PR #2234 完整报告

THUDM/slime

fix: pair --log-correct-samples rewards with the DP-local samples

合并时间：2026-08-12 13:47

原文链接：http://prhub.com.cn/THUDM/slime/pull/2234

执行摘要

- 一句话：修复 --log-correct-samples 在 DP>1 下的 IndexError 与错配
- 推荐动作：值得精读。这是一个典型的“全局与本地数据视图分离”的 bugfix，修复思路清晰：不破坏 pass@k 的全局分组需求，而是新增位置对齐的本地视图。重点关注 slime/utils/data.py 中 local_raw_reward 的派生方式、log_rollout_data 中的消费变更，以及测试对 dp1-permuted 的覆盖——这揭示了 first-fit packing 下 DP=1 也会错配的隐蔽点。附带的发现（--log-correct-samples 指标从未输出）值得跟进，但不在本 PR 内解决。

功能与动机

Issue #1784 报告了 --log-correct-samples 在 DP > 1 时 log_rollout_data 抛出 IndexError: list index out of range。PR body 给出了根因：_split_train_data_by_dp 把 per-sample 字段按 DP rank 切片，但 raw_reward 和 total_lengths 作为全局字段下发，process_rollout_data 只重切了 total_lengths，导致 log_rollout_data 用全局奖励列表索引本地 response_lengths。更隐蔽的是崩溃前会把样本 i 的奖励错误归到本地位置 i 的样本，即 DP=1 下因 first-fit packing 产生排列也会错配。

实现拆解

1. slime/utils/data.py 的 process_rollout_data 新增 local_raw_reward 派生逻辑：在按 partition 重切 total_lengths 之后，若 rollout_data 含 raw_reward，则用同样的 partition 索引生成 DP 本地视角 local_raw_reward，并保留全局 raw_reward 不变（供 log_passrate 按 [rollout_batch_size, n_samples_per_prompt] 分组）。这是对数据契约的扩展。
2. slime/backends/megatron_utils/data.py 的 log_rollout_data 消费本地视角：--log-correct-samples 分支中 raw_rewards 改为从 local_raw_reward 读取，使其与 response_lengths / total_lengths / loss_masks 位置对齐；同时在通用日志循环的跳过列表中新增 local_raw_reward，避免重复产出指标。
3. 新增 CPU 单元测试 tests/test_process_rollout_data.py：用 _FakeBox 模拟 Box、monkeypatch ray.get 为恒等函数，覆盖 dp1-permuted / dp2-interleaved / dp2-contiguous / dp4-balanced 四种 partition，验证 local_raw_reward 的位置对齐、raw_reward 保持全局、正确样本选择逻辑与拥有的样本一致、缺失 raw_reward 时容错。
4. CI 注册：在 .github/workflows/pr-test.yml 及模板 .github/workflows/pr-test.yml.j2 的 cpu-unittest 任务中登记 test_process_rollout_data.py。

关键文件：

- slime/utils/data.py（模块 数据工具；类别 source；类型 core-logic；符号 process_rollout_data）：修复核心：process_rollout_data 新增 local_raw_reward 派生，建立全局 raw_reward 与 DP 本地 per-sample 字段的桥梁，修复 IndexError 的根因。
- slime/backends/megatron_utils/data.py（模块 训练日志；类别 source；类型 core-logic；符号 log_rollout_data）：消费端修复：--log-correct-samples 分支改用 local_raw_reward，并在通用日志循环跳过该键，防止重复指标。
- tests/test_process_rollout_data.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_local_raw_reward_is_dp_local_and_aligned, test_correct_sample_selection_matches_owned_samples, test_raw_reward_stays_global, test_missing_raw_reward_is_tolerated）：新增 CPU 回归测试，覆盖 dp1-permuted / dp2 / dp4 分区下的数据契约，为修复提供回归保障，并补充 CI 注册。
- .github/workflows/pr-test.yml（模块 CI 流水线；类别 infra；类型 infrastructure）：在 cpu-unittest 作业中注册新测试文件，确保回归测试随 CI 运行。
- .github/workflows/pr-test.yml.j2（模块 CI 模板；类别 infra；类型 infrastructure）：CI 工作流模板同步更新，保证模板生成的新工作流也包含该测试。

