PR #2397 完整报告

radixark/miles

fix(fsdp): move the AMD Triton attention bridge in-tree

合并时间：2026-08-14 12:34

原文链接：http://prhub.com.cn/radixark/miles/pull/2397

执行摘要

- 一句话：树内化 AMD Triton 注意力桥，修复 ROCm FSDP 导入失败
- 推荐动作：值得精读。该 PR 是一个教科书式的"构建期隐蔽故障根治"案例：先定位根因（零上下文补丁按行号落点 + --unidiff-zero 关闭保护 + 3way 回退失效），再选择删除 overlay 而非重新生成补丁，并用构建期导入检查与 CPU 单元测试双保险防止复发。阅读重点可放在 _build_model_with_attn_bridge 的 gating 逻辑、ALL_ATTENTION_FUNCTIONS 注册模式，以及测试中通过 object.__new__ 隔离 actor 的构造技巧。

功能与动机

PR body 指出 import miles.backends.fsdp_utils 在 rocm/sgl-dev:miles-rocm720-mi35x-20260811 上直接抛 IndentationError，根因是 docker/amd_patch/latest/miles.patch 的零上下文插入 hunk 按行号落点，而 actor.py 早已漂移；git apply --unidiff-zero 关闭了上下文保护，且 3way 回退所需的 pre-image blob 不在仓库中，导致构建步骤永远静默退出 0。ROCm CI 又在镜像之上安装 PR checkout，所以该问题从未被 CI 发现。作者据此选择删除 overlay 而不是重新生成补丁，并引用 Dockerfile.rocm 中已有的 TODO："remove these patches once the changes are merged into the main codebase."

实现拆解

1. 移除覆盖层：删除 docker/amd_patch/latest/miles.patch 与 docker/amd_patch/latest/sglang_attn_bridge/ 目录，同步从 docker/Dockerfile.rocm 移除 git apply --unidiff-zero 步骤；仓库中该 flag 的唯一使用者随之消失。
2. 桥接包进树：将 sglang_attn_bridge 下的 __init__.py、hf_sglang_triton_patch.py、triton_attn_bwd.py 原样重命名到 miles/backends/fsdp_utils/sglang_attn_bridge/，保持 git 重命名历史，内容不改，这样 actor.py 的相对导入归属与包所在位置一致。
3. 新增统一构建入口：在 miles/backends/fsdp_utils/actor.py 的 FSDPTrainRayActor 中新增 _build_model_with_attn_bridge，主模型 init() 与 _create_ref_model() 两个调用点全部改走该方法；启用条件收紧为 attn_implementation == "triton" and torch.version.hip is not None，ROCm 上先以 eager 构建，再通过 ALL_ATTENTION_FUNCTIONS 注册并选中桥；相较原补丁删除了死赋值 self._use_triton_bridge，并将无保护作用的 getattr 改为直接属性访问。非 ROCm 分支等价于原先的 from_pretrained 透传。

4. 补测试与构建期防复发：新增 tests/fast/backends/test_fsdp_attn_bridge.py，用 _RecordingModelCls 桩记录 from_pretrained 收到的参数，覆盖三个分支：非 triton 透传不变、非 ROCm 忽略 triton、ROCm 时 *eager* 构建后激活桥；同时在 docker/Dockerfile.rocm 中加入安装后导入 miles.backends.fsdp_utils（并沿用已有 import sglang; import sgl_kernel 检查模式）的构建检查，让坏树在镜像构建当天就以非零退出暴露。

