PR #26617 完整报告

sgl-project/sglang

test/disaggregation: dp-attention keeps total_tokens e2e, simpler LB algos -> unit tests

合并时间：2026-05-29 07:50

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26617

执行摘要

- 一句话：精简端到端测试，简单 LB 算法移至单元测试
- 推荐动作：值得精读，尤其关注以下设计决策：
  1. 通过 __new__ 绕过复杂初始化的测试模式，适合需要测试内部调度逻辑但不愿启动完整服务器的场景。
  2. 模块文档显式列出了测试合约（_make_controller 注入的属性），便于后续维护。
  3. 状态不一致回归守卫（TestStatusAwarenessInconsistency）是一种良好的设计意图锁定手段，值得在其他调度器测试中推广。

功能与动机

PR body 指出四个 CUDA 测试用例中三个是简单算法的端到端测试，每次运行需完整 PD-disagg + DP-attention 服务器冷启动（约 6-8 分钟），而单元测试可在 base-a-test-cpu 秒级完成。目标是在保持对最复杂路径（total_tokens，包含 token 记账、平局打破和 estimated_tokens）的端到端覆盖的同时，大幅减少 CI 开销。

实现拆解

1. 分析现有测试覆盖：识别出四个端到端测试类和一个跳过的 ExternalRouting，其中三个（RoundRobin、TotalRequests、TotalTokens）共享相同的基类，仅负载均衡方法不同，但每种都需要完整的 GPU 环境。
2. 保留最复杂路径：将基类 TestDisaggregationDPAttention 的 LOAD_BALANCE_METHOD 从 "auto" 改为 "total_tokens"（最复杂的调度算法，因为包含 token 预估值和平局打破），并将原先分散在子类的 test_bench_serving 合并到基类中，使单个端到端测试既能验证 GSM8K 准确性又能验证性能门限。
3. 创建 CPU 单元测试文件：新增 test/registered/unit/managers/test_data_parallel_controller.py，通过 __new__ 绕过 DataParallelController.__init__，直接注入调度器所需的四个属性（workers、status、round_robin_counter、dp_budget），从而无需 GPU 即可测试 DPBudget 的 update_budget 和 dispatch 方法、RoundRobin 调度器、FollowBootstrapRoom 调度器、TotalRequests 调度器以及状态不一致性回归守卫。
4. 删除冗余文件：移除 test/registered/unit/managers/test_dp_budget.py（其 TestDPBudgetUpdateBudget 已迁移至新文件），删除端到端文件中的三个子类和 ExternalRouting 类。

5. 更新 CI 注册：原端到端测试保留在 base-c stage 的 8-gpu-h20 runner 上，新单元测试注册为 base-a-test-cpu suite，预计 11 秒。

关键文件：

- test/registered/unit/managers/test_data_parallel_controller.py（模块 调度器；类别 test；类型 test-coverage；符号 _make_controller, _load, _req, TestDPBudgetUpdateBudget）：新增的 CPU 单元测试文件，是本次重构的核心。通过 __new__绕过复杂初始化，直接测试 DPBudget 和所有调度器逻辑，包含状态一致性回归守卫。
- test/registered/disaggregation/test_disaggregation_dp_attention.py（模块 端到端测试；类别 test；类型 test-coverage；符号 TestDisaggregationDPAttention, test_bench_serving, test_gsm8k）：端到端测试文件，修改后仅保留 total_tokens 路径，合并 test_bench_serving 到基类，删除了三个子类和一个跳过的类。
- test/registered/unit/managers/test_dp_budget.py（模块 预算；类别 test；类型 deletion；符号 TestDPBudgetUpdateBudget, _load）：被删除的旧单元测试文件，其 DPBudget 更新测试已迁移到新的综合单元测试文件中。

