PR #7032 完整报告

verl-project/verl

[tool, rollout] feat: support parameter sync steps in Skip Manager

合并时间：2026-07-14 10:04

原文链接：http://prhub.com.cn/verl-project/verl/pull/7032

执行摘要

- 一句话：SkipManager 支持参数同步步长内多批量缓存
- 推荐动作：建议在后续 PR 中解决 review 指出的两个问题：保持 parameter_sync_step == 1 时向后兼容，以及清理不完整步骤目录。当前版本可用于实验，但生产部署需谨慎。该 PR 展示了 SkipManager 对 separate async 模式的适配思路，值得了解目录结构和合并逻辑。

功能与动机

在 separate async 模式下，一个 global step 会多次调用 sample，但原有的缓存机制只能保存单个 tq_batch.pt 文件，无法区分多个 mini-batch。关联 Issue #7007 要求 Enable SkipManager in the sync trainer CI and adapt the param_sync_step accordingly，此 PR 实现了对 parameter_sync_step 的适配。

实现拆解

1. 读取参数同步步长：在 RolloutTqSkip.__init__ 中通过 OmegaConf.select 从全局配置中读取 trainer.v1.{trainer_mode}.parameter_sync_step，默认值为 1。
2. 内层子目录管理：新增 _get_v1_inner_dir 方法，生成 {step}/{inner_idx}/ 路径；_check_valid_v1_step 检查步骤的所有内层目录是否完整存在；_find_first_missing_inner 返回第一个缺失的内层索引。
3. 完整步骤校验：将原有 _check_valid_v1_step_path 改为判断单目录，新增 _check_valid_v1_step 方法对 parameter_sync_step 个内层目录进行全量检查。_get_available_steps_v1 和 _resolve_load_step_v1 改用 _check_valid_v1_step 确保只有完整缓存的步骤才被识别。
4. 保存与加载适配：prepare_data 根据 _find_first_missing_inner 决定当前 mini-batch 写入哪个内层目录；maybe_load_and_inject 中的 load_dump_data 负责在加载时合并所有内层数据。
5. 文档与提示更新：在类注释中解释了 parameter_sync_step > 1 时的目录结构，并在 maybe_load_and_inject 中增加了缓存缺失时的打印提示。
6. skip_manager.py 文档补充：在 annotate_tq 的 phase="sample" 文档中描述了多个 mini-batch 保存到独立内层子目录的行为。

关键文件：

- verl/utils/skip/rollout_skip.py（模块 跳过管理器；类别 source；类型 core-logic；符号 init, _get_v1_inner_dir, _check_valid_v1_step, _find_first_missing_inner）：核心变更文件，新增 parameter_sync_step 读取、内层子目录管理方法、完整步骤校验逻辑，并修改了缓存目录推导和可用步骤获取逻辑。
- verl/utils/skip/skip_manager.py（模块 跳过管理器；类别 source；类型 core-logic）：更新了 annotate_tq 的文档注释，描述了 parameter_sync_step > 1 时多 mini-batch 保存到独立内层子目录的行为。

关键符号：init, _get_v1_inner_dir, _check_valid_v1_step, _find_first_missing_inner, _check_valid_v1_step_path, _get_available_steps_v1, _resolve_load_step_v1

