PR #7408 完整报告

verl-project/verl

[perf] feat: add profiler post hook, fix behaviors and allow comprehensive torch profiler

合并时间：2026-08-19 12:59

原文链接：http://prhub.com.cn/verl-project/verl/pull/7408

执行摘要

- 一句话：新增 profiler 收尾钩子，重构 torch 与 rollout 全链路 profiling 采集
- 推荐动作：值得精读。三个设计决策值得借鉴：① finish hook 采用“每步 relocate + 末尾一次性执行命令”，避免重复上传累积目录；② torch profiler 的 schedule 粒度从 RL step 改为 update mini-batch，并关闭 ProfilerStep 行来去噪；③ rollout profiler 的 ranks 按全局 GPU rank 映射到 replica，并在 trace 重定位时过滤 tp 同伴，保证只露出用户请求的 rank。建议同时关注 reviewer 对 rollout profiling 必要性的质疑，评估是否应默认关闭并以 Prometheus 替代。

功能与动机

PR body 明确提出目标：add profiler post hook, fix behaviors and allow comprehensive torch profiler。此前 torch profiler 的调度粒度、trace 文件布局和 rollout 引擎的 profiler 产物收集都不完整，用户需要手动从 Ray 固定目录或 per-replica 子目录寻找 trace。本 PR 通过 POST hook 把 " 收集 / 上传 " 自动化，并让 discrete 与 continuous 两种模式都能得到可归属到 role/step 的 Chrome trace。Reviewer 的质疑也反映了需求边界：rollout 引擎是否需要在 RL 框架内做细粒度 profile，还是应由 Prometheus 承担。

实现拆解

1. 扩展配置契约（verl/utils/profiler/config.py）：ProfilerConfig 新增 relocate_results、finish_hook_cmd、finish_hook_all_ranks、finish_hook_ranks 字段，并在 union/intersect 中传播这些字段；TorchProfilerScheduleConfig 语义从“step”改为“update mini-batch”，文档明确离散 / 连续两种模式下 schedule 的取舍；新增 rollout_trace_dir、rollout_profiler_global_ranks、relocate_rollout_traces 等辅助函数，统一 rollout 引擎 trace 目录推导与重定位逻辑。
2. 实现 DistProfiler 收尾钩子（verl/utils/profiler/profile.py）：stop(run_command=True) 每次 stop 都调用 _run_finish_hook——每步搬迁 nsys 报告等框架固定目录产物到 save_path，但用户 finish_hook_cmd 只在最后一个 profiled step 执行一次，避免重复上传累积目录；_run_finish_command 通过 /bin/sh -c 执行并导出 VERL_PROFILE_* 环境变量，异常只告警不打断训练。
3. 修复 torch profiler 行为（verl/utils/profiler/torch_profile.py）：get_torch_profiler 不再按 role 建子目录，全部 trace 落在 save_path，文件名通过 build_trace_basename 嵌入 role 与 profile_step；CPU activity 始终开启以保证 record_function 阶段标记可见；

有 schedule 时设置 prof.record_steps = False 消除 ProfilerStep#<n> 噪声；discrete update 阶段用 _scheduled_mini_batch_range 反推窗口覆盖的 mini-batch 区间并写入 _mb 后缀。

4. 打通 trainer 与 rollout 引擎（verl/trainer/ppo/ray_trainer.py、v1/trainer_base.py、vllm_async_server.py、sglang/trtllm_async_server.py）：_stop_profiling 在最后一个 profiled step 传 run_command=True；trainer 同时启停 _rollout_server_managers；vLLM 侧 _should_profile 只对选中 replica 配置引擎 profiler，stop_profile 后调用 relocate_rollout_traces 把 per-replica 子目录中的 trace 按全局 rank 过滤迁移；shared worker group（actor_rollout_wg 与 ref/critic 同对象）只 profile 一次。
5. 测试与文档配套：新增 tests/workers/test_engine_workers_mini_batch_trace_on_cpu.py（mini-batch 标注与 step 次数）、tests/trainer/ppo/v1/test_rollout_profiling_on_cpu.py（rollout 引擎驱动与去重）、tests/trainer/test_constants_ppo_on_cpu.py（torch 覆盖 nvtx 注入）；大幅扩充 test_server_profiler.py、test_torch_profile.py、test_nvtx_profile.py；同步更新 docs/perf、examples/profile 与 .github/workflows。

