PR #30870 完整报告

sgl-project/sglang

Avoid TileLang CUDA runtime pollution

合并时间：2026-07-14 22:30

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30870

执行摘要

- 一句话：避免 TileLang CUDA 运行时污染导致启动失败
- 推荐动作：建议精读。本 PR 展示了一种基于延迟导入和 CUDA 库选择策略来隔离不同 CUDA 运行时的有效模式。其中关于 /proc/self/maps 候选处理以及 NamedTuple 的设计选择值得在类似场景中复用。

功能与动机

导入模型注册表时，DeepSeek-V4 的 TileLang 内核会加载其私有 CUDA stub 库，导致无关模型（如 Qwen3-VL Hopper）在 CUDA IPC / FlashInfer 集合通信初始化时失败，无法启动服务。

实现拆解

1. 延迟导入 MHC 内核：在 python/sglang/srt/models/deepseek_v4.py 中引入 MhcOps NamedTuple 和 _get_mhc_ops 函数（带 functools.cache），将原先模块顶层的条件导入（if _is_xpu: ... else: from sglang.kernels.ops.layernorm.mhc import ...）移至函数内部。只有当 DeepSeek-V4 的层实际执行 hc_pre 或 forward 时才调用该函数加载内核，避免模型注册时自动导入 TileLang。
2. 优化 CUDA 运行时选择：在 python/sglang/srt/distributed/device_communicators/cuda_wrapper.py 中重写 find_loaded_library，改为收集 /proc/self/maps 中所有匹配 lib_name 的候选路径，并处理 (deleted) 后缀。最后优先返回不包含 "stub" 的路径（即真正的 CUDA 运行时），避免 TileLang 的 libcudart_stub.so 被选中导致符号解析失败。
3. 新增单元测试：创建 test/registered/unit/distributed/test_cuda_wrapper.py，使用 monkeypatch 模拟 /proc/self/maps 内容，验证以下场景：a) 当存在 TileLang stub 和真正的 cudart 时，优先返回真正的 cudart；b) 当路径含有 (deleted) 后缀时，能正确剥离后缀并返回有效路径。

关键文件：

- python/sglang/srt/models/deepseek_v4.py（模块 DeepSeek 模型；类别 source；类型 data-contract；符号 MhcOps, _get_mhc_ops）：核心修改：延迟导入 TileLang 驱动的 MHC 内核，通过 MhcOps NamedTuple 和 _get_mhc_ops 函数将模块级导入改为按需加载，避免模型注册时触发 CUDA 运行时冲突。

- python/sglang/srt/distributed/device_communicators/cuda_wrapper.py（模块 CUDA 封装；类别 source；类型 dependency-wiring）：重写 find_loaded_library 函数：收集所有候选路径，去除 (deleted) 后缀，并优先返回不含 "stub" 的路径（即真正的 CUDA 运行时），避免 TileLang 的 libcudart_stub.so 被错误选中。
- test/registered/unit/distributed/test_cuda_wrapper.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_find_loaded_library_prefers_real_cudart_over_tilelang_stub, test_find_loaded_library_strips_deleted_suffix）：新增单元测试，验证 find_loaded_library 在两种关键场景下的行为：优先选择真正的 cudart 而非 TileLang stub，以及正确处理 (deleted) 后缀。

关键符号：MhcOps, _get_mhc_ops, find_loaded_library, test_find_loaded_library_prefers_real_cudart_over_tilelang_stub, test_find_loaded_library_strips_deleted_suffix

关键源码片段

python/sglang/srt/models/deepseek_v4.py

核心修改：延迟导入 TileLang 驱动的 MHC 内核，通过 MhcOps NamedTuple 和 _get_mhc_ops 函数将模块级导入改为按需加载，避免模型注册时触发 CUDA 运行时冲突。

```
# -*- coding: utf-8 -*-
# --- 前置代码省略，直接展示关键新增部分 ---

# 定义一个 NamedTuple 来承载 MHC 内核操作，替代原始元组，提高可读性和类型安全
class MhcOps(NamedTuple):
    hc_split_sinkhorn: Callable[..., Any]
    mhc_fused_post_pre: Optional[Callable[..., Any]]
    npu_hc_pre: Optional[Callable[..., Any]]


@functools.cache
def _get_mhc_ops() -> MhcOps:
    """按需加载 MHC 内核，仅在 DeepSeek-V4 层实际需要时才导入 TileLang 相关的模块。

    模型注册阶段会遍历所有模型模块，如果此时直接导入 ``sglang.kernels.ops.layernorm.mhc``
    就会触发 TileLang 的初始化，加载其私有的 CUDA stub 库，破坏无关模型的 CUDA IPC 设置。
    通过延迟到 forward 执行时再导入，可以避免这种全局副作用。
    返回的 MhcOps 实例包含三个可调用对象，对于 XPU 平台只有 ``hc_split_sinkhorn`` 有效。
    """
    if _is_xpu:
        from sgl_kernel import hc_split_sinkhorn
        return MhcOps(hc_split_sinkhorn, None, None)

    from sglang.kernels.ops.layernorm.mhc import (
        hc_split_sinkhorn,
        mhc_fused_post_pre,
        npu_hc_pre,
    )
```

```
    return MhcOps(hc_split_sinkhorn, mhc_fused_post_pre, npu_hc_pre)


# --- 后续代码中使用处示例 ---
# 在 hc_pre 函数中：
# return _get_mhc_ops().npu_hc_pre(...)
# 在 forward 函数中：
# pre, post, comb = _get_mhc_ops().hc_split_sinkhorn(...)
# residual, post, comb, hidden_states = _get_mhc_ops().mhc_fused_post_pre(...)
```