关键源码片段

verl/utils/skip/rollout_skip.py

核心变更文件，新增 parameter_sync_step 读取、内层子目录管理方法、完整步骤校验逻辑，并修改了缓存目录推导和可用步骤获取逻辑。

```
# 文件：verl/utils/skip/rollout_skip.py
# 关键变更：支持 parameter_sync_step > 1 的 multi-batch 缓存

class RolloutTqSkip(RolloutSkip):
    """Rollout skip for V1 TransferQueue-based trainer (``skip.rollout_tq``).

    When ``parameter_sync_step > 1`` (separate async), one global step performs multiple
    ``sample`` calls. Each mini-batch is saved to a separate inner sub-directory
    ``{step}/{inner_idx}/`` so the full step is captured; on load all inner dirs are
    merged. For ``parameter_sync_step == 1`` (sync / colocate async) the directory
    structure is (``{step}/{0}/tq_batch.pt``).
    """

    def __init__(self, local_config, global_config):
        super().__init__(local_config, global_config)
        # 从全局配置读取参数同步步长，默认值为 1（表示没有额外的 param_sync 步骤）
        self.parameter_sync_step = int(
            OmegaConf.select(
                global_config,
                f"trainer.v1.{global_config.trainer.v1.trainer_mode}.parameter_sync_step",
                default=1,
            )
        )

    def _get_v1_inner_dir(self, step: int, inner_idx: int) -> Path:
        """Return the dump directory for one mini-batch within a step."""
        # 示例：对于 step=10, inner_idx=2，返回 .../projects/.../10/2/
        return self._get_step_dump_dir(step) / str(inner_idx)

    def _check_valid_v1_step(self, step: int) -> bool:
        """Check whether ALL inner dirs for *step* exist (complete cache)."""
```

```
        # 必须所有内层子目录都存在且包含有效文件，才算一个完整的步骤缓存
        return all(
            self._check_valid_v1_step_path(self._get_v1_inner_dir(step, i))
            for i in range(self.parameter_sync_step)
        )

    def _find_first_missing_inner(self, step: int) -> int:
        """Return the first inner index whose dir does not yet exist."""
        # 用于 decide 下一个应该保存的 inner index
        for i in range(self.parameter_sync_step):
            if not self._check_valid_v1_step_path(self._get_v1_inner_dir(step, i)):
                return i
        return -1 # all present
```

评论区精华

Gemini Code Assist 机器人提出了两个高优先级问题：

1. 向后兼容性：当 parameter_sync_step == 1 时，使用 {step}/{0}/ 子目录会破坏与旧版缓存格式 {step}/tq_batch.pt 的兼容性。建议仅在 parameter_sync_step > 1 时使用子目录。
2. 数据污染风险：如果上一次运行被中断，部分内层目录已存在但步骤缓存不完整，新运行启动时 _find_first_missing_inner 会跳过已有目录写入下一个索引，导致新旧数据混合。建议在第一次准备数据时清理不完整的步骤目录。

这两个问题在 PR 中尚未解决，且最终提交的代码未体现修改，但仍被 wuxibin89 批准合并。

- parameter_sync_step == 1 时的向后兼容性 (design): 未采纳建议，PR 仍使用统一的子目录格式。
- 中断运行导致数据污染 (correctness): 未采纳建议，PR 未添加清理逻辑。

风险与影响

- 风险：
  1. 向后兼容性风险：当 parameter_sync_step == 1 时（同步或 colocate async），缓存目录从 {step}/tq_batch.pt 变为 {step}/0/tq_batch.pt，导致之前运行的缓存无法被读取，可能引起运行时错误或重复 rollout。
  2. 数据污染风险：如果训练进程被中断，部分内层目录残留，下次运行时可能混合新旧数据，造成模型更新异常。
  3. 现有功能无测试覆盖：仅通过手动实验验证，缺乏 CI 测试，后续重构可能引入回归。 - 影响：影响范围限于使用 SkipManager 的 separate async 训练模式。对于 parameter_sync_step > 1 的用户，这是重要的功能增强；对于 parameter_sync_step = 1 的现有用户，存在向后兼容性问题。影响程度中等，但需要关注数据完整性。 - 风险标记：向后兼容性破坏，数据污染风险，缺少测试覆盖

关联脉络

- PR #6897 [tool, rollout] feat: Adapt SkipManager on Trainer V1: 本 PR 是在 #6897 的基础上为 SkipManager 增加 parameter_sync_step 支持，关联 Issue #7007 也源自 #6897 的讨论。