PR #29910 完整报告

sgl-project/sglang

[Dep] Upgrade flashinfer to 0.6.14

合并时间：2026-07-10 08:52

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29910

执行摘要

- 一句话：升级 flashinfer 依赖至 0.6.14
- 推荐动作：该 PR 是必要的依赖升级，值得仔细 review 和测试。建议合并后密切关注 CI 中各测试项（尤其是 attention 和 MoE）的执行结果。MoE 测试的修改值得学习，展示了如何适配上游库的 breaking change。同时，建议跟踪 flashinfer-cubin 发布状态，及时恢复 pyproject.toml 依赖。

功能与动机

升级 flashinfer 到 0.6.14，以获取最新特性与修复，并保持与 flashinfer 新版本的兼容性。由于 flashinfer 0.6.14 的 cubin 包暂未发布到 PyPI，改为从 flashinfer.ai/whl 显式安装。

实现拆解

1. 版本号更新：在 pyproject.toml 中将 flashinfer_python[cu13] 版本从 0.6.12 改为 0.6.14，并移除 flashinfer_cubin 依赖行。在 engine.py 中更新 assert_pkg_version 检查的版本号。在 common.py 中更新 check_pkg_version_at_least 的文档字符串。
2. CI 安装脚本改造：在 ci_install_dependency.sh 中新增 install_flashinfer_cubin 函数，从 flashinfer.ai/whl 直接安装 cubin 包。同时调整 FLASHINFER_CUBIN_REQUIRED 的获取方式，改为从 FLASHINFER_PYTHON_REQUIRED 派生。主流程添加调用。
3. Dockerfile 调整：增加 flashinfer-cubin 安装步骤，将 cubin 包复制到 /flashinfer_jit_output 并在最终阶段复制。更新 FLASHINFER_VERSION 变量。
4. MoE 测试适配：修改 test_cutedsl_moe.py 中的 test_v2_cuda_graph_parity 方法。由于 flashinfer 废弃了非 CUDA graph 模式下的预分配缓冲区接口，测试改为使用 torch.cuda.CUDAGraph 显式 capture 并 replay，同时放宽两次 replay 之间的精度比较方式（由 assert_close 改为 assertLess 以允许一定误差）。

关键文件：

- python/pyproject.toml（模块 依赖配置；类别 config；类型 configuration）：核心依赖版本升级，并移除了 flashinfer_cubin 依赖
- python/sglang/srt/entrypoints/engine.py（模块 入口配置；类别 source；类型 core-logic）：更新 flashinfer 版本检查条件
- python/sglang/srt/utils/common.py（模块 工具函数；类别 source；类型 core-logic）：更新 check_pkg_version_at_least 的文档字符串示例

- scripts/ci/cuda/ci_install_dependency.sh（模块 CI 脚本；类别 infra；类型 infrastructure；符号 install_flashinfer_cubin）：新增显式安装 flashinfer-cubin 的函数，并调整版本获取方式
- docker/Dockerfile（模块 Docker 构建；类别 infra；类型 infrastructure）：增加 cubin 安装步骤并更新版本变量
- test/registered/moe/test_cutedsl_moe.py（模块 MoE 测试；类别 test；类型 test-coverage；符号 test_v2_cuda_graph_parity）：适配 flashinfer 新版 CUDA graph wrapper API

关键符号：assert_pkg_version, check_pkg_version_at_least, install_flashinfer_cubin, test_v2_cuda_graph_parity

关键源码片段

python/pyproject.toml

核心依赖版本升级，并移除了 flashinfer_cubin 依赖

```
# pyproject.toml (head) 依赖列表中的 flashinfer 部分
dependencies = [
    # ... 其他依赖省略
    "flash-attn-4==4.0.0b15",
    # flashinfer_python 版本从 0.6.12 提升到 0.6.14
    "flashinfer_python[cu13]==0.6.14", # 与 Dockerfile jit-cache 版本保持对齐
    # flashinfer_cubin 行被移除，现在由 CI/Docker 显式安装
    "gguf",
    # ...
]
```

scripts/ci/cuda/ci_install_dependency.sh

新增显式安装 flashinfer-cubin 的函数，并调整版本获取方式

```
# scripts/ci/cuda/ci_install_dependency.sh 关键新增

# 在 main() 中新增调用
main() {
    # ... 前面的步骤
    install_flashinfer_cubin # 显式安装 flashinfer-cubin
    download_flashinfer_cache
    # ...
}

# 新函数：安装 flashinfer-cubin 包
install_flashinfer_cubin() {
    if [ "$UNINSTALL_CUBIN" = false ]; then
        echo "flashinfer-cubin==${FLASHINFER_CUBIN_REQUIRED} already installed, skipping install"
    else
        # cubin 包无 CUDA 版本后缀，使用基础仓库地址
```

```
        $PIP_CMD install "flashinfer-cubin==${FLASHINFER_CUBIN_REQUIRED}" \
            --index-url https://flashinfer.ai/whl $PIP_INSTALL_SUFFIX
    fi
    mark_step_done "${FUNCNAME[0]}"
}
```

