PR #23835 完整报告

sgl-project/sglang

[NPU] Add GitHub test summary and deduplicate test code. Part 1

合并时间：2026-05-02 19:18

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23835

执行摘要

- 一句话：为 NPU 测试添加 GitHub 总结功能并去重测试代码
- 推荐动作：此 PR 值得关注其 mixin 重构模式和 GitHub Summary 集成方案，可作为测试标准化的参考。但需注意 mixin 的隐式行为可能增加调试难度，建议为 mixin 补充详细的使用文档和单元测试。建议在 Part 2 中考虑参数化更多配置。

功能与动机

根据 PR 作者描述："It is necessary to add a GitHub summary to present test data and results in a more readable format. Also, the test code needs to be cleaned up — specifically, duplicates should be removed." 旧版测试结果散落在控制台日志中，无法快速定位失败原因；且多个测试文件重复实现了服务器启动、环境配置和评测逻辑，增加了维护成本。

实现拆解

实现步骤

1. 新增 GitHub Summary 工具函数：在 python/sglang/test/ascend/test_ascend_utils.py 中添加 write_results_to_github_step_summary，接收一个词典并按照固定表头（Model、Server、Client、性能指标、精度指标、状态）生成 Markdown 表格行，通过 write_github_step_summary 追加写入。使用 write_github_step_summary_once 确保表头只写入一次。
2. 新增 MMLU 评测基类：新建 python/sglang/test/ascend/test_mmlu.py，定义 TestMMLU 类（非继承 unittest.TestCase），提供 test_mmlu 方法。该方法通过类属性 accuracy_mmlu 获取阈值，调用 run_eval 执行评测，并通过异常捕获将结果（包括错误）写入 summary。
3. 增强 GSM8K Mixin：在 python/sglang/test/ascend/gsm8k_ascend_mixin.py 中补全 GSM8KAscendMixin，使其作为声明式 mixin 提供 setUpClass（自动启动服务器）、tearDownClass（清理进程）和 test_gsm8k（评测逻辑），并集成 summary 输出。支持通过类属性配置模型路径、启动参数、环境变量和预期精度。
4. 去重测试文件：修改 test_npu_deepep_auto_deepseek_v3_2_w8a8.py、test_npu_piecewise_graph_prefill.py、test_npu_llada2_mini.py 和 test_npu_eagle3.py，删除所有重复的 setUpClass、tearDownClass 及 test_gsm8k 实现，改为继承

GSM8KAscendMixin（或再加上 TestMMLU）。服务器配置、环境变量和预期阈值等改为类属性声明。

5. 保留性能测试并集成 Summary：对于包含 test_bs_1_speed 或 test_latency 的测试类，保留原有方法并为其添加错误捕获和 summary 写入。
6. 更新权重路径常量：在 test_ascend_utils.py 中统一新增 LLaDA2_0_MINI_WEIGHTS_PATH 等常量，避免各测试硬编码路径。
7. VLM 工具调整：对 python/sglang/test/ascend/vlm_utils.py 做小幅导入调整，保持与 mixin 兼容。

关键文件：

- python/sglang/test/ascend/test_ascend_utils.py（模块 Ascend 工具；类别 test；类型 test-coverage；符号 write_results_to_github_step_summary, write_github_step_summary_once）：核心变更文件：新增 write_results_to_github_step_summary 和 write_github_step_summary_once 函数，实现 GitHub Step Summary 输出，是本次重构的基础设施。
- python/sglang/test/ascend/test_mmlu.py（模块 MMLU 基类；类别 test；类型 test-coverage；符号 TestMMLU, test_mmlu）：新增的 MMLU 评测基类，封装了 MMLU 评测逻辑并集成 summary 输出，通过多重继承在其他测试中复用。
- test/registered/ascend/basic_function/parallel_strategy/expert_parallelism/test_npu_deepep_auto_deepseek_v3_2_w8a8.py（模块 专家并行测试；类别 test；类型 test-coverage；符号 TestDeepEpDeepseekV32, setUpClass, tearDownClass, test_mmlu）：重构典型示例，展示如何通过继承 GSM8KAscendMixin 和 TestMMLU 消除大量重复代码，从 107 行减至 62 行。

关键符号：write_results_to_github_step_summary, write_github_step_summary_once, TestMMLU.test_mmlu

