PR #6736 完整报告

verl-project/verl

[trainer] feat: add off_policy metrics

合并时间：2026-06-16 08:55

原文链接：http://prhub.com.cn/verl-project/verl/pull/6736

执行摘要

- 一句话：添加 off-policy 指标和 replay buffer 陈旧性排序
- 推荐动作：值得精读，尤其是 replay buffer 的排序策略和 off-policy 指标定义。但需注意 review 中未解决的 padding 过滤问题，若后续需要准确指标，建议合并前采用该建议。abort 重试条件放宽可能引入风险，建议评估是否加最大重试次数。

功能与动机

根据 PR 描述，添加 off-policy 指标用于监控策略陈旧性，并通过优先采样旧数据减少 replay buffer 中的 staleness。FullyAsyncLLMServerClient 在部分 rollout 中遇到 abort 后应自动重试，但原条件依赖 async_training.partial_rollout 配置，导致重试逻辑未生效（#6736 body）。

实现拆解

1. ReplayBuffer 支持 global_steps 记录和排序采样

- 文件: verl/trainer/ppo/v1/replay_buffer.py
  - 新增 prompt_global_steps 字典（partition_id → {prompt_key: global_steps}），在 _sync_metadata_from_transfer_queue 中从 prompt 的 tag 中解析 global_steps 并记录。
- 修改 sample 方法：在选取 prompt 时，先对 finished_keys ∪ failure_keys 按 prompt_global_steps 升序排序，再取前 batch_size 个，确保最旧的数据优先被采样。若 tag 中缺失 global_steps 则默认 0。

2. Trainer 添加 off-policy 指标

- 文件: verl/trainer/ppo/v1/trainer_base.py
- 在 _compute_metrics 中，从 batch.tags 提取每个 trajectory 的 min_global_steps 和 max_global_steps，然后计算：
  - trajectory_spans = max_global_steps - min_global_steps + 1：一条轨迹跨越的模型版本数。
  - trajectory_staleness = (global_steps - 1) - max_global_steps：该轨迹相对于当前策略的滞后步数（下限）。
  - trajectory_staleness_worst = (global_steps - 1) - min_global_steps：滞后步数上限。

- 更新 metrics 字典，记录 mean/max/min 等统计值。
- 修复两处文档注释：one-policy → on-policy。

3. LLMServerClient 和 vLLM 服务器完善 global_steps 传递

- 文件: verl/workers/rollout/llm_server.py
  - 在 LLMServerClient.generate 中，对从远程服务器返回的 output，通过 setdefault 填充 min_global_steps 和 max_global_steps 为 global_steps，确保单次生成（非部分 rollout）也有完整字段。
  - 在 FullyAsyncLLMServerClient.generate 中，移除了 abort 重试对 async_training.partial_rollout 配置的依赖，只要 stop_reason 是 "aborted" 或 "abort" 就重试（原条件需同时存在配置，导致多轮重试失效）。
- 文件: verl/workers/rollout/vllm_rollout/vllm_async_server.py
  - 在 generate 中，当请求被 abort 时，构造 TokenOutput 实例时加入 extra_fields = {"global_steps": self.global_steps}，避免客户端访问 output.extra_fields["global_steps"] 时触发 KeyError。

4. 配套测试

- 文件: tests/trainer/ppo/v2/test_replay_buffer_on_cpu.py
- 新增三个测试：
  - test_sync_metadata_records_prompt_global_steps：验证 _sync_metadata_from_transfer_queue 能否正确记录每个 prompt 的 global_steps。
  - test_sample_prioritizes_smallest_global_steps：验证当可用 prompt 数 > batch_size 时，是否按 global_steps 升序选择最旧的。
  - test_sample_orders_by_global_steps_across_finished_and_failure：验证排序跨 finished 和 failure 两类 prompt 统一进行。
  - 修改 PromptSpec 数据类加入 global_steps 字段，辅助构造测试数据。