关键符号：process_rollout_data, log_rollout_data

关键源码片段

slime/utils/data.py

修复核心：process_rollout_data 新增 local_raw_reward 派生，建立全局 raw_reward 与 DP 本地 per-sample 字段的桥梁，修复 IndexError 的根因。

```
# slime/utils/data.py

def process_rollout_data(args, rollout_data_ref, dp_rank, dp_size):
    assert len(rollout_data_ref) == dp_size
    rollout_data = ray.get(rollout_data_ref[dp_rank].inner)

    partition = rollout_data.pop("partition")
    total_lengths = rollout_data["total_lengths"]

    # 保存整个 rollout batch 的 seqlen，供 Timer 使用
    Timer().seq_lens = total_lengths
    # total_lengths 需要按 DP partition 重切，与 per-sample 字段对齐
    rollout_data["total_lengths"] = [total_lengths[i] for i in partition]

    # raw_reward 必须保持全局：log_passrate 需要把它 reshape 成
    # [rollout_batch_size, n_samples_per_prompt] 分组，只有完整 batch 才行。
    # 但像 --log-correct-samples 这样按位置配对奖励与 per-sample 列表
    # （response_lengths / loss_masks / log_probs）时，必须使用 DP 本地视图，
    # 否则样本 i 的奖励会错配到本地位置 i 的样本，甚至越界。
    if "raw_reward" in rollout_data:
```

```
        rollout_data["local_raw_reward"] = [rollout_data["raw_reward"][i] for i in partition]

    return rollout_data
```

slime/backends/megatron_utils/data.py

消费端修复：--log-correct-samples 分支改用 local_raw_reward，并在通用日志循环跳过该键，防止重复指标。

```
# slime/backends/megatron_utils/data.py 中的 log_rollout_data 相关片段

# 通用指标循环：跳过不应直接作为指标输出的内部键，local_raw_reward 就是其中之一
for key, val in rollout_data.items():
    if key in [
        "tokens",
        "multimodal_train_inputs",
        "loss_masks",
        "sample_indices",
        "rollout_ids",
        "rollout_mask_sums",
        "global_batch_sizes",
        "num_microbatches",
        "micro_batch_indices",
        "source_names",
        # DP-local 视图，与全局 raw_reward 同时存在，但循环已会输出 raw_reward，
        # 两者归约均值相同，跳过避免重复指标
        "local_raw_reward",
    ]:
        continue
    # ... 其余 (sum, count) 加权平均逻辑

# --log-correct-samples 分支：必须用 DP 本地视图，才能与 response_lengths 等对齐
if args.log_correct_samples:
    if mpu.get_tensor_model_parallel_rank() == 0 and mpu.is_pipeline_last_stage():
        # ...
        # raw_reward 本身是整个 rollout batch（log_passrate 需要完整分组），
        # 这里必须用 DP 本地视角 local_raw_reward，否则会 IndexError 或错配
        raw_rewards = rollout_data["local_raw_reward"]
        correct_response_lengths = []
        correct_total_lengths = []
        correct_loss_masks = []
        correct_entropy = []
        for i, raw_reward in enumerate(raw_rewards):
            if raw_reward == 1:
                correct_response_lengths.append(response_lengths[i])
                correct_total_lengths.append(total_lengths[i])
                correct_loss_masks.append(loss_masks[i])
        # ...
```

tests/test_process_rollout_data.py

新增 CPU 回归测试，覆盖 dp1-permuted / dp2 / dp4 分区下的数据契约，为修复提供回归保障，并补充 CI 注册。

```
# tests/test_process_rollout_data.py

class _FakeBox:
    """模拟 slime.ray.utils.Box：真实数据在 .inner 属性下。"""

    def __init__(self, inner):
        self.inner = inner


@pytest.fixture
def unwrap_ray_get(monkeypatch):
    """process_rollout_data 只用 Ray 解引用每 rank 的 Box。

    把 ray.get 猴子补丁成恒等函数，可让测试保持单进程（无需启动集群），
    同时仍走真实函数逻辑。
    """
    monkeypatch.setattr(ray, "get", lambda ref: ref)


def _split_train_data_by_dp(partitions, raw_reward, response_lengths, total_lengths):
    """复刻 RolloutManager._split_train_data_by_dp 为每个 rank 打包数据的方式。"""
    return [
        _FakeBox(
            {
                "partition": partition,
                "response_lengths": [response_lengths[j] for j in partition],
                "raw_reward": list(raw_reward),
                "total_lengths": list(total_lengths),
            }
        )
        for partition in partitions
    ]


# 8 个样本，只有奇数下标是正确的；长度编码了全局下标，
# 这样一旦错配就能在断言消息里看出来。
RAW_REWARD = [0, 1, 0, 1, 0, 1, 0, 1]
RESPONSE_LENGTHS = [100, 101, 102, 103, 104, 105, 106, 107]
TOTAL_LENGTHS = [200, 201, 202, 203, 204, 205, 206, 207]


@pytest.mark.parametrize(
    "partitions",
    [
        pytest.param([[0, 2, 4, 6], [1, 3, 5, 7]], id="dp2-interleaved"),
        pytest.param([[0, 1, 2, 3], [4, 5, 6, 7]], id="dp2-contiguous"),
```

```
        pytest.param([[0, 3], [1, 6], [2, 5], [4, 7]], id="dp4-balanced"),
        # 即使 dp_size=1，partition 也可能是排列：first-fit 打包按长度重排样本
        pytest.param([[3, 0, 7, 1, 5, 2, 6, 4]], id="dp1-permuted"),
    ],
)
def test_local_raw_reward_is_dp_local_and_aligned(unwrap_ray_get, partitions):
    dp_size = len(partitions)
    refs = _split_train_data_by_dp(partitions, RAW_REWARD, RESPONSE_LENGTHS, TOTAL_
    LENGTHS)

    for dp_rank, partition in enumerate(partitions):
        rollout_data = process_rollout_data(args=None, rollout_data_ref=refs, dp_rank=dp_rank,
        dp_size=dp_size)

        local_raw_reward = rollout_data["local_raw_reward"]
        # 本地视图必须与 partition 一致，且与 per-sample 字段位置对齐
        assert local_raw_reward == [RAW_REWARD[j] for j in partition]
        assert len(local_raw_reward) == len(rollout_data["response_lengths"])
        assert len(local_raw_reward) == len(rollout_data["total_lengths"])
```

