PR #29440 完整报告

sgl-project/sglang

[MLX] Add correctness tests for qwen2_moe and qwen3_moe

合并时间：2026-07-07 11:32

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29440

执行摘要

- 一句话：为 MLX 后端 Qwen MoE 模型添加正确性测试
- 推荐动作：建议精读 test_mlx_reference_correctness.py 的设计：内存安全策略、模型释放顺序、batched 隔离性测试等思路值得其他后端参考。烟雾测试的框架简单可靠，适合作为更多 MLX 模型测试的起点。

功能与动机

根据 issue #19137 的模型支持计划，MLX 后端上的 Qwen MoE 系列（qwen2_moe / qwen3_moe）缺少正确性覆盖。现有测试仅检查输出是否合理（如包含 'Paris'），无法检测 KV 缓存、槽分配等细微错误。本 PR 新增强守卫（参考等价测试）来捕获回归。

实现拆解

1. 创建参考等价测试（test_mlx_reference_correctness.py）：在进程内先加载未 patch 的 mlx_lm 模型，对每个 prompt 贪心解码生成参考 token 序列；完全释放该模型后，构造 MlxModelRunner（使用 SGLang 管道）对相同 prompt 解码，断言逐 token 一致。同时测试批次解碼隔离性：单个 prompt 在批次内解码应与单独解码结果一致。内含内存预检查（_available_gb）防止 Metal OOM 硬重启。
2. 创建黑盒烟雾测试（test_qwen2_moe_mlx_correctness.py, test_qwen3_moe_mlx_correctness.py）：启动 SGLANG_USE_MLX=1 的 SGLang 服务器，通过 /v1/chat/completions 发送请求，断言输出的文本非空、包含 2+2 -> 4、包含 Paris。使用 --disable-radix-cache 和 --disable-cuda-graph 确保路径独立。
3. 根据 Review 建议移动文件：将烟雾测试从 test/registered/models/ 移动到 test/registered/mlx/models_e2e/，统一按后端布局。
4. 对齐 FakeOverlapScheduler stub（test_attention_patching.py）：在 test_finalize_pending_job_updates_scheduler_last_batch 中添加 self.assertEqual(scheduler.forward_ct, 1)，适配主分支 #29217 引入的 forward_ct 计数，修复预存的 MLX CI 失败。

关键文件：

- test/registered/unit/hardware_backend/mlx/test_mlx_reference_correctness.py（模块参考测试；类别 test；类型 test-coverage；符号 _available_gb, TestMlxReferenceCorrectness, setUpClass, tearDownClass）：进程内参考等价测试，

提供最强正确性保障，设计精巧（内存安全、精确匹配、隔离性验证）

- test/registered/mlx/models_e2e/test_qwen2_moe_mlx_correctness.py（模块 Qwen2 测试；类别 test；类型 test-coverage；符号 TestQwen2MoeMlxCorrectness, setUpClass, tearDownClass, _chat）：qwen2_moe 黑盒烟雾测试，验证服务端基本功能
- test/registered/mlx/models_e2e/test_qwen3_moe_mlx_correctness.py（模块 Qwen3 测试；类别 test；类型 test-coverage；符号 TestQwen3MoeMlxCorrectness, setUpClass, tearDownClass, _chat）：qwen3_moe 黑盒烟雾测试，与 qwen2_moe 测试结构一致
- test/registered/unit/hardware_backend/mlx/test_attention_patching.py（模块 注意力补丁；类别 test；类型 test-coverage）：对齐 FakeOverlapScheduler stub，加入 forward_ct 断言，修复预存 CI 失败

关键符号：_available_gb, TestMlxReferenceCorrectness.setUpClass, TestMlxReferenceCorrectness.tearDownClass, TestMlxReferenceCorrectness._reference_greedy, TestMlxReferenceCorrectness._prefill, TestMlxReferenceCorrectness._decode, TestMlxReferenceCorrectness._sglang_greedy, TestQwen2MoeMlxCorrectness.setUpClass, TestQwen2MoeMlxCorrectness.tearDownClass, TestQwen2MoeMlxCorrectness._chat, TestQwen2MoeMlxCorrectness.test_basic_generation_nonempty, TestQwen2MoeMlxCorrectness.test_simple_arithmetic, TestQwen2MoeMlxCorrectness.test_simple_fact, TestQwen3MoeMlxCorrectness.setUpClass, TestQwen3MoeMlxCorrectness.tearDownClass, TestQwen3MoeMlxCorrectness._chat, TestQwen3MoeMlxCorrectness.test_basic_generation_nonempty, TestQwen3MoeMlxCorrectness.test_simple_arithmetic, TestQwen3MoeMlxCorrectness.test_simple_fact

