PR #25208 完整报告

sgl-project/sglang

[AMD] ci: register 5 framework tests to run on AMD CI

合并时间：2026-05-17 13:56

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25208

执行摘要

- 一句话：AMD CI 注册 5 个框架测试，缩小覆盖差距
- 推荐动作：本 PR 虽为 CI 配置变更，但其筛选策略和 Reviewer 反馈处理的思路值得关注：1）通过关键字过滤排除硬件相关测试，避免虚假失败；2）对于平台特有差异，使用 skipIf(is_hip()) 精确跳过，而非全局禁用；3）坚持注册的测试必须可运行，避免注册即禁用。推荐 CI 维护者和跨平台贡献者阅读。

功能与动机

缩小 AMD CI 与 NVIDIA CI 之间的测试覆盖差距。PR body 指出 'On main, ~200 tests are registered for CUDA via register_cuda_ci(...) but not for AMD.' 本 PR 挑选首批无 NVIDIA 硬件依赖的框架测试注册到 AMD CI，实现每次提交时的 AMD CI 信号。

实现拆解

实现过程分为 4 步：

1. 筛选候选测试：从 197 个 CUDA-only 测试中过滤掉含有 flashinfer、trtllm、fa3、cutlass、nvfp4 等 NVIDIA 关键字的文件，仅保留无硬件依赖的框架测试，并限定在 stage-b 1-gpu-small/large 套件内以避免工作流变更。
2. 注册 AMD CI：在 5 个测试文件中添加 register_amd_ci(est_time=..., suite=...) 行，指定预计执行时间和 AMD 套件名，与现有的 register_cuda_ci(...) 并列。
3. 处理 AMD 失败：为 test_pooled_hidden_states.py 的 TestPooledHiddenStatesMISEngine 类添加 @unittest.skipIf(is_hip(), ...) 跳过（因依赖 flashinfer 后端，NVIDIA-only）；为 test_session_control.py 的 test_session_control_with_branching 方法添加跳过（在 AMD 上产生确定性 1-token 偏差，怀疑 ROCm 数值精度差异）。
4. 根据 Review 反馈精简：移除最初注册但需要全禁用的 3 个测试（test_serving_transcription.py、test_session_latency.py、test_streaming_session_swa.py）的 AMD 注册行，使所有注册的测试都真正在 AMD 上执行。最终 5 个文件均通过 AMD CI 实测验证。

配套文件：所有变更均为测试文件，无核心逻辑修改。

关键文件：

- test/registered/sessions/test_session_control.py（模块 会话控制；类别 test；类型 test-coverage；符号 test_session_control_with_branching）：主要变更文件：添加 register_amd_ci 注册 AMD CI，并因 ROCm 精度差异跳过一个测试方法，是展示平台差异处理的典型示例。
- test/registered/prefill_only/test_pooled_hidden_states.py（模块 池化隐层；类别 test；类型 test-coverage；符号 TestPooledHiddenStatesMISEngine）：添加 AMD 注册并对依赖 flashinfer 的 MIS 测试类整体跳过，是处理 NVIDIA-only 依赖的典型方案。
- test/registered/core/test_engine_child_pids.py（模块 子进程；类别 test；类型 test-coverage）：纯注册文件，无 skip，展示最基本的 AMD 注册模式。
- test/registered/observability/test_tracing.py（模块 追踪；类别 test；类型 test-coverage）：纯注册文件，无 skip，验证框架追踪模块在 AMD 上的执行。
- test/registered/sessions/test_streaming_session.py（模块 流式会话；类别 test；类型 test-coverage）：纯注册文件，无 skip，测试流式会话功能在 AMD 上的覆盖。

关键符号：test_session_control_with_branching, TestPooledHiddenStatesMISEngine

关键源码片段

test/registered/sessions/test_session_control.py

主要变更文件：添加 register_amd_ci 注册 AMD CI，并因 ROCm 精度差异跳过一个测试方法，是展示平台差异处理的典型示例。

```
"""
Usage:
python3 -m unittest test_session_control.TestSessionControl.test_session_control
...
"""
import asyncio
import json
import unittest

import aiohttp
import requests

from sglang.srt.utils import is_hip, kill_process_tree # 新增 is_hip 用于 AMD 平台检测
from sglang.srt.utils.hf_transformers_utils import get_tokenizer
from sglang.test.ci.ci_register import register_amd_ci, register_cuda_ci # 新增 register_amd_ci
from sglang.test.test_utils import (
    DEFAULT_SMALL_MODEL_NAME_FOR_TEST,
    DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
    DEFAULT_URL_FOR_TEST,
    CustomTestCase,
    popen_launch_server,
)

register_cuda_ci(est_time=87, stage="extra-a", runner_config="1-gpu-large")
register_amd_ci(est_time=87, suite="stage-b-test-1-gpu-large-amd") # 注册到 AMD large 套件
```

```
# ... 中间省略类定义和方法 ...

class TestSessionControl(CustomTestCase):
    # ... setUpClass, tearDownClass, test_session_control, run_session_control_with_branching ..
    .

    @unittest.skipIf(
        is_hip(), # 在 AMD ROCm 平台上跳过此方法
        "Session-branching produces a deterministic 1-token divergence from "
        "plain generation on AMD (greedy temperature=0). The other 4 of 5 "
        "branch outputs match exactly; passes on CUDA. Suspected ROCm "
        "numerical-precision difference in the session KV-cache reuse path. "
        "Re-enable when that divergence is fixed.",
    )
    def test_session_control_with_branching(self):
        root_prompt = "First, let me explain in one sentence about AI"
        chunks_per_step = [
            [
                "Then, briefly, the positive side of AI is",
                "But, briefly, AI could be harmful to human",
            ],
            ["For example", "For example"],
        ]
        self.run_session_control_with_branching(
            root_prompt=root_prompt, chunks_per_step=chunks_per_step, gen_len=8
        )
        # ... 更多测试场景 ...
```

