PR #2298 完整报告

THUDM/slime

[NFC] Add observability subfolder

合并时间：2026-08-21 17:14

原文链接：http://prhub.com.cn/THUDM/slime/pull/2298

执行摘要

- 一句话：新增 observability 子模块并迁移日志指标工具
- 推荐动作：值得作为大型 Python 项目做 NFC 结构整理的参考案例浏览。重点关注三点：一是新建子包时如何通过 rename 保留 git 历史；二是跨模块搬移后如何通过 import 集中更新与测试迁移保证行为不变；三是 rollouts_metrics 与 train_metric_utils 的职责划分（rollout 侧指标 vs 训练侧归约）可复用到其他 RL 训练框架。对功能理解而言本 PR 无新逻辑，不必深入精读每个函数。

功能与动机

PR body 明确说明动机："Move logging, debugging, tracing related utils to a new folder so that the general utils folder can be slimer."。即 slime/utils 目录承载了日志、计时器、TensorBoard、指标、trace 等多类职责，与数据、参数等通用工具混杂，边界模糊；本 PR 以 observability（可观测性）为主题收敛相关代码，为后续可观测性能力扩展提供清晰模块边界。

实现拆解

1. 新建 slime/observability 包

- 从 slime/utils/ 以重命名方式迁入基础可观测性工具：logging_utils.py、timer.py、tensorboard_utils.py、profile_utils.py 等，保留 git 历史。
- 新增 observability/init.py 作为包入口，统一导出日志、计时、追踪相关能力。

2. 汇聚指标与日志逻辑到三大新模块

- slime/observability/rollout_metrics.py（+271 行）：从 slime/ray/rollout.py 迁出 compute_metrics_from_samples、compute_perf_metrics_from_samples、log_rollout_data、log_eval_rollout_data 及 SGLang 性能字段常量，涵盖 response_len、zero_std、spec、prefix_cache、top_p 等指标计算。
- slime/observability/train_metric_utils.py（+405 行）：汇聚三处来源——megatron_utils/data.py 的 gather_log_data 与 log_rollout_data、megatron_utils/cp_utils.py 的 reduce_train_step_metrics 与 gather_and_reduce_log_dict、utils/train_metric_utils.py 的 log_perf_data 逻辑，成为训练侧指标归约与上报的统一入口。

- slime/observability/rollout_data_utils.py（+153 行）：从 slime/ray/rollout.py 迁出 _cpu_tensor、tensorize_rollout_data_for_training、validate_rollout_routed_experts_for_replay、validate_rollout_id_annotated、save/load_debug_rollout_data，并去掉私有下划线前缀。

3. 清理旧位置删除大量重复代码

- slime/ray/rollout.py 删除 409 行私有工具函数，仅保留 ServerGroup 与 RolloutServer 核心逻辑。
- slime/backends/megatron_utils/data.py 删除 322 行日志与指标代码，回归 get_batch / DataIterator 数据加载职责。
- slime/backends/megatron_utils/cp_utils.py 删除 108 行归约逻辑，保留 CP 切片与 all_gather 等并行原语。
- 整体删除 slime/utils/train_metric_utils.py（54 行）。

4. 同步更新全部依赖方 import

- 改动面覆盖 actor.py、model.py、slime/utils/data.py、slime/ray/train_actor.py、slime/rollout/sglang_rollout.py、slime/rollout/sglang_streaming_rollout.py、tools/convert_hf_to_torch_dist.py 等 40+ 处引用。
- 典型模式：slime/utils/logging_utils -> slime/observability/logging_utils；slime/backends/megatron_utils/data.log_rollout_data -> slime.observability.train_metric_utils.log_rollout_data。

5. 测试迁移配套与修复

- 新增 tests/test_rollout_data_utils.py（102 行）覆盖 tensorize、R3 校验、debug 数据存取 round-trip。
- 删除 tests/test_rollout_routing_replay_validation.py，其 R3 校验用例合并进前述新文件。
- tests/observability/test_trace_utils.py 由 tests/utils/test_trace_utils.py 重命名迁入。
- 后续两个 commit：bugfix 修复迁移遗漏引用，fix ci 修复 CI 验证问题，最终合入。

关键文件：

- slime/observability/train_metric_utils.py（模块 指标工具；类别 source；类型 dependency-wiring；符号 reduce_train_step_metrics, rollout_log_metric_contribution, gather_and_reduce_log_dict, gather_log_data）：汇聚训练侧指标归约与上报逻辑的核心新模块，集中了 reduce_train_step_metrics、rollout_log_metric_contribution、gather_and_reduce_log_dict 等从 cp_utils 与 megatron data.py 迁出的关键函数，是本次重构的汇聚点。
- slime/observability/rollout_metrics.py（模块 指标计算；类别 source；类型 core-logic；符号 compute_metrics_from_samples, compute_perf_metrics_from_samples, token_perf, _compute_sglang_request_perf_metrics）：从 slime/ray/rollout.py 迁出的 rollout 侧指标计算核心，涵盖 compute_metrics_from_samples、compute_perf_metrics_from_samples 与 SGLang 请求级性能指标解析，是 observability 子包中逻辑最完整的模块。