关键文件：

- verl/utils/profiler/config.py（模块 配置层；类别 source；类型 dependency-wiring；符号 to_torch_kwargs, rollout_trace_dir, build_vllm_profiler_args, rollout_profiler_global_ranks）：profiler 配置契约的核心变更点：新增 finish_hook/relocate 字段、schedule 语义改为 mini-batch 粒度，并新增 rollout trace 目录与重定位辅助函数。
- verl/utils/profiler/profile.py（模块 分析核心；类别 source；类型 dependency-wiring；符号 _hook_print, stop, run_finish_hook, _run_finish_hook）：finish hook 与 relocate 流程的落实地：stop() 改为每步 relocate、末尾一次性执行用户命令，并新增 build_rollout_dist_profiler。
- verl/utils/profiler/torch_profile.py（模块 分析调度；类别 source；类型 core-logic；符号 _scheduled_mini_batch_range, _resolve_continuous_schedule_kwargs, _flush_partial_window）：torch profiler 的核心行为修复：schedule 按 mini-batch 推进、关闭 ProfilerStep 行、trace 文件名带 role/step、CPU activity 常开。
- verl/workers/rollout/vllm_rollout/vllm_async_server.py（模块 引擎集成；类别 source；类型 core-logic；符号 start_profile, _should_profile）：rollout 引擎侧集成落地：_should_profile 决定是否驱动引擎 profiler，stop 后重定位 trace 并按全局 rank 过滤。
- verl/trainer/ppo/ray_trainer.py（模块 训练器；类别 source；类型 core-logic；符号 _stop_profiling）：trainer 侧收尾逻辑：_stop_profiling 决定最后一个 profiled step 才执行 finish 命令。
- tests/trainer/ppo/v1/test_rollout_profiling_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 _StubTrainer, on_step_end, on_sample_end, _trainer）：新增的关键测试：验证 trainer 驱动 rollout 引擎 profiler、连续模式不重启、shared worker group 只 profile 一次。
- tests/workers/test_engine_workers_mini_batch_trace_on_cpu.py（模块 测试；类别 test；类型 test-coverage；符号 _engine, _worker, _record_names, fake_record_function）：新增的关键测试：验证更新循环每个 mini-batch 命名与 profiler.step 次数，前向 -only

阶段不推进 profiler。

关键符号：DistProfiler.stop, DistProfiler.run_finish_hook, DistProfiler._run_finish_hook, DistProfiler._run_finish_command, build_rollout_dist_profiler, get_torch_profiler, Profiler._scheduled_mini_batch_range, Profiler._resolve_continuous_schedule_kwargs, Profiler._flush_partial_window, relocate_rollout_traces, rollout_profiler_global_ranks, rollout_trace_dir, vLLMHttpServer._should_profile, vLLMHttpServer.start_profile, vLLMHttpServer.stop_profile, NsightSystemsProfiler.relocate_results, PPOTrainer._stop_profiling, constants_ppo._uses_torch_profiler