关键源码片段

python/sglang/test/ascend/test_ascend_utils.py

核心变更文件：新增 write_results_to_github_step_summary 和 write_github_step_summary_once 函数，实现 GitHub Step Summary 输出，是本次重构的基础设施。

```
# HEADER 常量定义了 GitHub Summary 的表格头
HEADER = '''### Models
| Model | Server | Client | Output Throughput | Expected Output Throughput | Latency |
Expected Latency | Accuracy | Expected Accuracy | Status |
| ----- | ------ | ------ | -------- | ------------------ | ------- | ---------------- | -------- | --------- | ------ |'''

def write_results_to_github_step_summary(results: dict):
    # 只在 CI 环境中写入 Summary
    if not is_in_ci():
        return

    # 写入表头（仅第一次）
```

```
    write_github_step_summary_once(HEADER)

    # 辅助函数：从 metrics 中提取浮点数并格式化为指定精度
    get_float = lambda metrics, item, precision: (
        f'{metrics[item]:.{precision}f}'
        if isinstance(metrics.get(item, '-'), (int, float))
        else metrics.get(item, '-')
    )

    summary_rows = ''
    for model, metrics in results.items():
        # 清除模型路径前缀，只保留模型名称
        model = model.replace(MODEL_WEIGHTS_DIR, '').replace(HF_MODEL_WEIGHTS_DIR, '')

        output_throughput = get_float(metrics, 'output_throughput', 2)
        output_throughput_threshold = metrics.get('output_throughput_threshold', 'N/A')
        accuracy = get_float(metrics, 'accuracy', 4)
        accuracy_threshold = metrics.get('accuracy_threshold', 'N/A')
        latency = get_float(metrics, 'latency', 4)
        latency_threshold = metrics.get('latency_threshold', 'N/A')
        server = metrics.get('server', 'N/A')
        client = metrics.get('client', 'N/A')
        error = metrics.get('error', '')
        # 状态标记：error 为空表示成功，否则显示错误信息
        status = '✅' if error == '' else '❌ ' + str(error)

        summary_rows += f'| {model} | {server} | {client} | {output_throughput} | {output_
        throughput_threshold} | {latency} | {latency_threshold} | {accuracy} | {accuracy_threshold}
        | {status} |\n'

    write_github_step_summary(summary_rows)

def write_github_step_summary_once(summary: str):
    # 使用函数属性确保表头只写入一次
    if getattr(write_github_step_summary_once, 'has_written', False):
        return
    write_github_step_summary_once.has_written = True
    write_github_step_summary(summary)
```

python/sglang/test/ascend/test_mmlu.py

新增的 MMLU 评测基类，封装了 MMLU 评测逻辑并集成 summary 输出，通过多重继承在其他测试中复用。

```
import subprocess
from types import SimpleNamespace

from sglang.test.ascend.test_ascend_utils import write_results_to_github_step_summary
from sglang.test.run_eval import run_eval
```

```
class TestMMLU:
    '''MMLU 评测基类，用于被其他测试类多重继承'''

    def test_mmlu(self):
        # 通过 getattr 支持子类覆盖阈值，默认 0.00
        accuracy_mmlu_threshold = getattr(self, 'accuracy_mmlu', 0.00)

        model_metrics = {
            'server': getattr(
                self, 'server_cmd',
                subprocess.list2cmdline(map(str, self.other_args))
            ),
            'client': 'simple_eval_mmlu',
            'accuracy_threshold': getattr(self, 'accuracy_mmlu', 'N/A'),
        }

        try:
            args = SimpleNamespace(
                base_url=self.base_url,
                model=self.model,
                eval_name='mmlu',
                num_examples=128,
                num_threads=32,
            )
            print('Starting mmlu test...')
            metrics = run_eval(args)
            model_metrics['accuracy'] = metrics['score']
            self.assertGreater(metrics['score'], accuracy_mmlu_threshold)
        except Exception as e:
            # 捕获异常并放入 model_metrics，确保 summary 中能显示失败
            model_metrics['error'] = e
            self.fail(f'Test failed for {self.model}: {e}')
        finally:
            # 无论成功或失败，都将结果写入 GitHub Summary
            write_results_to_github_step_summary({self.model: model_metrics})
```

test/registered/ascend/basic_function/parallel_strategy/expert_parallelism/test_npu_deepep_auto_deepseek_v3_2_w8a8.py

