PR #7241 完整报告

verl-project/verl

[megatron, hardware] fix: pure-torch fast_hadamard_transform fallback for DSA on ROCm

合并时间：2026-08-04 17:22

原文链接：http://prhub.com.cn/verl-project/verl/pull/7241

执行摘要

- 一句话：ROCm 下 DSA 增加纯 torch 哈达玛变换回退
- 推荐动作：值得精读 verl/models/mcore/patch.py 中 apply_fast_hadamard_transform_shim 的实现：它展示了一种“模块名级兼容垫片 + 已捕获符号回填”的模式，对处理第三方包硬编码 import 的硬件移植问题很有参考价值。建议在合入后补充针对回填逻辑的单元测试，并在 ROCm CI 上增加 DeepSeek-V4 DSA 的冒烟用例。

功能与动机

DeepSeek sparse attention (DSA) 在 Megatron-LM 中通过 from fast_hadamard_transform import hadamard_transform 在 import 时绑定符号，并在 indexer 的 rotate_activation 中断言其不为 None。Dao-AILab 的包只提供 nvcc 内核且 setup.py 硬性要求 nvcc，ROCm 上无法构建，导致导入结果为 None、每次 DSA 前向崩溃。本 PR 需要在不依赖该 CUDA 包的情况下让 ROCm 上的 DSA 可用。

实现拆解

1. 实现纯 PyTorch 的 FWHT：在 verl/models/mcore/patch.py 中新增 pure_torch_hadamard_transform(x, scale)，采用向量化蝴蝶算法，在 fp32 下累积，最终保证与 F.linear(x, scipy.linalg.hadamard(dim)) * scale 数值等价、与 Dao-AILab CUDA kernel 位级一致。该函数对最后维度做 2 的幂校验，并恢复原始 dtype 与 shape。
2. 注入同名模块：apply_fast_hadamard_transform_shim 先通过 importlib.import_module("fast_hadamard_transform") 探测真实包，成功即返回；探测失败（包括 OSError 这类非 ImportError 的情况）时在 sys.modules 中注册一个 types.ModuleType 的 stub，并将 pure_torch_hadamard_transform 赋给 hadamard_transform 属性，使后续任何 from fast_hadamard_transform import ... 都能命中。
3. 回填已绑定 None 的导入者：因为 Megatron 的 dsa.py 在 import 时已执行绑定并得到 None，需要遍历 sys.modules，直接修改捕获了 None 的模块 __dict__；刻意避开 getattr 以避免触发 PEP-562 的模块级 __getattr__ 钩子（可能 raise 或触发惰性加载）。
4. 接入统一 patch 入口：apply_patch() 在开头调用 shim，确保无论模型创建前（hf_to_mcore_config_dpskv3）还是创建后（mbridge 路径）都完成注入，避免引擎层重复接线。

5. 配套与测试：verl/workers/engine/megatron/delta_export.py 中一处 assert 语句压缩为单行（纯格式变化）；PR 未在仓库内添加正式测试文件，作者在评论中提供了 H200 上对比 pure-torch 与 CUDA 结果的临时脚本 test_mcore_hadamard_transform.py。

关键文件：

- verl/models/mcore/patch.py（模块 模型补丁；类别 source；类型 core-logic；符号 pure_torch_hadamard_transform, apply_fast_hadamard_transform_shim）：核心变更文件：新增 pure_torch_hadamard_transform 与 apply_fast_hadamard_transform_shim，并在 apply_patch() 入口接入，解决 ROCm 上 DSA 因 fast_hadamard_transform 不可用而崩溃的问题。
- verl/workers/engine/megatron/delta_export.py（模块 增量导出；类别 source；类型 style）：仅将 assert 语句由多行压缩为单行，纯格式调整，无行为变化，但作为伴随改动一并合入。

