14:51

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34041

执行摘要

- 一句话：Docker 镜像改用 sgl-deep-ep 发布 wheel，删除 DeepEP 源码编译阶段
- 推荐动作：值得快速浏览，作为「从源码编译迁移到发布 wheel」的 Docker 依赖治理范例。关注三点：一是 CUDA 12 用 cu129 wheel + PEP 440 local version 满足公开版本约束的技巧；二是从 pyproject 单一来源读取版本避免双重 pin 的实践；三是确认原 configs.cuh 超时定制是否已内置于发布 wheel，这决定是否存在潜在的通信超时行为变化。

功能与动机

PR body 明确指出：sgl-deep-ep==0.1.0 已是 SGLang 的正式发布依赖，主 Dockerfile 应当直接消费发布 wheel，而不是每次构建镜像都克隆并编译 DeepEP 源码。这能消除构建期的源码拉取、补丁和编译开销，同时让 Dockerfile 与 pyproject.toml 的依赖版本保持一致。

实现拆解

1. 删除 DeepEP 源码构建阶段docker/Dockerfile：移除 deepep_builder 阶段（约 89 行），包括按 GRACE_BLACKWELL/HOPPER_SBO 分支 clone DeepEP 不同分支 /commit、对 csrc/kernels/configs.cuh 的超时参数 sed 修改、按 CUDA 版本设置 TORCH_CUDA_ARCH_LIST 并执行 setup.py bdist_wheel 等逻辑。
2. framework 阶段移除 DeepEP wheel/ 源码拷贝：原 COPY --from=deepep_builder /wheels 与 /build/DeepEP 源码目录及安装逻辑全部删除，避免镜像携带深层源码目录。
3. 版本单一来源：在 torch_deps 阶段用 sed 从 python/pyproject.toml 提取 sgl-deep-ep==X.Y.Z 版本号，Dockerfile 不再独立维护版本 pin，杜绝双源漂移。
4. CUDA 分支差异化安装：CUDA 13 直接依赖 pyproject 的 ==0.1.0 从公共索引解析安装；CUDA 12 先以 --no-deps 方式从 https://docs.sglang.ai/whl/cu129/ 预装 sgl-deep-ep==0.1.0+cu129，利用 PEP 440 local version 语义使其满足 ==0.1.0 约束，再在完整依赖解析时跳过重新下载。
5. 清理 build-arg 与 workflow 引用：删除 GRACE_BLACKWELL、HOPPER_SBO、DEEPEP_COMMIT、HOPPER_SBO_DEEPEP_COMMIT、BUILD_AND_DOWNLOAD_PARALLEL 等参数，并同步移除 .github/workflows/_docker-build-and-publish.yml、nightly-72-gpu-gb200.yml、release-docker-dev.yml 中的 --build-arg GRACE_BLACKWELL=... 传参。

6. 配套验证：通过外部迁移契约测试（旧 Dockerfile RED、新 Dockerfile GREEN）、Dockerfile 静态解析（345 条指令、11 个预期阶段、27 个 COPY --from 引用全部解析）、PyPI CUDA 13 与 SGLang cu129 wheel 在 x86_64/aarch64 的 Python 3.12 解析验证，以及 workflow YAML 解析和 pre-commit。

关键文件：

- docker/Dockerfile（模块 镜像构建；类别 infra；类型 infrastructure；符号 deepep_builder, torch_deps, hpc_ops_builder, flashinfer_cache）：核心变更文件：删除 deepep_builder 整个构建阶段，改为从发布 wheel 安装 DeepEP，并清理 5 个 build-arg；CUDA 12/13 采用不同的安装路径。
- .github/workflows/_docker-build-and-publish.yml（模块 镜像发布；类别 infra；类型 infrastructure）：移除多个构建任务中传入的 GRACE_BLACKWELL build-arg，与 Dockerfile 参数清理联动。
- .github/workflows/nightly-72-gpu-gb200.yml（模块 夜间流水线；类别 infra；类型 infrastructure）：夜间 GB200 构建去掉 GRACE_BLACKWELL=1 传参，配合 Dockerfile 参数清理。
- .github/workflows/release-docker-dev.yml（模块 发布工作流；类别 infra；类型 infrastructure）：开发版发布流程去掉 GRACE_BLACKWELL=0 传参，与 Dockerfile 参数清理保持一致。