重构典型示例，展示如何通过继承 GSM8KAscendMixin 和 TestMMLU 消除大量重复代码，从 107 行减至 62 行。

```
import os
import unittest

from sglang.test.ascend.gsm8k_ascend_mixin import GSM8KAscendMixin
from sglang.test.ascend.test_ascend_utils import DEEPSEEK_V3_2_W8A8_WEIGHTS_PATH
from sglang.test.ascend.test_mmlu import TestMMLU
from sglang.test.ci.ci_register import register_npu_ci
from sglang.test.test_utils import CustomTestCase
```

```
register_npu_ci(est_time=400, suite='nightly-16-npu-a3', nightly=True)


class TestDeepEpDeepseekV32(GSM8KAscendMixin, TestMMLU, CustomTestCase):
    '''DeepSeek V3.2 模型专家并行精度测试（使用 --deepep-mode auto）'''

    # 声明式配置：模型路径、服务器启动参数、环境变量、预期精度
    model = DEEPSEEK_V3_2_W8A8_WEIGHTS_PATH
    timeout_for_server_launch = 60000

    other_args = [
        '--trust-remote-code',
        '--tp-size', '16',
        '--quantization', 'modelslim',
        '--moe-a2a-backend', 'deepep',
        '--deepep-mode', 'auto',
        '--mem-fraction-static', '0.82',
        '--disable-cuda-graph',
        '--disable-radix-cache',
        '--context-length', '40960',
        '--max-prefill-tokens', '40960',
        '--max-total-tokens', '40960',
    ]

    env = {
        **os.environ,
        'PYTORCH_NPU_ALLOC_CONF': 'expandable_segments:True',
        'STREAMS_PER_DEVICE': '32',
        'SGLANG_DEEPEP_NUM_MAX_DISPATCH_TOKENS_PER_RANK': '16',
        'HCCL_BUFFSIZE': '1600',
        'HCCL_OP_EXPANSION_MODE': 'AIV',
        'SGLANG_NPU_USE_MLAPO': '0',
        'SGLANG_NPU_USE_MULTI_STREAM': '1',
        'TASK_QUEUE_ENABLE': '0',
    }

    accuracy = 0.95 # GSM8K 精度阈值
    accuracy_mmlu = 0.85 # MMLU 精度阈值


if __name__ == '__main__':
    unittest.main()
```

评论区精华

gemini-code-assist[bot] 在 review 中提出两个关键问题：

1. write_results_to_github_step_summary 函数中状态逻辑错误——错误键名 expected_accuracy 应为 accuracy_threshold，且成功条件应为 error == "" 而非 error！

= ""；同时 f-string 中有多余的冒号。

2. test_npu_piecewise_graph_prefill.py 缺少 GSM8KAscendMixin 的导入，会导致 NameError。 这些评论均已被作者在后续提交中修复。 另外，审核者 ping1jing2 询问 test_npu_llada2_mini.py 中 hardcode 的配置是否应支持参数化，作者补充了 TODO 注释。

- GitHub Summary 状态判断逻辑错误 (correctness): 作者在后续提交中修复了这些逻辑错误。
- 缺失 GSM8KAscendMixin 导入 (correctness): 作者在后续提交中添加了导入。
- LLaDA2 测试参数 hardcode 是否需要 TODO (design): 已添加 TODO 注释。

风险与影响

- 风险： 1) Summary 函数逻辑错误：初始实现存在状态判断错误和键名错误，虽已修复，但仍是新引入代码的质量风险。 2) 导入缺失：部分测试文件曾缺少必要的 mixin 导入，若其他开发者参考此模式可能重蹈覆辙。 3) 隐式服务器启动：Mixin 自动启动服务器，若某个测试需要特化配置但未正确覆盖 other_args 或 env，可能导致不可预期行为。 4) 环境依赖性：Summary 函数只有 is_in_ci() 返回 True 时才会写入，本地调试时无法触发，可能掩盖与 Summary 相关的错误。 5) 路径常量管理：新增的权重路径常量基于环境变量或固定路径，环境未正确配置时会导致测试集体失败。
- 影响：
  - 对用户：无直接影响，所有变更限于 NPU 测试基础设施。
  - 对系统：无性能影响。
  - 对团队：显著减少重复代码（约 300 行净删除），提升测试可维护性；GitHub Summary 使 CI 结果可视化，便于快速定位失败。后续 Part 2 可能继续推广此模式。
  - 影响程度：中等，覆盖 4 个核心 NPU 测试用例，若 mixin 有 bug 会导致大量误报。
  - 风险标记：Summary 函数逻辑已修复但需验证，mixin 导入依赖需注意

关联脉络

- 暂无明显关联 PR