- slime/observability/rollout_data_utils.py（模块 数据处理；类别 source；类型 dependency-wiring；符号 _cpu_tensor, tensorize_rollout_data_for_training, validate_rollout_routed_experts_for_replay, validate_rollout_id_annotated）：承接从 slime/ray/rollout.py 迁出的 rollout 数据张量化、R3 路由回放校验、debug 数据存取逻辑，去掉私有下划线前缀成为公共接口，并配套新增独立测试文件。
- slime/ray/rollout.py（模块 rollout 执行；类别 source；类型 dependency-wiring；符号 _cpu_tensor, _tensorize_rollout_data_for_training, _validate_rollout_routed_experts_for_replay, _save_debug_rollout_data）：本次重构的最大清理对象，删除 409 行私有工具函数后仅保留 ServerGroup 与 RolloutServer 核心逻辑，且 generate/eval/_get_rollout_data 等关键方法全部改为调用 observability 公共接口。
- slime/backends/megatron_utils/data.py（模块 数据加载；类别 source；类型 dependency-wiring；符号 gather_log_data, log_rollout_data, quantile, log_multi_turn_data）：删除 322 行日志与指标相关代码后回归数据加载职责（get_batch / DataIterator），是本 PR 中 Megatron 侧清理幅度最大的文件，对外接口收缩明显。
- tests/test_rollout_data_utils.py（模块 单元测试；类别 test；类型 test-coverage；符号 _args, test_r3_validation_accepts_dense_zeros_and_complete_moe_routes, test_r3_validation_rejects_missing_pipeline_layers, test_r3_validation_rejects_wrong_shape）：新增的测试文件，承接原 test_rollout_routing_replay_validation.py 的 R3 校验用例并扩展 tensorize 与 debug 数据存取 round-trip 覆盖，是迁移后测试配套的落点。

关键符号：reduce_train_step_metrics, rollout_log_metric_contribution, gather_and_reduce_log_dict, gather_log_data, log_rollout_data, log_eval_rollout_data, compute_metrics_from_samples, compute_perf_metrics_from_samples, tensorize_rollout_data_for_training, validate_rollout_routed_experts_for_replay, validate_rollout_id_annotated, load_debug_rollout_data, save_debug_rollout_data

关键源码片段

slime/observability/rollout_data_utils.py

承接从 slime/ray/rollout.py 迁出的 rollout 数据张量化、R3 路由回放校验、debug 数据存取逻辑，去掉私有下划线前缀成为公共接口，并配套新增独立测试文件。

```
def _cpu_tensor(value, dtype: torch.dtype | None = None) -> torch.Tensor:
    # numpy 只读数组需要先复制，否则 torch.as_tensor 会共享底层 buffer，
    # 后续 detach().cpu() 可能因只读内存触发错误。
    if isinstance(value, np.ndarray) and not value.flags.writeable:
        value = value.copy()
    tensor = torch.as_tensor(value, dtype=dtype) if dtype is not None else torch.as_tensor(value)
    return tensor.detach().cpu().contiguous()


def tensorize_rollout_data_for_training(rollout_data: dict[str, Any]) -> None:
    """把 rollout 数据统一转为 CPU contiguous 张量，供 Ray object store 与训练消费。

    原实现是 slime/ray/rollout.py 的私有方法 _tensorize_rollout_data_for_training，
```

```
    迁移后去掉下划线前缀，成为 observability 模块的公共接口，调用方与测试
    都改为从这里导入。
    """
    for key, dtype in _ROLLOUT_DATA_TENSOR_DTYPES.items():
        if key in rollout_data:
            rollout_data[key] = [_cpu_tensor(value, dtype=dtype) for value in rollout_data[key]]

    if "multimodal_train_inputs" in rollout_data:
        # 多模态输入是 dict 列表，逐字段转换张量，非张量字段原样保留。
        rollout_data["multimodal_train_inputs"] = [
            (
                {
                    key: _cpu_tensor(value) if isinstance(value, (np.ndarray, torch.Tensor)) else
                    value
                    for key, value in mm_dict.items()
                }
                if mm_dict is not None
                else None
            )
            for mm_dict in rollout_data["multimodal_train_inputs"]
        ]

    if "rollout_mask_sums" in rollout_data:
        rollout_data["rollout_mask_sums"] = _cpu_tensor(
            rollout_data["rollout_mask_sums"],
            dtype=torch.float32,
        )
```