关键源码片段

test/registered/unit/hardware_backend/mlx/test_mlx_reference_correctness.py

进程内参考等价测试，提供最强正确性保障，设计精巧（内存安全、精确匹配、隔离性验证）

```
# test_mlx_reference_correctness.py
# 核心设计：先运行未 patch 的 mlx_lm 得到参考 token，再驱动 MlxModelRunner 并逐 token
断言相等
import gc, importlib, os, unittest
from sglang.test.ci.ci_register import register_cpu_ci
from sglang.test.test_utils import CustomTestCase

register_cpu_ci(est_time=1, suite="base-a-test-cpu")

_HAS_MLX = (importlib.util.find_spec("mlx") is not None and
            importlib.util.find_spec("mlx_lm") is not None)

MODEL_PATH = os.environ.get("SGLANG_MLX_TEST_MODEL",
                            "mlx-community/Qwen1.5-MoE-A2.7B-Chat-4bit")
MIN_FREE_GB = float(os.environ.get("SGLANG_MLX_TEST_MIN_FREE_GB", "12"))
```

```
def _available_gb():
    try:
        import psutil
        return psutil.virtual_memory().available / 1024**3
    except Exception:
        return None # psutil 不可用时跳过检查

@unittest.skipUnless(_HAS_MLX, "requires mlx + mlx_lm (Apple Silicon only)")
class TestMlxReferenceCorrectness(CustomTestCase):
    @classmethod
    def setUpClass(cls):
        # 预检查内存，避免 Metal OOM 硬重启
        avail = _available_gb()
        if avail is not None and avail < MIN_FREE_GB:
            raise unittest.SkipTest(
                f"可用内存不足：{avail:.1f} GB < {MIN_FREE_GB} GB，跳过测试"
            )

        # 加载未 patch 的参考模型，记录每个 prompt 的贪心解码 token
        from mlx_lm import load
        try:
            ref_model, cls.tokenizer = load(
                MODEL_PATH, tokenizer_config={"trust_remote_code": True}
            )
        except Exception as exc:
            raise unittest.SkipTest(f"无法加载模型 {MODEL_PATH}: {exc}")

        eos = getattr(cls.tokenizer, "eos_token_ids", None) or {
            cls.tokenizer.eos_token_id
        }
        cls.eos_ids = set(eos)

        # 为每个 prompt 生成参考 token 序列（省略具体循环）
        cls.cases = []
        for prompt in PROMPTS:
            # ...（具体实现使用 ref_model 贪心解码至 EOS）
            pass

        # 释放参考模型，确保峰值内存仅一份模型副本
        del ref_model
        gc.collect()
        import mlx.core as mx
        mx.clear_cache()

        # 在此之后才构造 SGLang 运行器（代码省略）
```

评论区精华

- 测试文件组织：yeahdongcn 建议将烟雾测试从 test/registered/models/ 移至 test/registered/mlx/models_e2e/ 以匹配后端布局。作者已执行移动。
- MLX 测试失败处理：作者指出 MLX CI 失败是主分支预先存在的问题，并更新 FakeOverlapScheduler stub 包含 forward_ct 断言，所有本地测试通过。
- 测试文件组织位置 (design): 作者已移动文件，并调整了 CI 注册路径。
- MLX 测试失败补救 (correctness): 更新 stub 后，所有本地测试通过。

风险与影响

- 风险：回归风险：参考测试仅在 Apple Silicon 本地运行（CI 跳过，因无 mlx 环境），可能无法及时捕获回归。内存安全：测试实现了预检查（psutil.virtual_memory）和严格的内存管理（先加载参考再释放，峰值仅一份模型），但依赖于 psutil 可用（except 时跳过检查），若内存不足可能触发 Metal OOM。模型依赖：测试默认使用外部 mlx-community 模型，网络或缓存问题会导致跳过，降低覆盖。
- 影响：用户：无直接影响。系统：对 MLX 后端增加了强回归检测，提升质量。团队：参考测试的设计（内存安全、精确匹配）可作为其他后端的模板，烟雾测试也易于扩展。
- 风险标记：测试仅在 Apple Silicon 本地运行，未在 CI 中实际执行，内存安全依赖 psutil，模型加载需网络或本地缓存

关联脉络

- PR #29217 forward_ct accounting for MLX overlap path: 该 PR 引入了 forward_ct 计数机制，本 PR 需要对齐 FakeOverlapScheduler stub 以适配新要求