PR #6248 完整报告

verl-project/verl

[tool] feat: Memory snapshot collection, add functionality to clear history after collection.

合并时间：2026-05-07 09:12

原文链接：http://prhub.com.cn/verl-project/verl/pull/6248

执行摘要

- 一句话：每次采集内存快照后自动清除历史记录
- 推荐动作：该 PR 修复了内存快照采集的累积 bug，改动简洁且针对性强。建议快速合入，后续可考虑增加 context/stacks 等参数的透传，以允许用户更精细地控制记录内容。

功能与动机

Issue #6208 报告了在 global_profiler.steps=[1,5,10] 场景下，step5 和 step10 的采集因历史快照数据未删除，导致采集到的数据是从开始到当前 step 的累积数据，越往后数据越大，不利于内存问题定位。PR body 中附有对比截图。

实现拆解

1. 新增 clear_memory_history 函数(verl/utils/memory_utils.py): 通过调用 device.memory._record_memory_history(enabled=None) 清除 PyTorch 记录的 allocation history，然后重新启用记录以保留后续采集能力。
2. 在 TorchMemoryProfiler.stop() 中自动清除(verl/utils/profiler/profile.py): 在 dump 快照后无条件调用 clear_memory_history（不引入配置开关，因为 reviewer tardis-key 认为当前累积行为是 bug，应直接修复）。
3. 将局部变量改为实例属性(profile.py): 将 trace_alloc_max_entries 和 stack_depth 从 __init__ 的局部变量改为 self.trace_alloc_max_entries / self.stack_depth，以便 stop() 中复用同一参数值调用 clear_memory_history。
4. 更新 import(profile.py): 增加 clear_memory_history 导入。

关键文件：

- verl/utils/memory_utils.py（模块 工具；类别 source；类型 core-logic；符号 clear_memory_history）：新增 clear_memory_history 函数，负责清除 PyTorch 的 allocation history 并重新启用记录，是本次修复的核心逻辑。
- verl/utils/profiler/profile.py（模块 工具；类别 source；类型 dependency-wiring）：修改 TorchMemoryProfiler 的 __init__ 和 stop 方法，在每次 dump 后调用 clear_memory_history，并将局部变量提升为实例属性以便复用。

关键符号：clear_memory_history, TorchMemoryProfiler.stop, TorchMemoryProfiler.init

关键源码片段

verl/utils/memory_utils.py

新增 clear_memory_history 函数，负责清除 PyTorch 的 allocation history 并重新启用记录，是本次修复的核心逻辑。

```
def clear_memory_history(trace_alloc_max_entries: int = 200_000, stack_depth: int = 32):
    """
    清除 PyTorch 记录的 allocation history，然后重新启用记录。
    用于在每次内存快照 dump 后重置，使下次快照只记录本次采集区间内的 allocation。
    """
    device = get_torch_device()
    if not device.is_available():
        logger.warning("[memory_visualize] Memory history recording is only available on
        accelerator devices")
        return
    try:
        # 清除所有已记录的历史
        device.memory._record_memory_history(enabled=None)
        # 以相同的参数重新启用记录
        enable_memory_visualize(trace_alloc_max_entries=trace_alloc_max_entries, stack_depth=
        stack_depth)
    except Exception as e:
        logger.warning(f"[memory_visualize] Failed to reset memory history: {e}")
```

verl/utils/profiler/profile.py

修改 TorchMemoryProfiler 的 __init__ 和 stop 方法，在每次 dump 后调用 clear_memory_history，并将局部变量提升为实例属性以便复用。

```
class TorchMemoryProfiler:
    # ...
    def stop(self):
        if not self.enable or not self.this_step:
            return
        self.this_step = False
        if not self._should_profile_this_rank():
            return
        out_dir = self.config.save_path or "outputs/profile"
        tag = "torch_memory"
        # 所有 rank 写入相同子目录
        try:
            self.sampler.dump_memory_snapshot(out_dir=out_dir, tag=tag, sub_dir=self.sub_dir)
        except Exception:
            pass
        # 每次快照后清除 history，使下次快照只反映新 interval 的变化
        if TorchMemoryProfiler._memory_history_enabled:
            clear_memory_history(trace_alloc_max_entries=self.trace_alloc_max_entries,
                                 stack_depth=self.stack_depth)
```

评论区精华

- gemini-code-assist[bot]提出了大量关于 missing context/stacks 参数的 review 建议，涉及 TorchMemoryToolConfig dataclass 和多个 YAML 配置文件。但 contributor 和 reviewer tardis-key认为当前 PR 的核心目的是修复累积 bug，不需要引入可配置的 clear_history 开关，更复杂的 context/stacks 参数传递可后续 PR 处理。
- tardis-key明确表示："The memory snapshot is supposed to clean the pre-step data, and the current behavior is a bug. So there is no need to add a parameter to control whether to clear it or not." 此评论得到了 contributor 的认同（回复"已处理"）。
- 是否添加 clear_history 配置开关 (design): 不引入配置开关，强制清除历史。
- 缺少 context 和 stacks 参数的透传 (correctness): PR 作者与 reviewer 认为当前 PR 聚焦于修复累积 bug，context/stacks 透传可后续处理。
- YAML 配置中缺少 clear_history 字段 (documentation): 不添加配置字段，因此 YAML 无需变更。

风险与影响

- 风险：
  - 回归风险低：改动仅影响 TorchMemoryProfiler 的内存快照采集路径，且是在 dump 之后才清除 history，不影响快照文件本身的内容和保存。
  - 若 PyTorch 版本不支持 _record_memory_history(enabled=None)，该函数会 catch 异常并仅打 warning，不会崩溃。
  - 没有配置开关：清除行为是强制性的，用户无法选择保留历史，但这是 reviewer 确认的预期修复方向。
- 影响：
  - 用户 / 开发者：多 step 采集时，每个 step 的快照文件仅包含该 step 区间内的 allocation 记录，便于使用 PyTorch Memory Viz 进行内存泄漏分析。
  - 系统：无性能影响，clear_memory_history 开销极低。
  - 团队：统一了 verl 与 vLLM 的内存快照行为，有助于 NPU (Ascend) 设备上的内存调试。
  - 风险标记：无配置开关，强制清除，缺少 context/stacks 透传

关联脉络

- PR #6216 [tool] fix: In the memory snapshot collection logic, opening history records is not compatible with NPU: 同一作者在同一模块（memory_utils）的先前修复，也是 NPU 兼容性相关。
- PR #6184 [veomni] feat: use VeOmni's native return_log_probs path to compute log_probs: 修改了 memory_utils.py 和 profiler 相关的配置，但与本 PR 无直接功能关联。