关键符号：pure_torch_hadamard_transform, apply_fast_hadamard_transform_shim

关键源码片段

verl/models/mcore/patch.py

核心变更文件：新增 pure_torch_hadamard_transform 与 apply_fast_hadamard_transform_shim，并在 apply_patch() 入口接入，解决 ROCm 上 DSA 因 fast_hadamard_transform 不可用而崩溃的问题。

```
def pure_torch_hadamard_transform(x, scale=1.0):
    # Fast Walsh-Hadamard 变换：向量化蝴蝶算法，沿最后维度计算，要求长度为 2 的幂
    import torch

    n = x.shape[-1]
    if n < 1 or n & (n - 1) != 0:
        raise ValueError(f"hadamard_transform requires last dim to be a power of 2, got {n}")

    orig_dtype = x.dtype
    orig_shape = x.shape
    # 提升到 fp32 累积，保证与 CUDA kernel 位级一致，结束时再恢复 dtype
    y = x.to(torch.float32).reshape(-1, n)
    h = 1
    while h < n:
        y = y.view(-1, n // (2 * h), 2, h)
        a = y[:, :, 0, :]
        b = y[:, :, 1, :]
        # 蝶形核心：每对输入拆成 a+b 与 a-b，交换位置后继续下一级
        y = torch.stack((a + b, a - b), dim=2).reshape(-1, n)
        h *= 2
    y = y * scale
    return y.reshape(orig_shape).to(orig_dtype)


def apply_fast_hadamard_transform_shim():
```

```
# 在真实 CUDA 包不可用时，用纯 torch 实现接管 fast_hadamard_transform
# DSA 在 Megatron-LM 中于 import 时绑定该符号，而 Dao-AILab 的包在
# ROCm 上无法构建（setup.py 硬依赖 nvcc），导入结果为 None
import importlib
import logging
import sys
import types

# 捕获所有异常（包括 OSError）：其他工具链编译的扩展同样可能加载失败；
# 一旦真实包可导入，所有 importer 都会解析到同一模块，无需注入
try:
    importlib.import_module("fast_hadamard_transform")
    return
except Exception:
    pass

# 注册 stub 模块到 sys.modules，使后续任何 from ... import ... 都能命中
module = types.ModuleType("fast_hadamard_transform")
module.hadamard_transform = pure_torch_hadamard_transform
sys.modules["fast_hadamard_transform"] = module

# 回填已捕获 None 的导入者（如 Megatron 的 dsa.py）：
# 直接改 __dict__，避免 getattr 触发 PEP-562 的模块级 __getattr__ 钩子
for mod in list(sys.modules.values()):
    mod_dict = getattr(mod, "__dict__", None)
    if mod_dict is not None and mod_dict.get("hadamard_transform", "keep") is None:
        mod_dict["hadamard_transform"] = pure_torch_hadamard_transform

logging.getLogger(__name__).warning(
    "fast_hadamard_transform is unavailable; falling back to a pure-torch Fast "
    "Walsh-Hadamard transform. Results match the CUDA kernel but DSA forward will be
    slower."
)
```

评论区精华

核心讨论集中在 apply_fast_hadamard_transform_shim 的探测逻辑：wuxibin89 首先指出原实现中判断包是否可用的 predictor 逻辑不正确（“This predictor is not right.”）；PeterYang12 随后重构为用 importlib 实际探测包安装情况，并在 H200 上验证纯 PyTorch 与 CUDA 输出一致，附测试脚本 test_mcore_hadamard_transform.py。wuxibin89 还追问“如果 fast_hadamard_transform 已存在是否应提前 return”，最终实现采用导入成功即返回的写法，确保 stub 不会覆盖真实包。

- 包可用性探测逻辑的正确性 (correctness): 重构为 importlib.import_module 实际探测，成功则直接 return，避免猜测式判断。
- 真实包存在时是否提前返回 (design): 最终实现采用 importlib.import_module 成功即返回的写法，stub 只会在真实包不可导入时安装。

风险与影响

- 风险：主要风险在三个方面：其一，sys.modules 全局注入是进程级副作用，虽然实现把真实包探测放在首位、只在导入失败时注入，但若运行期后续安装真实包，stub 可能阻止其生效，需要依赖环境部署顺序；其二，纯 torch 蝴蝶变换在长序列上性能低于 CUDA kernel，ROCm 上的 DSA 前向会有明显回退，代码中已通过 warning 日志提示；其三，缺少正式单测覆盖注入回填逻辑与数值等价性，回归保障依赖 ROCm CI。至于 *delta_export.py* 的改动仅是 assert 格式单行化，无行为影响。
- 影响：影响范围：修复 AMD/ROCm 平台上 Megatron 后端 DeepSeek-V4/V3.2 等使用 DSA 的模型训练 / 推理崩溃，使 ROCm 成为 DSA 的可用硬件路径；CUDA 用户完全无感（真实包存在时 shim 直接返回）。对团队而言，需要补充 ROCm DSA 的 CI 覆盖，并在升级 Megatron 或 Dao-AILab 包时关注该 shim 的兼容性。
- 风险标记：全局 sys.modules 注入，缺少正式测试覆盖，fallback 性能回退，影响 DSA 导入路径

关联脉络

- PR #7050 （PR body 引用的来源改动，具体标题未包含在本次上下文中）：PR body 明确说明本 PR 是从 #7050 中拆出，以便独立评审其中的 ROCm Dockerfile 与 FP8/rollout 改动。
- PR #7297 [megatron] fix: make DeepSeek-V4 context parallelism actually runnable: 同为打通 DeepSeek-V4 Megatron 可运行性的修复，涉及 mcore patch / DSA 相关路径，方向一致。