python/sglang/srt/distributed/device_communicators/cuda_wrapper.py

重写 find_loaded_library 函数：收集所有候选路径，去除 (deleted) 后缀，并优先返回不含 "stub" 的路径（即真正的 CUDA 运行时），避免 TileLang 的 libcudart_stub.so 被错误选中。

```
# -*- coding: utf-8 -*-
# --- 仅展示重写后的 find_loaded_library 函数 ---

def find_loaded_library(lib_name) -> Optional[str]:
    """
    根据 /proc/self/maps 查找已加载的共享库路径。
    返回第一个不包含 "stub" 的候选路径（若存在），否则返回第一个候选。
    返回 ``None`` 表示该库未被加载。
    """
    candidates = []
    with open("/proc/self/maps") as f:
        for line in f:
            # 过滤掉不包含库名或不含路径的行
            if lib_name not in line or "/" not in line:
                continue
            path = line[line.index("/"):].strip()
            # 如果路径末尾有 " (deleted)" 后缀，剥离它
            if path.endswith(" (deleted)"):
                path = path[:-len(" (deleted)")]
            filename = os.path.basename(path)
            # 确保文件名以 lib_name 开头
            if filename.rpartition(".so")[0].startswith(lib_name):
                candidates.append(path)

    if not candidates:
        return None

    # TileLang 会提供一个 libcudart_stub.so，该库仅能满足其 JIT 加载器的需要。
    # 选择该库会导致 CUDA IPC 和 FlashInfer all-reduce 在解析符号时失败。
    # 因此优先返回不包含 "stub" 的路径（真正的 CUDA 运行时）。
    for path in candidates:
        if "stub" not in os.path.basename(path):
            return path
    # 降级：所有候选都是 stub 时返回第一个（兼容极端场景）
    return candidates[0]
```

评论区精华

Review 中 gemini-code-assist[bot] 提出了两个关键改进建议：

- 处理 (deleted) 后缀：当共享库在运行中被删除时，/proc/self/maps 会追加 (deleted)，直接使用路径会导致 ctypes.CDLL 加载失败。建议在提取路径后剥离该后缀。最终实现采纳了此建议。
- 使用 NamedTuple 代替原始元组：建议使用 NamedTuple 或类封装返回的函数，避免下游使用魔法数字索引（如 _get_mhc_ops()[0]），提高可读性和安全性。最终实现采纳并添加了 MhcOps NamedTuple。
  - 处理 /proc/self/maps 中已删除库的 (deleted) 后缀 (correctness): 已采纳，在 find_loaded_library 中添加 if path.endswith(' (deleted)'): 处理。
  - 使用 NamedTuple 替代原始元组提高代码可读性和安全性 (design): 已采纳，创建了 MhcOps NamedTuple，并在调用处使用命名属性。

风险与影响

- 风险：
  1. 如果系统仅有 TileLang stub 可用（比如在 TileLang 专用测试环境），find_loaded_library 会回退到 stub，但这与之前的逻辑一致，不会恶化。
  2. 延迟导入改变了 MHC 内核的加载时机，可能增加首次推理的延迟，但 functools.cache 确保后续调用无额外开销。
  3. 对 /proc/self/maps 的解析逻辑修改，可能影响其他使用 find_loaded_library 的组件。整体风险较低，已有测试覆盖核心路径。 - 影响：对用户：修复了多模型（如 Qwen3-VL + DeepSeek-V4）混合部署时的启动失败问题，提升了部署稳定性。对系统：DeepSeek-V4 模型首次推理时会有一次额外的导入开销，但可忽略不计。对团队：提供了清晰的延迟导入模式，可指导其他类似 CUDA 运行时隔离的需求。影响范围主要涉及使用 DeepSeek-V4 且环境中包含 TileLang 的用户。 - 风险标记：CUDA 库选择逻辑变更，模块导入时机变更

关联脉络

- PR #30874 Bump TileLang version to fix lib naming issue: 同一作者在后续 PR 中更新 TileLang 版本以修复其库命名问题，与本 PR 的 CUDA stub 隔离相辅相成。PR 评论中直接提及。