关键符号：_make_controller, _load, _req, TestDPBudgetUpdateBudget.test_maps_running_plus_waiting_to_total_requests, TestDPBudgetUpdateBudget.test_maps_num_total_tokens_not_num_used_tokens, TestDPBudgetUpdateBudget.test_partial_update_only_affects_reported_rank, TestDPBudgetDispatch.test_total_requests_dispatch_picks_min_and_increments, TestDPBudgetDispatch.test_total_tokens_dispatch_applies_estimated_tokens, TestRoundRobinScheduler, TestFollowBootstrapRoomScheduler, TestTotalRequestsScheduler, TestStatusAwarenessInconsistency

关键源码片段

test/registered/disaggregation/test_disaggregation_dp_attention.py

端到端测试文件，修改后仅保留 total_tokens 路径，合并 test_bench_serving 到基类，删除了三个子类和一个跳过的类。

```
class TestDisaggregationDPAttention(PDDisaggregationServerBase):
    """PD-disagg + DP-attention e2e on `total_tokens` LB — the most complex
    dispatch (token accounting + tie-break + estimated_tokens). Simpler
    algorithms are unit-tested in
    test/registered/unit/managers/test_data_parallel_controller.py.
    """

    PREFILL_DP_SIZE = 4
    DECODE_DP_SIZE = 4
    LOAD_BALANCE_METHOD = "total_tokens" # 最复杂的负载均衡方法

    @classmethod
    def setUpClass(cls):
        super().setUpClass()
        # 临时禁用 JIT DeepGEMM
```

```
        envs.SGLANG_ENABLE_JIT_DEEPGEMM.set(False)
        cls.model = try_cached_model(DEFAULT_MODEL_NAME_FOR_TEST_MLA)
        cls.start_prefill()
        cls.start_decode()
        cls.wait_server_ready(cls.prefill_url + "/health", process=cls.process_prefill)
        cls.wait_server_ready(cls.decode_url + "/health", process=cls.process_decode)
        cls.launch_lb()

    def test_gsm8k(self):
        # 1400 个 GSM8K 示例，验证准确率 > 0.60
        args = SimpleNamespace(
            base_url=self.base_url,
            model=self.model,
            eval_name="gsm8k",
            api="completion",
            max_tokens=512,
            num_examples=1400,
            num_threads=128,
        )
        metrics = run_eval(args)
        self.assertGreater(metrics["score"], 0.60)

    def test_bench_serving(self):
        # 合并前的 RoundRobin 测试，使用 1000 个 prompt 验证 TPOT 和完成数
        args = get_benchmark_args(
            base_url=f"http://{self.base_host}:{self.lb_port}",
            dataset_name="random",
            tokenizer=self.model,
            num_prompts=1000,
            random_input_len=4096,
            random_output_len=1024,
            request_rate=float("inf"),
            max_concurrency=256,
        )
        result = run_benchmark(args)
        self.assertLess(result["mean_tpot_ms"], 20)
        self.assertEqual(result["completed"], 1000)
```

评论区精华

无 review 讨论；作者通过一条 /rerun-test 命令验证了修改后的端到端和单元测试均通过。

- 暂无高价值评论线程

风险与影响

- 风险：风险较低。主要风险包括：
  1. 端到端覆盖范围缩小，不再直接测试 round_robin 和 total_requests 的完整服务器路径，但这些算法的核心逻辑已由 CPU 单元测试覆盖，且 total_tokens 路径通过 tie-break

间接覆盖 total_requests 状态。

2. 单元测试依赖 DataParallelController.__new__ 注入属性，若未来 __init__ 增加调度器需要的新属性，需同步更新 _make_controller 和模块文档。
3. 状态不一致性测试（TestStatusAwarenessInconsistency）显式标记了 round_robin 和 total_* 调度器对待 inactive worker 的差异，未来若统一行为需主动处理该测试。 - 影响：正面影响：显著减少 CI 耗时——每次 PR 可节省约 6-8 分钟的 GPU 冷启动时间。对开发者：测试速度更快，更早获得反馈。对系统：核心调度逻辑仍然有充分的质量门禁。负面影响：极小，端到端测试仍然保留了对最复杂路径的覆盖。 - 风险标记：端到端覆盖缩减，单元测试契约依赖，状态不一致守卫

关联脉络

- 暂无明显关联 PR