评论区精华

评论较少，作者 keepkeen 在 PR 中主动说明了两个额外发现：一是 --log-correct-samples 块计算的 correct_response_lengths / correct_length/p* / correct_entropy 存入 rollout_data 但从未被 gather_log_data 消费，自 #1192 引入以来就未实际输出，且跨 rank 归约需要 (sum, count) 加权，作者明确声明不在本次修复范围内，可另开 PR；二是 issue comment 中作者请 zhuzilin review，zhuzilin 回复感谢，无其他 reviewer 争论。

- global raw_reward 与 DP 本地视图的设计取舍 (design): 保留 raw_reward 全局，新增 local_raw_reward 满足 per-sample 对齐需求，未被 reviewer 质疑。
- --log-correct-samples 指标可能从未被真正输出 (question): 作者建议另开 follow-up PR，当前 PR 只修崩溃和错配。

风险与影响

- 风险：主要风险是数据契约变更：新增 local_raw_reward 键不破坏旧逻辑（仅在含 raw_reward 时生成），但若其他调用方遍历 rollout_data 的全部键并做统一处理，可能意外把 local_raw_reward 当作普通指标处理——已在 log_rollout_data 的通用循环中显式跳过，但 slime/ray/rollout.py 等下游消费方未见改动，需确认是否有其他遍历逻辑。其次，local_raw_reward 在 process_rollout_data 中派生，若 partition 列表与 rollout_data 中 per-sample 字段的切片不一致（例如 future 改动引入不同 partition 语义），会产生静默错配。测试虽是 CPU 单测，但只用 monkeypatch 模拟 ray.get，未覆盖真实 Ray 集群下的 Box 解引用，存在少量集成风险。另外 PR 的 Merge branch 提交可能引入背景变更，但 diff 仅 5 个文件，影响可控。
- 影响：影响用户：启用 --log-correct-samples 且 DP > 1 的用户不再崩溃，且正确样本的长度 / 熵等统计指标从错配变为正确；此前崩溃前的错误数字也被修复。影响系统：

process_rollout_data 是训练侧通用数据处理入口，新增键对下游无破坏性，但内存占用略增（global 与 local 双份 reward 视图）。影响团队：修复了 #1784 阻塞的日志功能，并确立“全局字段 + 本地视图”的数据契约模式，为后续类似字段（如其他需要 DP 本地视角的全局统计量）提供参考。

- 风险标记：数据契约变更，新增字段需下游兼容，真实集群集成未覆盖

关联脉络

- PR #1784 [Bug] IndexError in log_rollout_data when --log-correct-samples enabled with DP > 1: 本 PR 直接修复该 issue 报告的 IndexError，根因分析与 issue 中的描述一致。
- PR #2213 Fix tau-bench token deltas for reasoning templates: 同为修复 rollout 数据流中的指标错配 / 对齐问题，涉及 rollout 数据与日志统计的一致性。
- PR #2184 sync source_names: 同属 rollout 数据从 rollout 侧到训练侧的字段传递与同步机制，修改了 slime/ray/rollout.py 的数据打包逻辑。