PR #26619 完整报告

sgl-project/sglang

[CI] ci-coverage-overview: schedule + manual only, include XPU/MUSA/multimodal_gen

合并时间：2026-05-30 16:39

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26619

执行摘要

- 一句话：优化 CI 覆盖率报告触发与覆盖范围，新增 XPU/MUSA 和多模态测试
- 推荐动作：建议合入本 PR。基础设施改进，通过断言保障可维护性，启发式规则有告警机制，风险可控。后续可继续减少 CPU 测试数量或迁移更多测试至注册框架。

功能与动机

随着测试注册框架迁移，大量测试文件移入 test/registered/，导致 CI 覆盖率报告被大量不相关的 PR 触发。同时，XPU 测试在报告中不可见，MUSA 缺乏基础注册支持，多模态生成测试因使用独立运行框架被完全忽略。本 PR 旨在优化触发策略、统一后端列表、添加缺失的测试覆盖，以提供更准确的每日覆盖率快照。

实现拆解

1. 触发频率优化：在 ci-coverage-overview.yml 中移除 pull_request 触发器，仅保留 schedule（每日 6 AM UTC）和 workflow_dispatch 手动触发。同时删除作业级别的 if: github.event_name != 'pull_request' 条件判断。
2. 统一后端列表：在 ci_coverage_report.py 中定义 BACKEND_DISPLAY_ORDER 常量，包含所有 HWBackend 成员（CUDA、AMD、NPU、CPU、XPU、MUSA）并加断言确保同步，替换原来五处硬编码的列表。新增的 MUSA 后端在 ci_register.py 中添加 HWBackend.MUSA 枚举成员和 register_musa_ci 标记函数，并更新 REGISTER_MAPPING。
3. 多模态测试集成：实现 collect_multimodal_gen_tests 函数，通过解析 python/sglang/multimodal_gen/test/ 目录下的文件路径和文件名 token 规则，挂载仿真 CIRegistry 记录，使多模态生成测试（约 53 个文件）出现在覆盖率报告中。规则默认按子目录分配后端（如 server/ → CUDA+AMD，server/musa/ → MUSA），同时支持文件名 token 覆盖（musa、npu），并过滤无测试函数的管理文件。
4. 修复 CI 作业依赖：由于 unit-test-coverage 作业需要 Rust 工具链来构建原生 gRPC 扩展，但运行镜像默认没有，导致定时运行一直失败。在作业中添加安装 Rust 和 protoc 的步骤，使用与 ci_install_dependency.sh 相同的安装脚本。

关键文件：

- scripts/ci/utils/ci_coverage_report.py（模块 CI 脚本；类别 infra；类型 infrastructure；符号 BACKEND_DISPLAY_ORDER, collect_multimodal_gen_tests, collect_all_tests,

MM_GEN_SUBDIR_BACKENDS）：核心变更文件，包含后端列表统一、多模态测试分类逻辑，以及报告生成的核心改动。

- python/sglang/test/ci/ci_register.py（模块 CI 注册；类别 test；类型 test-coverage；符号 HWBackend.MUSA, register_musa_ci, REGISTER_MAPPING）：添加 MUSA 后端支持，包括 HWBackend.MUSA、register_musa_ci 和 REGISTER_MAPPING 更新。
- .github/workflows/ci-coverage-overview.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：调整触发器，删除 pull_request 触发条件，添加 Rust 工具链安装修复定时运行失败。

