PR #27644 完整报告

sgl-project/sglang

[CI] Move JIT kernel tests + benchmarks to test/registered/jit; add in-package guard

合并时间：2026-06-10 03:37

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27644

执行摘要

- 一句话：移动 JIT 内核测试并添加包内注册保护
- 推荐动作：该 PR 是一项重要的 CI 基础设施改进，建议所有涉及 JIT 内核开发的成员仔细阅读。特别是预提交钩子的设计（复用 ut_parse_one_file 确保检测逻辑一致性）值得在其他项目借鉴。建议在合并后监测一段时间的 CI 运行，确保搬迁后的测试都能正确执行。

功能与动机

原始设计将 JIT 内核测试和基准放在可导入包内部，导致它们被包含在 wheel 中（浪费空间）且 run_suite.py 需要额外 glob 才能发现。一旦 glob 与被转移测试不同步，测试可能被静默忽略。搬迁到 test/registered/ 后，测试统一通过 standard glob 发现，并通过新增的预提交钩子将'静默跳过'转为硬失败，主动防御未来出现同样问题。

实现拆解

步骤：

1. 搬迁测试和基准：将 python/sglang/jit_kernel/{tests,benchmark}/ 下的所有 test_*.py 和 bench_*.py（共 105 个文件）分别移动到 test/registered/jit/ 和 test/registered/jit/benchmark/，保持子目录结构（kv_canary/、deepseek_v4/、diffusion/）。辅助模块（如 tests/utils.py）保留在包内，搬迁后的文件通过绝对导入引用。
2. 修复路径依赖：对定位 csrc/ 头文件或仓库根目录的测试，改用已安装包的路径锚定。例如 test_const_sync.py 中 _CONSTS_CUH 从 Path(__file__).resolve().parents[2] 改为 Path(sglang.jit_kernel.__file__).resolve().parent。
3. 更新 run_suite.py：删除原用于收集 python/sglang/jit_kernel/ 下测试和基准的额外 glob（8 行代码），所有文件统一通过 test/registered/ 下的 glob 收集。
4. 添加预提交保护：新增 scripts/ci/check_no_registered_tests_in_package.py，在 pre-commit 阶段扫描 python/sglang/ 下的 .py 文件，若检测到 register_*_ci(...) 调用则报错退出。该脚本复用 ci_register.ut_parse_one_file() 以确保检测逻辑与 run_suite.py 一致。
5. 更新 CI 触发过滤器：修改 .github/workflows/ 中的路径过滤器，将 jit_kernel 和 multimodal_gen 的触发路径更新为 test/registered/jit/**，并更新文档和技能文件中的路径引用。

6. 移动存储测试：将 test_hicache_nixl_storage.py 的 CI 套件从 base-a 改为 base-b（仅主机内存文件操作，无 GPU 计算）。

关键文件：

- scripts/ci/check_no_registered_tests_in_package.py（模块 CI 检查脚本；类别 infra；类型 infrastructure；符号 main）：新增预提交保护脚本，是本次 CI 改进的核心，确保未来不会再有注册测试混入包内。
- test/registered/jit/kv_canary/test_const_sync.py（模块 常量同步；类别 test；类型 rename-or-move；符号 test_int_consts_sync）：代表性地展示了搬迁后如何通过包锚定路径修复文件定位，是路径依赖修正的范例。
- test/run_suite.py（模块 测试运行器；类别 test；类型 test-coverage；符号 run_a_suite）：移除了原先针对包内 JIT 测试的额外 glob，使测试发现仅依赖 test/registered/ 标准 glob。

关键符号：main (check_no_registered_tests_in_package.py), run_a_suite (test/run_suite.py), test_int_consts_sync (test/registered/jit/kv_canary/test_const_sync.py)