关键符号：未识别

关键源码片段

docker/Dockerfile

核心变更文件：删除 deepep_builder 整个构建阶段，改为从发布 wheel 安装 DeepEP，并清理 5 个 build-arg；CUDA 12/13 采用不同的安装路径。

```
# torch_deps 阶段内：从 pyproject.toml 读取 DeepEP 版本，避免 Dockerfile 重复维护版本 pin
RUN --mount=type=cache,target=/root/.cache/pip \
    --mount=type=cache,target=/root/.cargo/registry \
    ... \
    && echo '__version__ = "0.0.0"' > sglang/version.py \
    && touch README.md \
    && touch LICENSE \
    # 用 sed 提取 pyproject.toml 中的 "sgl-deep-ep==X.Y.Z"，作为唯一版本来源
    && SGL_DEEP_EP_VERSION="$(sed -n 's/^[[:space:]]*"sgl-deep-ep==\([^"]*\)",/\1/p'
    pyproject.toml)" \
    && test -n "${SGL_DEEP_EP_VERSION}" \
    # CUDA 12 没有公共 wheel，需要从 SGLang cu129 索引预装 local version 的 wheel；
    # PEP 440 下 "0.1.0+cu129" 满足 pyproject 的 "==0.1.0" 约束，后续完整依赖解析不会冲突
    && if [ "${CUDA_VERSION%%.*}" = "12" ]; then \
         python3 -m pip install \
           "sgl-deep-ep==${SGL_DEEP_EP_VERSION}+cu129" \
           --index-url "https://docs.sglang.ai/whl/cu129/" \
           --no-deps; \
       fi \
```

```
    # CUDA 12 还需降级 cuda-python 与 flashinfer_python 索引标签
    && if [ "${CUDA_VERSION%%.*}" = "12" ]; then \
         sed -i 's/cuda-python>=13\.0/cuda-python>=12,<13/' pyproject.toml && \
         sed -i 's/flashinfer_python\[cu13\]/flashinfer_python[cu12]/' pyproject.toml; \
       fi \
    …
```

评论区精华

该 PR 没有任何 review 评论（comments_count=0、review_comments_count=0），讨论主要沉淀在 PR body 的 Validation 部分：作者明确说明本地无法运行 Docker，完整 BuildKit 构建由 release workflow 实际验证；同时用外部迁移契约测试（RED/GREEN）和 Dockerfile 静态解析来弥补无法本地构建的验证缺口。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 定制补丁可能丢失：原 deepep_builder 阶段会对 DeepEP 源码 configs.cuh 执行 NUM_CPU_TIMEOUT_SECS 100→1000、NUM_TIMEOUT_CYCLES 200000000000→2000000000000 的 sed 修改，迁移到发布 wheel 后这些修改不再显式应用。若 sgl-deep-ep==0.1.0 未内置同等超时容错，长尾通信场景可能出现 CPU/cycle 超时行为差异。
  2. 外部 wheel 索引依赖：CUDA 12 镜像依赖 SGLang cu129 索引同步发布 +cu129 local version wheel；若未来 pyproject 升级 sgl-deep-ep 版本而该索引未同步，构建将失败，需要维护双索引发布节奏。
  3. 构建未在本地完整验证：作者明确说明 Docker 不可用，完整 BuildKit 构建依赖 release workflow 发现潜在问题。
  4. 参数清理影响外部用户：删除 GRACE_BLACKWELL、HOPPER_SBO 等 build-arg 属于破坏性变更，任何外部基于该 Dockerfile 自定义构建的脚本若仍传这些参数会报错（不过 Docker 对未知 build-arg 默认容忍，风险较低）。 - 影响：影响范围集中在镜像构建链路：所有使用 docker/Dockerfile 的 workflow（release-docker-dev、nightly、_docker-build-and-publish）不再编译 DeepEP，构建时间明显缩短、镜像内不再包含 DeepEP 源码目录、体积减小，且每次构建不再依赖 GitHub 网络拉取 DeepEP 仓库。对运行时的模型推理、内核行为无任何代码路径变化；对最终用户无感知。团队后续升级 DeepEP 时只需改 pyproject.toml 一处版本，维护成本降低。 - 风险标记：构建流程重构，定制补丁可能丢失，外部 wheel 索引依赖，本地未完整验证构建

关联脉络

- PR #33932 Install DeepEP from release wheels: 同一 DeepEP 迁移工作的前序 / 并行 PR：CI 脚本已改为安装 sgl-deep-ep 发布 wheel，本 PR 将 Docker 镜像构建链路同步迁移。

- PR #34030 docker: pin Kimi images to SGLang commit: 同属 Docker 镜像构建基础设施治理，反映近期对 docker 构建稳定性的持续改进。