关键源码片段

verl/utils/profiler/torch_profile.py

torch profiler 的核心行为修复：schedule 按 mini-batch 推进、关闭 ProfilerStep 行、trace 文件名带 role/step、CPU activity 常开。

```
# verl/utils/profiler/torch_profile.py
# schedule 下 trace 文件会是一个 mini-batch 窗口而不是整个 stage,
# 依据 prof.step_num 反推当前窗口覆盖的 mini-batch 区间用于文件名。
# 该逻辑在 get_torch_profiler 的闭包内，schedule 为传入的调度参数字典。

def _scheduled_mini_batch_range(step_num: int) -> tuple[int, int]:
    # step() 每个 mini-batch 推进一次，因此窗口最后一个 mini-batch 是 step_num - 1；
    # 窗口起点由 schedule 反推，而不是用 active 估算，因为 stop() 可能
    # 在窗口未满时 flush（update loop 提前结束）。
    skip_first = int(schedule.get("skip_first", 0) or 0)
    wait = int(schedule.get("wait", 0) or 0)
    warmup = int(schedule.get("warmup", 0) or 0)
    active = max(int(schedule.get("active", 1) or 1), 1)

    last_mb = max(step_num - 1, 0)
    cycle_len = wait + warmup + active
    cycle = max(last_mb - skip_first, 0) // cycle_len if cycle_len else 0
    # 该 cycle 内真正开始记录的位置：跳过 skipped、idle 与 warmup
    first_mb = skip_first + cycle * cycle_len + wait + warmup
    return min(first_mb, last_mb), last_mb

def _trace_handler(prof):
    idx = handler_state["count"]
    handler_state["count"] += 1
    suffix = ""
    if schedule and name_mini_batch_window:
        step_num = getattr(prof, "step_num", None)
        if isinstance(step_num, int):
            first_mb, last_mb = _scheduled_mini_batch_range(step_num)
            suffix = f"_mb{first_mb}" if first_mb == last_mb else f"_mb{first_mb}-{last_mb}"
        else:
            suffix = f"_part{idx}"
```

```
        elif idx:
            suffix = f"_part{idx}"
        out_path = os.path.join(save_path, f"{base_file_name}{suffix}.json.gz")
        prof.export_chrome_trace(out_path)

    # 所有 trace 直接落在 save_path：role 已进入文件名，不再建子目录，
    # 避免一个 step 的 trace 散落在多个兄弟目录里，也避免 finish_hook_cmd
    # 只看到 save_path 时漏掉部分产物。

    # CPU activity 始终开启：verl 用 record_function 标记每个 stage，
    # 这些标记是 CPU 侧事件，纯 device trace 无法归属到 stage。
    activities = [torch.profiler.ProfilerActivity.CPU]
    if not contents or "cuda" in contents:
        activities.append(torch.profiler.ProfilerActivity.CUDA)

    # 有 schedule 时 prof.step() 每 mini-batch 推进一次，torch 默认会给每个
    # step 边界打 ProfilerStep#<n> 行，但 verl 的 step() 只代表一个 update
    # mini-batch，这些行只会增加噪声，因此关闭记录但保留调度与保存逻辑。
    if schedule:
        prof.record_steps = False
```

verl/workers/rollout/vllm_rollout/vllm_async_server.py

rollout 引擎侧集成落地：_should_profile 决定是否驱动引擎 profiler，stop 后重定位 trace 并按全局 rank 过滤。

```
    # verl/workers/rollout/vllm_rollout/vllm_async_server.py
    # rollout profiler 的 ranks 是全局 GPU rank（与训练角色一致），
    # 需要映射到持有这些 GPU 的 replica，再决定是否驱动该 replica 的引擎。

    def _should_profile(self) -> bool:
        """当前 replica 是否驱动引擎 profiler。"""
        return (
            self.profiler_controller.check_enable()
            and self.profiler_controller.check_this_rank()
            and self.profiler_controller.is_discrete_mode()
        )

    async def start_profile(self, **kwargs):
        if self.node_rank != 0:
            return
        if self._should_profile():
            await self.engine.start_profile(**kwargs)

    async def stop_profile(self):
        if self.node_rank != 0:
            return
        if self._should_profile():
            await self.engine.stop_profile()
            # 引擎把 trace 写在 per-replica 子目录里，stop 后按需搬到 save_path，
```

```
    # 让训练 worker 的 end-of-run 上传一次带走全部 trace；
    # 引擎自身不再执行 finish 命令，避免与训练 worker 重复上传同一目录。
    relocate_rollout_traces(
        self.profiler_controller.config,
        self.replica_rank,
        self.replica_world_size,
        self.profiler_keep_global_ranks,
    )
```