slime/ray/rollout.py

本次重构的最大清理对象，删除 409 行私有工具函数后仅保留 ServerGroup 与 RolloutServer 核心逻辑，且 generate/eval/_get_rollout_data 等关键方法全部改为调用 observability 公共接口。

```
# 迁移后统一从 observability 子包导入可观测性相关能力，原先散落在
# slime.utils 与 slime.ray 的私有工具不再保留在 rollout.py 中。
from slime.observability import logging_utils
from slime.observability.logging_utils import configure_logger, init_tracking
from slime.observability.rollout_data_utils import (
    load_debug_rollout_data,
    save_debug_rollout_data,
    tensorize_rollout_data_for_training,
    validate_rollout_id_annotated,
    validate_rollout_routed_experts_for_replay,
)
from slime.observability.rollout_metrics import log_eval_rollout_data, log_rollout_data


def generate(self, rollout_id):
    """RolloutServer 的主生成入口，本 PR 只调整工具函数调用路径。"""
```

```
    start_time = time.time()
    self.rollout_id = rollout_id
    set_current_rollout_id(rollout_id)
    self.health_monitoring_resume()
    if self.args.ci_test and self.args.use_fault_tolerance and rollout_id >= 2:
        self._try_ci_fault_injection()
    data, metrics = self._get_rollout_data(rollout_id=rollout_id)
    # 原私有方法 _save_debug_rollout_data / _log_rollout_data 迁入
    # observability 后成为模块级公共函数，调用点同步改名且行为不变。
    save_debug_rollout_data(
        self.args.save_debug_rollout_data,
        data,
        rollout_id=rollout_id,
        evaluation=False,
    )
    log_rollout_data(rollout_id, self.args, data, metrics, time.time() - start_time)
    if self.args.debug_rollout_only:
        # debug rollout only 模式下不转换为训练数据，直接返回。
        return
    data = self._convert_samples_to_train_data(data)
    return self._split_train_data_by_dp(data)
```

评论区精华

该 PR 没有任何 review 评论或评审线程（review_comments_count = 0）。可观察到的信号来自提交历史：主提交 [NFC] Add observability subfolder 之后追加了 bugfix 与 fix ci 两个修复提交，说明首版迁移存在遗漏引用或行为差异，CI 环节也一度被波及；由于提交消息未展开细节，具体修复点无法从已有材料定位，属于迁移型重构的常规补漏。

- 迁移后的遗留修复与 CI 修复（commit 推断） (other): 两个修复提交已合入，PR 以闭环状态合并到 main。具体修复点未在提交消息中展开，属于迁移型重构的常规补漏。

风险与影响

- 风险：
  - 旧 import 路径遗漏：47 个文件中有大量仅改 1-2 行 import 的改动点，若外部插件、examples 或未收录脚本仍引用 slime.utils.logging_utils、slime.utils.timer、slime.backends.megatron_utils.data.log_rollout_data 等旧路径，将直接触发 ImportError。
  - 公共模块接口收缩：slime/backends/megatron_utils/data.py 与 cp_utils.py 对外删除了 gather_log_data、log_rollout_data、reduce_train_step_metrics 等符号，任何未被本 PR 覆盖的调用方都会破裂。
  - 测试用例合并搬迁：test_rollout_routing_replay_validation.py 被删除，其 R3 校验用例（accepts_dense_zeros / rejects_missing_pipeline_layers / rejects_wrong_shape）合并到 test_rollout_data_utils.py，需确认 CI 收集与测试筛选配置没有遗漏旧路径。

    - NFC 变动依赖回归验证：三个 commit 中 bugfix 与 fix ci 表明迁移并非一次到位，合入前应确认这两处修复覆盖了真实缺陷。
- 影响：
    - 对框架使用者的影响：自定义 rollout 函数、外部指标上报脚本若依赖旧 import 路径需要同步更新；公共 API 行为（函数语义、指标键名、debug 数据格式）不变。
    - 对系统的影响：NFC 变更，运行时行为无功能变化，主要影响是代码组织与模块边界。
    - 对团队的影响：确立 observability 子模块边界，把日志、计时、追踪、指标、rollout 数据校验收敛到独立包内，未来扩展可观测性能力（如新指标、新追踪格式）不再污染通用 utils 目录。
    - 影响程度：影响面广但风险可控，属于结构性整理而非功能演进。
    - 风险标记：跨 47 文件 import 变更，删除旧公共模块路径，测试用例合并搬迁，NFC 变更依赖回归验证

关联脉络

- PR #2294 cleanup: 同为代码结构整理型 PR，清理 GLM-5 逐层比较脚本并移除冗余依赖，与本 PR 的瘦身 + 整理目标一致，且时间接近。
- PR #2266 Refactor --save-debug-train-data: 重构 debug 训练数据转储逻辑，与本 PR 迁入 observability/rollout_data_utils.py 的 save_debug_rollout_data / load_debug_rollout_data 属于同一 debug 数据功能线，后续随本 PR 统一收敛到 observability 模块。