关键文件：

- miles/backends/fsdp_utils/actor.py（模块 FSDP 后端；类别 source；类型 *core-logic*；符号 _build_model_with_attn_bridge）：FSDP 后端唯一发生行为变更的源码文件，新增 _build_model_with_attn_bridge 并统一主模型与 *ref* 模型的构建路径，是本次修复的核心。
- tests/fast/backends/test_fsdp_attn_bridge.py（模块 桥接测试；类别 test；类型 *test-coverage*；符号 _StubModel, _RecordingModelCls, _actor, test_non_triton_attn_implementation_is_passed_through_unchanged）：首次为桥接启用逻辑补齐测试，覆盖非 triton 透传、非 ROCm 忽略、ROCm 时 *eager* 构建后激活桥三路分支，防止未来回归。
- docker/amd_patch/latest/miles.patch（模块 AMD 补丁；类别 test；类型 deletion）：破坏源头：零上下文补丁错落导致 *actor.py* 语法错误；删除它同时移除了仓库中 *--unidiff-zero* 的唯一使用。
- miles/backends/fsdp_utils/sglang_attn_bridge/__init__.py（模块 注意力桥；类别 source；类型 *rename-or-move*）：桥接包随修复进树，纯重命名保留内容与 *git* 历史，供 *actor.py* 相对导入。
- miles/backends/fsdp_utils/sglang_attn_bridge/hf_sglang_triton_patch.py（模块 注意力桥；类别 source；类型 *rename-or-move*）：桥接核心实现文件，包含 apply_sglang_triton_attention_patch，随包整体移动。
- miles/backends/fsdp_utils/sglang_attn_bridge/triton_attn_bwd.py（模块 注意力桥；类别 source；类型 *rename-or-move*）：桥接反向 Triton kernel 所在文件，随包整体移动，无内容变化。
- docker/Dockerfile.rocm（模块 镜像构建；类别 infra；类型 infrastructure）：移除 *git apply --unidiff-zero* 补丁应用步骤，并新增安装后导入 miles.backends.fsdp_utils 的构建期检查，防止同类坏镜像再次发布。

关键符号：_build_model_with_attn_bridge, test_non_triton_attn_implementation_is_passed_through_unchanged, test_triton_is_ignored_off_rocm, test_triton_on_rocm_loads_eager_then_activates_the_bridge

关键源码片段

tests/fast/backends/test_fsdp_attn_bridge.py

首次为桥接启用逻辑补齐测试，覆盖非 triton 透传、非 ROCm 忽略、ROCm 时 *eager* 构建后激活桥三路分支，防止未来回归。

```
# tests/fast/backends/test_fsdp_attn_bridge.py —— 桥接启用条件的三路分支测试
from contextlib import nullcontext
```

```
from types import SimpleNamespace

import torch

from miles.backends.fsdp_utils.actor import FSDPTrainRayActor


# 桩模型：只带 config 字段，不真正加载 checkpoint
class _StubModel(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.config = SimpleNamespace()


# 记录 from_pretrained 收到的参数，用于断言透传行为
class _RecordingModelCls:
    def __init__(self):
        self.kwargs = None

    def from_pretrained(self, checkpoint_path, **kwargs):
        self.kwargs = dict(kwargs, checkpoint_path=checkpoint_path)
        return _StubModel()


# 用 object.__new__ 绕过 __init__，只构造测试所需字段，隔离桥接逻辑
# 与 FSDPTrainRayActor 完整的初始化流程。
def _actor(attn_implementation, model_cls):
    actor = object.__new__(FSDPTrainRayActor)
    actor.args = SimpleNamespace(attn_implementation=attn_implementation)
    actor.get_model_cls = lambda: model_cls
    return actor


def test_non_triton_attn_implementation_is_passed_through_unchanged():
    model_cls = _RecordingModelCls()
    actor = _actor("flash_attention_2", model_cls)

    _, patched = actor._build_model_with_attn_bridge("/ckpt", nullcontext)

    # 非 triton 值必须原样进入 from_pretrained，不能被桥截获
    assert model_cls.kwargs["attn_implementation"] == "flash_attention_2"
    assert patched == 0


def test_triton_is_ignored_off_rocm(monkeypatch):
    monkeypatch.setattr(torch.version, "hip", None)
    model_cls = _RecordingModelCls()
    actor = _actor("triton", model_cls)
```

```
    _, patched = actor._build_model_with_attn_bridge("/ckpt", nullcontext)

    # 非 ROCm 上 triton 原样交给 from_pretrained，由后者拒绝，保持旧行为
    assert model_cls.kwargs["attn_implementation"] == "triton"
    assert patched == 0


def test_triton_on_rocm_loads_eager_then_activates_the_bridge(monkeypatch):
    from transformers.modeling_utils import ALL_ATTENTION_FUNCTIONS

    monkeypatch.setattr(torch.version, "hip", "6.0.0")
    previous = ALL_ATTENTION_FUNCTIONS.pop("triton", None)
    try:
        model_cls = _RecordingModelCls()
        actor = _actor("triton", model_cls)

        model, _ = actor._build_model_with_attn_bridge("/ckpt", nullcontext)

        # ROCm 上以 eager 构建，随后桥注册 triton 并把它选回生效实现
        assert model_cls.kwargs["attn_implementation"] == "eager"
        assert "triton" in ALL_ATTENTION_FUNCTIONS
        assert model.config._attn_implementation == "triton"
    finally:
        if previous is None:
            ALL_ATTENTION_FUNCTIONS.pop("triton", None)
        else:
            ALL_ATTENTION_FUNCTIONS["triton"] = previous
```