关键符号：collect_multimodal_gen_tests, register_musa_ci

关键源码片段

scripts/ci/utils/ci_coverage_report.py

核心变更文件，包含后端列表统一、多模态测试分类逻辑，以及报告生成的核心改动。

```
# Display order for backend tables / sections.
# 源从 HWBackend 取，通过断言确保同步
BACKEND_DISPLAY_ORDER = ("CUDA", "AMD", "NPU", "CPU", "XPU", "MUSA")
assert set(BACKEND_DISPLAY_ORDER) == {
    b.name for b in HWBackend
}, "BACKEND_DISPLAY_ORDER is out of sync with HWBackend"

# multimodal_gen 测试分类规则
MULTIMODAL_GEN_TEST_DIR = "python/sglang/multimodal_gen/test"

# 子目录默认后端映射
_MM_GEN_SUBDIR_BACKENDS = {
    "": ("CUDA",), # 顶级
    "server": ("CUDA", "AMD"),
    "server/musa": ("MUSA",),
    "server/ascend": ("NPU",),
    "layers": ("CUDA",),
    "unit": ("CUDA",),
    "cli": ("CUDA",),
    "manual": ("CUDA",),
}

# 文件名 token 后端覆盖（musa、npu 标记）
_MM_GEN_FILENAME_BACKEND_TOKENS = {
    "musa": ("MUSA",),
    "npu": ("NPU",),
}

def collect_multimodal_gen_tests(
    mm_gen_dir: str,
) -> list[CIRegistry]:
    """扫描 multimodal_gen 测试目录，合成 CIRegistry 记录。"""
```

```
    if not mm_gen_dir or not os.path.isdir(mm_gen_dir):
        return []
    results = []
    for root, dirs, files in os.walk(mm_gen_dir):
        for f in files:
            if not f.startswith("test_") or not f.endswith(".py"):
                continue
            if f in _MM_GEN_HELPER_FILENAMES:
                continue
            full = os.path.join(root, f)
            rel = os.path.relpath(full, mm_gen_dir)
            subdir = os.path.dirname(rel)
            stem = os.path.splitext(f)[0]
            tokens = set(stem.split("_"))
            # token 覆盖优先
            backends = None
            for token, b in _MM_GEN_FILENAME_BACKEND_TOKENS.items():
                if token in tokens:
                    backends = b
                    break
            if backends is None:
                # 子目录默认
                backends = _MM_GEN_SUBDIR_BACKENDS.get(subdir, ("CUDA",))
            for backend_name in backends:
                backend = HWBackend[backend_name]
                # 判定 nightly
                nightly = "nightly" in tokens
                results.append(
                    CIRegistry(
                        backend=backend,
                        filename=full,
                        est_time=1.0,
                        nightly=nightly,
                        suite="unit-test-generated",
                    )
                )
    return results
```

python/sglang/test/ci/ci_register.py

添加 MUSA 后端支持，包括 HWBackend.MUSA、register_musa_ci 和 REGISTER_MAPPING 更新。

```
# HWBackend 新增 MUSA 成员
class HWBackend(Enum):
    CPU = auto()
    CUDA = auto()
    AMD = auto()
    NPU = auto()
    XPU = auto()
```

```
    MUSA = auto() # 新增 MUSA 后端

# register_musa_ci 函数，作为 AST 标记，运行时 no-op
def register_musa_ci(
    est_time: float,
    suite: Optional[str] = None,
    nightly: bool = False,
    disabled: Optional[str] = None,
    *,
    stage: Optional[str] = None,
    runner_config: Optional[str] = None,
):
    """Marker for MUSA CI registration (parsed via AST; runtime no-op)."""
    return None

# 更新 REGISTER_MAPPING
REGISTER_MAPPING = {
    "register_cpu_ci": HWBackend.CPU,
    "register_cuda_ci": HWBackend.CUDA,
    "register_amd_ci": HWBackend.AMD,
    "register_npu_ci": HWBackend.NPU,
    "register_xpu_ci": HWBackend.XPU,
    "register_musa_ci": HWBackend.MUSA, # 新增映射
}
```

评论区精华

本 PR 无实质 review 讨论。作者在 PR body 中细致说明了变更背景和验证结果（本地运行和 workflow_dispatch 效果），并明确列出了未涉及的范围（HPU、MPS、RVV、SMG、Xeon）以及可能的后续工作（减少 CPU 测试数量）。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。主要变更集中在 CI 机械配置，不涉及核心推理或调度逻辑。潜在风险包括：1）BACKEND_DISPLAY_ORDER 断言在添加新后端时若忘记更新会阻止报告生成（已通过断言主动告警）；2）多模态测试分类启发式规则可能漂移（已设计 stderr 警告）；3）Rust 工具链安装可能失败（已使用 CI 通用脚本）。
- 影响：对 SGLang 用户无影响。对 CI 系统：减少不必要的覆盖率报告运行（之前 20 次全部是 PR 触发），节省 runner 资源。对开发者：覆盖率报告现在更准确地反映测试覆盖，包括之前缺失的 XPU（5 个）、MUSA（5 个）和多模态生成测试（52 个），有利于识别测试盲区。
- 风险标记：CI 配置变更，多模态分类启发式，断言可能导致报告失败

关联脉络

- PR #24725 nightly 迁移引入标签门控：PR body 提到该增量迁移导致大量测试移入 test/registered/ 目录，是触发频率问题的根本原因。
- PR #26613 相同的变更但来自个人 fork: 本 PR 替代 #26613，使用相同的 commits。