评论区精华

唯一一条 review 评论由 wuxibin89 在 verl/workers/rollout/vllm_rollout/vllm_async_server.py 第 184 行提出："Do we really need to profile rollout(vllm/sglang) in RL framework? Maybe prometheus metrics is a better option." 这是对 " 在 RL 框架内细粒度 profile 推理引擎 " 必要性的设计质疑，建议优先使用 Prometheus 指标。PR 最终仍被 APPROVED，实现保留了 rollout 引擎 profiler 并新增 trace 重定位来收敛产物——说明维护者认可开发期细粒度 profile 与线上监控是不同场景；但评论区没有作者回复，该设计权衡未被完全消解。

- 是否需要在 RL 框架内 profile rollout（vLLM/SGLang）（design）: PR 最终 APPROVED 并合入，实现保留 rollout 引擎 profiler 但增加 relocate_results 收敛产物，表明维护者认可开发期 profiling 的价值；作者未在评论区回复，该设计争议未被完全消解。

风险与影响

- 风险：
  1. 跨模块配置契约变更：ProfilerConfig 新增字段依赖 union/intersect 传播，若其他调用方直接构造配置而未携带新字段，多角色合并时 finish_hook 配置可能丢失；已有测试覆盖但需排查存量构造点。
  2. rollout 引擎路径：vllm_async_server.py 中 _should_profile 只在选中 replica 上配置引擎 profiler，build_vllm_profiler_args 还按 _VLLM_VERSION >= 0.13.0 切换 legacy env 与 CLI 参数，两种通道的兼容性风险较高。
  3. finish_hook 执行：_run_finish_command 在每个选中 rank 跑 /bin/sh -c，命令超时或误写可能挂住训练结尾；代码是 best-effort（异常不打断），但输出流式打印仍可能阻塞。
  4. trace 布局变化：角色从子目录改入文件名，依赖旧路径的脚本会失效；relocate_rollout_traces 按全局 rank 过滤时对未知命名模式采用保守保留策略，不会丢文件但可能多出 tp 同伴的 trace。
  5. schedule 语义变化：continuous 模式下 schedule 前移会丢弃 active 窗口之前的 log-prob/rollout 阶段，用户需要理解 skip_first/wait/warmup 的新含义。 - 影响：对用户：profiler 配置方式新增 relocate_results 与 finish_hook_* 字段，trace 输出路径与命名变化，依赖旧布局的脚本需更新；对开启 profiling 的任务，trainer 会额外启停 rollout 引擎 profiler 并做 trace 迁移，带来可接受的 IO/ 调度开销。对系统：profiler 成为跨 trainer、worker、rollout 引擎的统一生命周期管理，trace 收集自动化后更利于大规模训练的性能复盘。对团队：profiler 子系统 API 规范化，后续 nsys/rollout 等工具的产物收集可复用同一套 relocate/finish_hook 机制。 - 风险标记：跨模块配置契约变

更，rollout 引擎路径改造，新配置字段传播风险，shell 命令执行，trace 布局变化影响脚本

关联脉络

- PR #7422 [rollout] fix: preserve dummy load_format in disaggregated rollout: 同改 vllm_async_server.py、trtllm_async_server.py 与 sglang_rollout 启动链路，与本 PR 的 rollout 引擎 profiler 配置改造处于同一功能线。
- PR #7434 [vllm] fix: vllm always need to resume weights before weight sync: 同改 engine_workers.py 与 vLLM 权重 / 生命周期逻辑，本 PR 也在 engine_workers.py 中调整了 profiler.step 与 mini_batch 标注。
- PR #7413 [sglang] fix: lora sglang e2e: 同改 async_sglang_server.py 与 sglang_rollout 引擎链路，均属于 rollout 引擎侧启动与行为修复的连续演进。