PR #30212 完整报告

sgl-project/sglang

[AMD] Register 3 ROCm-portable JIT kernel tests for AMD CI

合并时间：2026-07-09 04:38

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30212

执行摘要

- 一句话：为 AMD CI 注册 3 个 JIT 内核测试
- 推荐动作：本 PR 结构清晰，采用了合理的跨平台测试设计模式，值得阅读。特别关注 reference_* 包装器架构和 is_hip() 用法，可作为后续 AMD 测试适配的参考。

功能与动机

由于大部分 JIT 内核包含 CUDA 特有代码无法在 ROCm 上编译，本 PR 将恰好三个经过验证可在 ROCm 上通过的测试注册到 AMD CI，以提供基本的 JIT 内核回归保护。

实现拆解

1. 在三个测试文件中导入 register_amd_ci 并调用 register_amd_ci(est_time=..., suite="jit-kernel-unit-test-amd")。
2. 在 test_rmsnorm.py 中添加纯 PyTorch 的 torch_rmsnorm 实现作为 ROCm 参考，并创建 reference_rmsnorm 包装器根据 is_hip() 选择 flashinfer 或 torch 路径。
3. 在 test_rope.py 中实现之前的存根函数 torch_impl_rope（包含辅助函数 _rope_rotate），并创建 reference_rope 包装器进行 is_hip() 分发。
4. 将测试中的参考调用从 flashinfer_改为 reference_，使 AMD 和 NVIDIA 都能正确验证。

关键文件：

- test/registered/jit/test_rope.py（模块 RoPE；类别 test；类型 test-coverage；符号 _rope_rotate, reference_rope）：实现了 _rope_rotate 和 reference_rope 纯 PyTorch 参考，添加 AMD CI 注册，是本次变更的核心文件之一。
- test/registered/jit/test_rmsnorm.py（模块 RMSNorm；类别 test；类型 test-coverage；符号 torch_rmsnorm, reference_rmsnorm）：添加了 torch_rmsnorm 和 reference_rmsnorm 纯 PyTorch 参考实现，以及 AMD CI 注册。
- test/registered/jit/test_dsv32_indexer_fusion.py（模块 DSA 索引器融合；类别 test；类型 test-coverage）：仅添加 register_amd_ci 调用和导入修改，但该测试本身已有 is_hip() 守卫且可于 AMD 运行。

关键符号：_rope_rotate, reference_rope, torch_impl_rope, torch_rmsnorm, reference_rmsnorm

关键源码片段

test/registered/jit/test_rope.py

实现了 _rope_rotate 和 reference_rope 纯 PyTorch 参考，添加 AMD CI 注册，是本次变更的核心文件之一。

```
def _rope_rotate(x: torch.Tensor, cos: torch.Tensor, sin: torch.Tensor, is_neox: bool):
    """Rotate the first ``rotary_dim`` channels of ``x`` in place.

    ``x``: [nnz, num_heads, head_size]; ``cos``/``sin``: [nnz, rotary_dim // 2].
    Matches flashinfer's ``apply_rope_with_cos_sin_cache_inplace`` convention:
    NeoX splits the rotary block into halves; non-NeoX (GPT-J) uses interleaved
    even/odd pairs. Channels beyond ``rotary_dim`` are left untouched.
    """
    rotary_dim = cos.shape[-1] * 2
    xf = x[..., :rotary_dim].to(torch.float32)
    cos = cos[:, None, :] # [nnz, 1, rotary_dim // 2]
    sin = sin[:, None, :]
    if is_neox:
        x1, x2 = xf[..., : rotary_dim // 2], xf[..., rotary_dim // 2 :]
        out1 = x1 * cos - x2 * sin
        out2 = x2 * cos + x1 * sin
        rotated = torch.cat((out1, out2), dim=-1)
    else:
        x1, x2 = xf[..., 0::2], xf[..., 1::2]
        out1 = x1 * cos - x2 * sin
        out2 = x2 * cos + x1 * sin
        rotated = torch.stack((out1, out2), dim=-1).flatten(-2)
    x[..., :rotary_dim] = rotated.to(x.dtype)


def torch_impl_rope(q, k, cos_sin_cache, positions, is_neox):
    """Pure-PyTorch RoPE reference (in place), used as the ROCm fallback."""
    rotary_dim = cos_sin_cache.shape[-1]
    half = rotary_dim // 2
    gathered = cos_sin_cache[positions.long()]
    cos, sin = gathered[:, :half], gathered[:, half:]
    _rope_rotate(q, cos, sin, is_neox)
    _rope_rotate(k, cos, sin, is_neox)


def reference_rope(q, k, cos_sin_cache, positions, is_neox):
    # NVIDIA uses flashinfer (the reference); flashinfer is CUDA-only, so on
    # ROCm fall back to the torch reference (matches flashinfer's cos/sin-cache
    # application semantics).
    if is_hip():
        torch_impl_rope(q, k, cos_sin_cache, positions, is_neox)
    else:
        flashinfer_rope(q, k, cos_sin_cache, positions, is_neox)
```