关键文件：

- verl/trainer/ppo/v1/trainer_base.py（模块 训练器；类别 source；类型 core-logic；符号 PPOTrainer._compute_metrics, PPOTrainer.get_teacher_client）：核心变更：添加 off-policy 指标计算，修改 _compute_metrics 函数，新增 trajectory_spans 和 trajectory_staleness 统计。
- verl/trainer/ppo/v1/replay_buffer.py（模块 训练器；类别 source；类型 core-logic；符号 ReplayBuffer.init, ReplayBuffer._sync_metadata_from_transfer_queue, ReplayBuffer.sample）：核心变更：新增 prompt_global_steps 记录，修改 sample 方法按 global_steps 升序选择 prompt，减少抽样陈旧性。
- verl/workers/rollout/llm_server.py（模块 Rollout；类别 source；类型 core-logic；符号 LLMServerClient.generate, FullyAsyncLLMServerClient.generate）：重要变更：修复 LLMServerClient 单次生成缺失 min/max_global_steps，修复 FullyAsyncLLMServerClient abort 重试条件。

- tests/trainer/ppo/v2/test_replay_buffer_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 test_sync_metadata_records_prompt_global_steps, test_sample_prioritizes_smallest_global_steps, test_sample_orders_by_global_steps_across_finished_and_failure）：测试配套：新增三个测试用例覆盖 replay buffer 的 global_steps 记录与排序行为。
- verl/workers/rollout/vllm_rollout/vllm_async_server.py（模块 Rollout；类别 source；类型 core-logic）：配套修复：在 abort 时构造 TokenOutput 中加入 extra_fields（global_steps），避免客户端 KeyError。

关键符号：ReplayBuffer.init, ReplayBuffer._sync_metadata_from_transfer_queue, ReplayBuffer.sample, PPOTrainer._compute_metrics, LLMServerClient.generate, FullyAsyncLLMServerClient.generate

关键源码片段

verl/trainer/ppo/v1/trainer_base.py

核心变更：添加 off-policy 指标计算，修改 _compute_metrics 函数，新增 trajectory_spans 和 trajectory_staleness 统计。

```
# 位于 PPOTrainer._compute_metrics 方法中，在原有指标后追加
# 从 batch.tags 中提取模型版本信息
min_global_steps = np.array([tag["min_global_steps"] for tag in batch.tags], dtype=int)[non_
padding_mask]
max_global_steps = np.array([tag["max_global_steps"] for tag in batch.tags], dtype=int)[non_
padding_mask]

# trajectory_spans: 一条轨迹跨越的不同模型版本数（1 表示完全在同一版本生成）
trajectory_spans = max_global_steps - min_global_steps + 1
# trajectory_staleness: 该轨迹相对于当前训练步骤的滞后步数（下限）
trajectory_staleness = (global_steps - 1) - max_global_steps
# trajectory_staleness_worst: 滞后步数上限（使用最旧版本计算）
trajectory_staleness_worst = (global_steps - 1) - min_global_steps

metrics.update({
    "training/off_policy/trajectory_spans/mean": trajectory_spans.mean(),
    "training/off_policy/trajectory_spans/max": trajectory_spans.max(),
    "training/off_policy/trajectory_spans/min": trajectory_spans.min(),
    "training/off_policy/trajectory_staleness/mean": trajectory_staleness.mean(),
    "training/off_policy/trajectory_staleness/max": trajectory_staleness.max(),
    "training/off_policy/trajectory_staleness/min": trajectory_staleness.min(),
    "training/off_policy/trajectory_staleness_worst/mean": trajectory_staleness_worst.mean(),
    "training/off_policy/trajectory_staleness_worst/max": trajectory_staleness_worst.max(),
    "training/off_policy/trajectory_staleness_worst/min": trajectory_staleness_worst.min(),
})
```