test/registered/prefill_only/test_pooled_hidden_states.py

添加 AMD 注册并对依赖 flashinfer 的 MIS 测试类整体跳过，是处理 NVIDIA-only 依赖的典型方案。

```
"""
Tests for the return_pooled_hidden_states feature on the scoring API.
...
"""
import json
import unittest

import requests
import torch

from sglang.srt.entrypoints.engine import Engine
from sglang.srt.utils import is_hip, kill_process_tree # 新增 is_hip
from sglang.test.ci.ci_register import register_amd_ci, register_cuda_ci # 新增 register_amd_ci
from sglang.test.test_utils import (
    DEFAULT_SMALL_MODEL_NAME_FOR_TEST,
    DEFAULT_TIMEOUT_FOR_SERVER_LAUNCH,
    DEFAULT_URL_FOR_TEST,
```

```
    CustomTestCase,
    popen_launch_server,
)

register_cuda_ci(est_time=100, stage="base-b", runner_config="1-gpu-small")
register_amd_ci(est_time=100, suite="stage-b-test-1-gpu-small-amd") # 注册到 AMD small 套件

# ... 模型路径覆盖 ...

# ------------------------------------------------------------------------
# Engine — MIS delimiter mode ( 需要 flashinfer 后端，NVIDIA-only)
# ------------------------------------------------------------------------

@unittest.skipIf(
    is_hip(), # 整个类在 AMD 上跳过
    "Multi-Item Scoring (enable_mis) requires the flashinfer prefill/decode "
    "backend, which is NVIDIA-only.",
)
class TestPooledHiddenStatesMISEngine(CustomTestCase):
    """Validates return_pooled_hidden_states in MIS (delimiter) scoring mode."""
    @classmethod
    def setUpClass(cls):
        cls.engine = Engine(
            model_path=_SEQCLS_MODEL,
            disable_radix_cache=True,
            chunked_prefill_size=-1,
            enable_mis=True, # MIS 模式使用 flashinfer
            json_model_override_args=json.dumps(...),
            mem_fraction_static=0.15,
        )
    # ... 测试方法 ...
```

评论区精华

Review 中有两个关键讨论：

1. 测试注册与禁用模式：Reviewer @yctseng0211 指出 'I don't think simply disabling a test while keeping it registered is a good approach'。初始 PR 包含了 3 个注册后又因 mxfp4 等问题完全禁用的测试，reviewer 认为这样注册没有意义。作者采纳意见，移除了这三个测试的 AMD 注册，确保注册的测试都有实际执行。
2. mxfp4 在 AMD 上的兼容性：Reviewer @yctseng0211 质疑 mxfp4 在 AMD MI355 上是否已验证。作者回复已确认 mxfp4 在 MI35x 上可用，因此删除了相关测试的注册（而不是增加跳过）。

- 注册并禁用的测试管理模式 (design): 作者移除了 3 个注册后又禁用的测试的 AMD 注册行，确保所有注册的测试在 AMD 上实际执行。
- mxfp4 在 AMD 上的支持和验证 (question): 作者确认 mxfp4 在 MI35x 上可用，因此删除了相关测试的 AMD 注册（而非增加跳过），因为测试使用了 gpt-oss-20b（mxfp4 量化）。

风险与影响

- 风险：
  1. 新增测试稳定性风险：新注册的 5 个测试可能在特定 AMD GPU 型号上出现非预期失败，尤其是 streaming_session 测试预计 691 秒，可能超时。
  2. 数值精度差异：test_session_control_with_branching 已因 ROCm 精度差异跳过，但其他 test_session_control 方法可能在不同 AMD 型号上仍有潜在差异。
  3. 资源消耗：新增测试在 AMD CI 上增加约 1068 秒的执行时间，但分布在 large 和 small 套件中，总体影响可控。
  4. 无回归风险：所有变更仅影响 AMD CI 注册，不修改任何核心逻辑，CUDA CI 完全不受影响。 - 影响：对 AMD CI：增加 5 个框架测试覆盖，包括引擎子进程、追踪、池化隐藏状态、会话控制和流式会话，使 AMD CI 能在每次提交时捕获这些模块的回归。 对 NVIDIA CI：无影响，CUDA 注册行未变。 对团队：AMD 贡献者可以获得更及时的跨平台反馈；后续批次可以参考本次筛选和淘汰流程。
- 风险标记：新增测试稳定性风险，ROCM 精度差异跳过测试，AMD CI 执行时间增加

关联脉络

- PR #25260 [AMD][CI] Register Eagle constrained decoding test: 同属 AMD CI 覆盖扩展系列，本 PR 是首批框架测试注册，25260 是后续的推测解码测试注册。