评论区精华

XinyuJiangCMU 在 PR 评论中给出结论性意见：“Looks good to me. The true on-policy path can be handled in a follow-up PR.”，作者 Arist12 回应感谢并请 Zhichenzzz 安排 review，最终 Zhichenzzz 以 APPROVED 收尾。PR body 还明确了一条边界讨论：桥接 forward 调用的 sglang.srt.layers.attention.triton_ops.extend_attention 已迁移到 sglang.kernels.ops.attention.extend_attention 并增加了 k_scale/v_scale 参数，该端到端 triton 路径在 pinned sglang-miles 分支上本来就是坏的，修复需要独立的前向 / 反向验证，作者明确留在后续 PR。

- 真正 on-policy 路径留待后续 PR (design): 本 PR 仅恢复 ROCm 镜像的 import 与 FSDP 后端可用性；--attn-implementation triton 的端到端修复不在范围，留作后续 PR。

风险与影响

- 风险： 1) 运行时风险：ROCm 构建期只检查 import，不校验 --attn-implementation triton 的运行时路径；由于 sglang extend_attention 的 API 已迁移且签名变化，triton 桥在运行时仍可能失败，作者已声明不在本 PR 范围，需要后续 forward/backward 验证。2) 逻辑耦合风险：桥接启用依赖 self.args.attn_implementation 与 torch.version.hip 的组合判断，若未来参数名或设备探测方式变化，该分支可能偏移；新增测试覆盖了三路分支

，但未覆盖 apply_sglang_triton_attention_patch 内部行为（该函数本身无测试）。 3) 构建期回归风险：docker/Dockerfile.rocm 移除补丁应用步骤后，若未来再出现对旧 docker/amd_patch/latest 路径的引用会直接失败；目前仓库内已无其他引用，但该路径删除属于不可逆变更。 4) 非 ROCm 影响为零：桥接模块全部在被 gated 的分支内惰性导入，from_pretrained 对非 triton 值走原路径，因此 CUDA 平台行为不变。

- 影响：用户侧：ROCm 发布镜像恢复 import miles.backends.fsdp_utils，FSDP 后端在 flash_attention_2、sdpa、eager 三种 attention 实现上可用，解决了镜像开箱即坏的问题。系统侧：仓库中唯一的零上下文补丁与 --unidiff-zero 用法被移除，消除了一整类“构建静默成功但安装后语法错误”的风险；构建期导入检查把故障前移到镜像构建阶段。团队侧：AMD Triton 注意力桥从 docker overlay 收编进源码树，代码归属与导入关系一致，便利后续维护与 review。
- 风险标记：构建期回归风险，triton 运行时路径未做端到端验证，桥接模块内部行为缺少测试

关联脉络

- PR #2670 fix(docker): update torch_memory_saver for CUDA VMM granularity: 同属 Dockerfile / Dockerfile.rocm 镜像依赖修复线，说明 ROCm 镜像维护是连续作战，与本 PR 消除镜像级故障的目标一致。
- PR #2660 [AMD] Point the ROCm image at the v0.5.16 wheels release with TE 2.17.0: 同改 Dockerfile.rocm 并涉及 AMD 脚本，是 ROCm 镜像演进路径的一部分；本 PR 移除的 overlay 正是这些构建环节中的一个隐患。
- PR #2600 fix(docker): pin cutlass-dsl 4.6.2 and flashinfer 0.6.15.post1 over the sglang base: 同样是修复构建期隐性问题的 Docker 层变更，通过 pin 依赖解决训练挂起，与本 PR 通过移除错误补丁解决导入崩溃同属镜像质量收口。