test/registered/jit/test_rmsnorm.py

添加了 torch_rmsnorm 和 reference_rmsnorm 纯 PyTorch 参考实现，以及 AMD CI 注册。

```
def torch_rmsnorm(input, weight, *, output, eps=EPS):
    x = input.float()
    normed = x * torch.rsqrt(x.pow(2).mean(dim=-1, keepdim=True) + eps)
    output.copy_((normed * weight.float()).to(output.dtype))


def reference_rmsnorm(input, weight, *, output, eps=EPS):
    # NVIDIA uses flashinfer (the bitwise reference); flashinfer is CUDA-only,
    # so on ROCm fall back to the torch reference (matches flashinfer math).
    if is_hip():
        torch_rmsnorm(input, weight, output=output, eps=eps)
    else:
        flashinfer_rmsnorm(input, weight, output=output, eps=eps)
```

评论区精华

- 提取 reference 包装器（kpham-sgl）：建议不要直接在 flashinfer_函数内添加 is_hip() 分支，而是创建 reference_ 包装器，将 AMD 和 NVIDIA 路径分离。作者采纳并在第二次 commit 中实现。
- test_dsv32_indexer_fusion 数值失败：该测试在 NVIDIA H100 上存在一个不依赖于本 PR 的数值不匹配问题（可能与 #29613 的 DSA indexer fusion 相关），属已有 flaky 测试。
- 提取 reference_* 包装器 (design): 作者采纳，在第二个 commit 中实现 reference_rmsnorm 和 reference_rope，将 is_hip() 分发封装到包装器中。
- test_dsv32_indexer_fusion 数值失败 (correctness): kpham-sgl 同意该测试 flaky，但本 PR 安全可合并。

风险与影响

- 风险：
  1. NVIDIA 路径不变：所有 is_hip() 守卫保证 NVIDIA 仍使用原始 flashinfer 参考，无回归风险。
  2. 新增参考精度：纯 PyTorch 参考实现可能不完全匹配 flashinfer 的位精确结果，但现有断言 atol=1e-2, rtol=1e-2 容差足够。
  3. 现有 flaky 测试：test_dsv32_indexer_fusion 在 NVIDIA 上已有数值问题，但本 PR 未修改该测试内核，不会加剧。AMD CI 上该测试正常通过。 - 影响：对 AMD 用户：在 AMD CI 中增加了 JIT 内核回归测试覆盖，提高了平台稳定性。 对 NVIDIA 用户：无影响，测试逻辑完全不变。 对开发者：提供了一个良好的跨平台测试模式（reference wrapper + is_hip() 分发），可推广到其他测试。
- 风险标记：测试环境依赖变化，数值精度风险，现有 flaky 测试

关联脉络

- PR #30446 [AMD] Register 2 CPU-bound 1-GPU tests (phase_checker, scripted_runtime_core) for AMD PR CI: 同一作者（michaelzhang-ai）在同一领域（注册 AMD 测试到 CI）的工作，共享 register_amd_ci 机制和 CI 编排模式。