verl/trainer/ppo/v1/replay_buffer.py

核心变更：新增 prompt_global_steps 记录，修改 sample 方法按 global_steps 升序选择 prompt，减少抽样陈旧性。

```
  def sample(self, partition_id: str, batch_size: int) -> KVBatchMeta:
      """采样一批数据。优先返回最旧的 finished/failure prompt，减少策略陈旧性。"""
      last_debug_time = time.time()
      self._sync_metadata_from_transfer_queue()
      # 等待直到有足够的可采样 prompt（finished + failure）
      while len(self.finished_keys[partition_id]) + len(self.failure_keys[partition_id]) < batch_size:
          time.sleep(self.poll_interval)
          self._sync_metadata_from_transfer_queue()
          # 定期日志
          if time.time() - last_debug_time > VERL_REPLAY_BUFFER_DEBUG_INTERVAL_SECONDS:
              logger.info(f"pending: {len(self.pending_keys[partition_id])}, ...")
              last_debug_time = time.time()

      finished_keys = self.finished_keys[partition_id]
      failure_keys = self.failure_keys[partition_id]
      # 按 global_steps 升序排序（缺失则视为 0，即最旧）
      prompt_global_steps = self.prompt_global_steps[partition_id]
      sampleable_keys = sorted(
          finished_keys.union(failure_keys),
          key=lambda key: prompt_global_steps.get(key, 0)
      )
      selected_prompt_uids = sampleable_keys[:batch_size]
      tq.kv_clear(partition_id=partition_id, keys=selected_prompt_uids)

      # 收集所有属于选中 prompt 的 trajectory key 和 tag
      keys, tags = [], []
      selected = set(selected_prompt_uids)
      for key, tag in self.partitions[partition_id].items():
          uid = key.split("_")[0]
          if uid in selected:
              keys.append(key)
              tags.append(tag)
      return KVBatchMeta(partition_id=partition_id, keys=keys, tags=tags)
```

评论区精华

仅有一条来自 gemini-code-assist[bot] 的 review comment，指出 off-policy 指标计算未使用 non_padding_mask 过滤 padding 序列，会导致均值失真。机器人附上了具体建议代码。PR 作者未采纳该建议，最终提交未包含相应修改，因此该问题仍未解决。

- Off-policy 指标应使用 non_padding_mask 过滤 padding 序列 (correctness): 作者未采纳，指标仍包含 padding 数据。

风险与影响

- 风险：

    - 指标准确率：review 评论指出 off-policy 指标未过滤 padding 序列，均值和极值可能被 dummy 数据污染，影响监控可靠性。位置：verl/trainer/ppo/v1/trainer_base.py 中 _compute_metrics 新增代码。
    - abort 重试无限循环：FullyAsyncLLMServerClient 现在只要遇到 abort 就重试，若服务器持续返回 abort（如资源不足），可能陷入无限重试循环。原配置检查提供了一定的防护。位置：verl/workers/rollout/llm_server.py 中条件简化。
    - 排序性能：sample 方法每次都对所有可采样 prompt 进行全排序，当分区中 prompt 数量很大时可能引入延迟。但当前 batch_size 较小，影响有限。
    - global_steps 缺失降级：若外部未在 tag 中提供 global_steps，则默认 0，会导致该 prompt 总是被优先采样，可能加剧陈旧性。
- 影响：
    - 用户影响：所有使用 PPO async trainer（> v1）的训练都会输出两组新指标；replay buffer 采样顺序改变，可能影响训练行为（更倾向使用旧数据，可能促进稳定性但延迟新数据使用）；abort 重试修复使得部分 rollout 从中断恢复更可靠。
    - 系统影响：无侵入，兼容旧配置。若用户未设置 global_steps，采样退化为默认 0，原有行为不变（只是新数据不再被优先采样）。
    - 团队影响：需要监控新指标的有效性，并考虑是否实施 padding 过滤。
    - 风险标记：padding 未过滤，abort 重试无限循环风险，排序性能开销，global_steps 默认 0 降级风险

关联脉络

- PR #6710 [trainer] feat: add unify trainer abstraction for sync and async training: 6710 引入了统一的 trainer 基类 PPOTrainer 和 replay_buffer，本 PR 在此基础上添加 off-policy 指标和排序逻辑，共享 trainer_base.py 和 replay_buffer.py。
- PR #6716 [trainer] fix: use FullyAsyncLLMServerClient for async trainer: 6716 将 FullyAsyncLLMServerClient 移入核心模块，本 PR 修复了其中 abort 重试条件，并完善了 global_steps 字段传递，属同一功能线。