test/registered/moe/test_cutedsl_moe.py

适配 flashinfer 新版 CUDA graph wrapper API

```
# test/registered/moe/test_cutedsl_moe.py 中的 test_v2_cuda_graph_parity 方法（已修改）
def test_v2_cuda_graph_parity(self):
    """Verify non-graph and cuda_graph v2 wrappers produce identical results.

    由于 flashinfer 0.6.14 重构了 wrapper，不再支持非 graph 模式下传递预分配缓冲区，
    因此测试改为：先预热三次，再通过 torch.cuda.CUDAGraph 显式 capture，
    然后 replay 两次并比较两次 replay 的结果（允许较小的误差）。
    """
    test_cases = [
        (128, 256, 512, 256, 2),
        (256, 256, 512, 256, 4),
    ]
    for (num_tokens, hidden_size, intermediate_size, num_experts, top_k) in test_cases:
        with self.subTest(...):
            tensors = _create_cutedsl_wrapper_tensors(...)
            wrapper_args = dict(num_experts=num_experts, top_k=top_k,
                                hidden_size=hidden_size, intermediate_size=intermediate_size)
            wrapper_no_graph = CuteDslMoEWrapper(**wrapper_args, use_cuda_graph=False)
            wrapper_graph = CuteDslMoEWrapper(**wrapper_args, use_cuda_graph=True,
                                              max_num_tokens=num_tokens)
            with torch.no_grad():
                out_no_graph = _run_wrapper(wrapper_no_graph, tensors)
                # 预热三次，确保 graph 准备就绪
                for _ in range(3):
                    _run_wrapper(wrapper_graph, tensors)
                torch.cuda.synchronize()
                # 显式 capture graph
                graph = torch.cuda.CUDAGraph()
                with torch.cuda.graph(graph):
                    graph_output = _run_wrapper(wrapper_graph, tensors)
                torch.cuda.synchronize()
                # 两次 replay 并克隆结果
                graph.replay()
                torch.cuda.synchronize()
                out_graph1 = graph_output.clone()
                graph.replay()
                torch.cuda.synchronize()
                out_graph2 = graph_output.clone()
            # 验证非 graph 与 graph 输出一致
            torch.testing.assert_close(out_no_graph, out_graph1,
```

```
                    atol=1e-2, rtol=1e-2,
                    msg="non-graph vs cuda_graph wrapper outputs diverge")
# 验证两次 replay 的输出差异小于阈值
max_diff = (out_graph1 - out_graph2).abs().max().item()
self.assertLess(max_diff, 0.5,
                f"cuda_graph replay diverged too much: max_diff={max_diff}")
# ... 后续参考精度比较省略
```

评论区精华

Review 中主要讨论了三方面问题：

- Dockerfile 中 cubin 安装的必要性：Fridge003 担心增加镜像大小，b8zhong 解释 cubin 原本就由 pip 包含，显式安装是为保持一致，且包大小仅 150MB，可以接受。
- MoE 测试修改原因：Fridge003 询问为何修改测试，b8zhong 指出 flashinfer 重构了 CUDA graph wrapper，无法再传递预分配缓冲区，因此测试改为使用 CUDA graph capture 方式验证正确性。
- 是否保留 pyproject.toml 中的 cubin 依赖：mmangkad 倾向于保留以便自动安装，但最终由于 cubin 0.6.14 未发布到 PyPI，决定移除并采用显式安装，待后续恢复。
  - Dockerfile 中显式安装 flashinfer-cubin 的必要性与镜像大小影响 (design): 保留该变更，因为 cubin 包原本就存在，显式安装镜像大小影响不大。
  - MoE 测试修改以适应 flashinfer 的 CUDA graph wrapper 重构 (correctness): 采纳新测试方式，使用 graph capture 和 replay 验证正确性，并放宽 replay 之间的精度比较为 max_diff < 0.5。
  - 是否将 flashinfer-cubin 作为 pyproject 依赖保留 (design): 决定移除 pyproject 依赖，因为 cubin 0.6.14 尚未发布到 PyPI，改为在 CI/Docker 中显式安装。待后续 PyPI 发布后可考虑恢复。

风险与影响

- 风险：兼容性风险：flashinfer 0.6.14 可能引入与旧版本不兼容的 API 变更（如 CUDA graph wrapper），需要确保所有使用 flashinfer 的模块（attention、MoE 等）正常工作。目前只有 MoE 测试做了适配，其他使用路径（如 attention backend）未修改，可能隐藏回归。 部署风险：CI 和 Docker 构建方式改变，如果 flashinfer.ai/whl 不可达或版本不对应，可能导致安装失败。pyproject.toml 移除了 cubin 依赖，用户从源码安装时需要确保正确安装 cubin 包。 性能影响：无直接性能变更，但新版 flashinfer 可能包含性能优化或退化，需基准测试验证。
- 影响：用户影响：用户需要确保安装的 flashinfer 版本与要求一致（0.6.14），否则在启动时会被拦截。从源码安装的用户可能需要额外步骤安装 flashinfer-cubin。 系统影响：CI 流程将显式安装 cubin 而不是通过 pip 自动解析，Docker 镜像中 cubin 会作为单独层被包含，镜像大小增加约 150MB。 团队影响：团队需要确认所有依赖 flashinfer 的功能（如 attention、MoE 等）在新版本下通过 CI 测试。历史 PR 中有大量与 flashinfer 相关的修改，需注意回归。
- 风险标记：依赖升级兼容性风险，测试适配可能覆盖不足，安装流程变更

关联脉络

- 暂无明显关联 PR