关键源码片段

scripts/ci/check_no_registered_tests_in_package.py

新增预提交保护脚本，是本次 CI 改进的核心，确保未来不会再有注册测试混入包内。

```
#!/usr/bin/env python3
"""
Pre - commit hook: 检测并拒绝 sglang 包内（python / sglang /）的注册测试。
"""
import glob
import importlib.util
import os
import sys

# 需要 AST 解析的标记字符串；无此标记的文件直接跳过
_MARKERS = (
    "register_cuda_ci",
    "register_amd_ci",
    "register_cpu_ci",
    "register_npu_ci",
    "register_xpu_ci",
    "register_musa_ci",
)

def main() -> int:
    # 直接加载 ci_register 模块（避免拉入整个 sglang 包）
    spec = importlib.util.spec_from_file_location(
        "ci_register",
        os.path.join("python", "sglang", "test", "ci", "ci_register.py"),
    )
    ci_register = importlib.util.module_from_spec(spec)
```

```
    spec.loader.exec_module(ci_register)

    offenders = []
    for f in sorted(glob.glob("python/sglang/**/*.py", recursive=True)):
        try:
            with open(f, "r", encoding="utf-8") as fh:
                source = fh.read()
        except (OSError, UnicodeDecodeError):
            continue
        # 快速过滤：不包含任何标记则跳过
        if not any(marker in source for marker in _MARKERS):
            continue
        try:
            registries, _has_main_entry = ci_register.ut_parse_one_file(f)
        except Exception:
            # 解析异常也视为存在注册（宁可误报）
            offenders.append(f)
            continue
        if registries:
            offenders.append(f)

    if offenders:
        print("ERROR: CI - registered test(s)/benchmark(s) found inside the sglang package:")
        for f in offenders:
            print(f"  {f}")
        return 1
    return 0

if __name__ == "__main__":
    sys.exit(main())
```

test/registered/jit/kv_canary/test_const_sync.py

代表性地展示了搬迁后如何通过包锚定路径修复文件定位，是路径依赖修正的范例。

```
from __future__ import annotations

import re
from pathlib import Path

import sglang.jit_kernel # 用包路径锚定
from sglang.jit_kernel.kv_canary import consts
from sglang.test.ci.ci_register import register_cuda_ci

register_cuda_ci(est_time=5, suite="base-b-kernel-unit-1-gpu-large")

# 通过已安装的 jit_kernel 包定位头文件，确保搬迁后路径正确
_CONSTS_CUH: Path = (
    Path(sglang.jit_kernel.__file__).resolve().parent
    / "csrc"
```

```
    / "kv_canary"
    / "consts.cuh"
)

def _camel_to_upper_snake(name: str) -> str:
    return re.sub(r"([A-Z])", r"_\1", name).lstrip("_").upper()

def _decode(expr: str) -> int:
    expr = expr.strip().rstrip("UuLl")
    if "<<" in expr:
        return 1 << int(expr.split("<<")[1].strip())
    return int(expr, 0)

def _parse_constexpr_ints(source: str) -> dict[str, int]:
    pattern = re.compile(r"constexpr\s+(?:[\w:]+)\s+(k[A-Za-z]\w*)\s*=\s*([^;]+);")
    return {name: _decode(rhs) for name, rhs in pattern.findall(source)}

def _parse_enum_class(source: str, enum_name: str) -> dict[str, int]:
    pattern = re.compile(
        r"enum\s+class\s+" + re.escape(enum_name) + r"\s*:\s*[^\{]+\{([^}]+)\}"
    )
    body = pattern.search(source).group(1)
    member_re = re.compile(r"(k[A-Za-z]\w*)\s*=\s*([^,]+)")
    return {name: _decode(rhs) for name, rhs in member_re.findall(body)}

def test_int_consts_sync() -> None:
    cpp = _parse_constexpr_ints(_CONSTS_CUH.read_text(encoding="utf-8"))
    cpp_normalized = {_camel_to_upper_snake(n[1:]): v for n, v in cpp.items()}
    # 后续与 consts.py 中的值对比
```

评论区精华

无实质性审核讨论；作者在 PR body 中详细阐述了动机和实现，并通过 /tag-and-rerun-ci extra 命令触发额外 CI 运行后自行合并。

- CI 重新标记与合并 (other): 作者自行合并，未产生设计争议。

风险与影响

- 风险：
  1. 路径依赖遗漏：虽然对已知的路径敏感文件做了锚定，但 121 个文件中可能仍有其他文件依赖旧路径。
  2. 预提交钩子误报：新钩子可能错误地将一些合法的包内注册（如条件注册的辅助脚本）标记为违规，但脚本只检查 python/sglang/ 下的直接注册，且复用 ut_parse_one_file，误报概率低。
  3. CI 过滤器同步：如果未来有新的 JIT 测试放在 test/registered/jit/ 外，可能不会被触发，需开发者注意。 - 影响：用户影响：JIT 内核测试不再包含在 wheel 中（对普通用户无

感知）。开发影响：JIT 内核开发者需将所有新测试和基准放到 test/registered/jit/ 下；若在包内不小心放置了注册调用，pre-commit 将直接失败。CI 影响：测试发现更统一，减少了维护两个 glob 路径的负担。所有现有测试的 CI 套件分配不变。 - 风险标记：路径依赖修正不完全，预提交钩子可能误报，CI 触发过滤器需持续同步

关联脉络

- 暂无明显关联 PR