

PR #3172 完整报告

radixark/miles

[feat] snapshot eval during --debug-train-only by reusing the eval dispatcher

合并时间: 2026-09-15 12:17

原文链接: <http://prhub.com.cn/radixark/miles/pull/3172>

执行摘要

- 一句话: 纯 SFT train-only 复用 EvalDispatcher 支持 checkpoint 快照评估
- 推荐动作: 值得精读。三个设计决策值得关注: 一是“复用现有 EvalDispatcher 而非新增 train-only 专用路径”, 避免平行实现漂移; 二是把引擎启动决策派生为 args 上的单一字段, 删除散落在 sglang_config/specs/controller 里的多处短路, 降低不一致风险; 三是用“显式校验 + 强制 3 个 eval 点的 e2e”代替静默跳过, 任何短路都会让 CI 失败。后续 3 个修复 commit (resume id、final force、router sizing) 展示了边界条件的排查思路, 可作为同类接入的参考。

功能与动机

PR body 明确说明动机: "Let a pure-SFT --debug-train-only run evaluate its checkpoints with the existing snapshot-eval machinery (EvalDispatcher/EvalFleet), instead of adding a train-only-specific eval path" (让纯 SFT 的 --debug-train-only 运行用现有 snapshot-eval 机制评估 checkpoint, 而不是新增 train-only 专用 eval 路径)。此前 train-only 模式会关闭所有推理路径, 纯 SFT 无法评估 checkpoint; 该 PR 选择复用而非平行实现, 并取代 #2882。

实现拆解

实现按以下 5 步展开:

1. 引擎启动决策集中化 (miles/utils/arguments.py + miles/backends/sglang_utils/sglang_config.py + miles/ray/specs/inference.py) :
miles_validate_args 在 --debug-train-only 下先归零 rollout_num_gpus (放在 colocate 规范化之后, 避免 colocate 把 actor GPU 数写回), 再派生新字段
starts_inference_engines = not debug_train_only or eval_num_gpus > 0。
_compute_raw_sglang_config 与 _compute_rollout_offset 改读该字段: 纯 train-only 且无 eval fleet 时返回空模型配置、offset 为 0; 有 eval fleet 时只构建 eval 模型, 且 eval bundle 排在 actor bundles 之后。specs_router 与 specs_inference_engine 删掉各自的 debug_train_only 短路, 引擎启停完全由配置文件决定。
2. train.py 接入 EvalDispatcher (train.py) : 新增 eval_dispatcher =
EvalDispatcher(args, actor_model, rollout_executor); eval-only 特例和训练前基线评估都改走 eval_dispatcher.dispatch, 训练循环内的周期 eval 改为 dispatch(rollout_id, force=rollout_id == num_rollout - 1), 并给 should_run_periodic_action 补上

`args.num_rollout` 参数（否则最后一轮 cadence 不命中，force 永不触发）；循环后调用 `eval_dispatcher.drain()` 排空异步 eval。

3. 边界修复：resume 场景从 `dispatch(start_rollout_id)` 改为 `dispatch(start_rollout_id - 1)`，评估已完成 rollout，避免 checkpoint-reuse 模式请求不存在的 checkpoint（`ckpt_missing`）以及 staging 模式把快照导出到第一个恢复迭代即将覆盖的 `step_N` 目录；`EvalDispatcher._export` 在 `--offload-train` 下对 actor 做 onload/offload，对齐 `save_training_model`；`compute_miles_router_config` 新增 `num_engines` 参数，`max_connections` 从 `sglang_server_concurrency * num_engines` 计算，避免 train-only 下 `rollout_num_gpus=0` 导致连接池为 0、每个 eval 引擎健康检查超时、所有点退化为 `skipped_unhealthy`；`init_http_client` 在 snapshot eval 开启时始终创建客户端，且无 rollout GPU 时按 eval fleet（`eval_num_gpus // eval_num_gpus_per_engine`）计算并发。
4. 参数校验收紧（`miles/utils/arguments.py`）：snapshot eval 校验从“禁止 `debug_train_only/debug_rollout_only`”放宽为显式排除 `--debug-rollout-only` 与 `--load-debug-rollout-data`（replay 路径不加载 rollout 函数，无物可 eval），并在 `--debug-train-only` 下强制 `eval_function_path != rollout_function_path`（SFT 训练函数只消费 message 列表，不能评估 HF snapshot）。
5. 测试配套：新增 `e2e tests/e2e/short/test_qwen3_0.6B_sft_snapshot_eval.py`（stage-c-2-gpu-h200, labels short/eval/megatron, 声明 hopper/blackwell 硬件），metric checker 用 `--ci-metric-checker-expect-num 3` 强制恰好 3 个 eval 点；fast 侧新增 `TestSnapshotEvalValidation`、`TestInitHttpClientConcurrency`、`TestEvalBeforeTrain`、`TestFinalEval`，并更新 `test_sglang_config.py` 的 `TestRolloutOffset`、`test_rollout_server.py`、`test_checkpoint_eval.py`、`test_rollout_executor.py`、`test_placement_group_shared_ppo.py`、`test_config.py`。

关键文件：

- `train.py`（模块 训练驱动；类别 source；类型 core-logic；符号 train）：同步训练主入口的 eval 驱动改造：eval-only、训练前基线、周期 eval 全部从 `rollout_executor.eval.remote` 切换为 `EvalDispatcher.dispatch`，新增 drain 收尾，并修正 eval cadence 需 `num_rollout` 参数才能让最终 force 生效的边界问题。
- `miles/utils/arguments.py`（模块 参数校验；类别 source；类型 core-logic；符号 `miles_validate_args`）：引擎启动决策的新判据源与 snapshot eval 校验重构：`--debug-train-only` 下强制 `rollout_num_gpus=0` 并派生 `starts_inference_engines`；快照 eval 从完全禁止 debug 模式改为显式拒绝 `debug_rollout_only/load-debug-rollout-data`，并要求 train-only 显式指定 eval function。
- `miles/backends/sglang_utils/sglang_config.py`（模块 引擎配置；类别 source；类型 core-logic；符号 `_compute_raw_sglang_config`，`_compute_rollout_offset`）：引擎配置生成的核心：以 `starts_inference_engines` 取代 `debug_train_only` 短路，train-only 无 eval fleet 时不建任何模型、有 eval fleet 时只建 eval 模型；rollout offset 仅在无推理引擎时为 0，否则 eval fleet 排在 actor bundles 之后。
- `miles/ray/specs/inference.py`（模块 引擎规格；类别 source；类型 core-logic；符号 `specs_router`，`specs_inference_engine`，`_compute_launch_command`）：删除 `specs_router/specs_inference_engine` 各自的 `debug_train_only` 空返回，引擎启停完全

由 `resolve_sglang_config` 决定；同时把 `router` 配置改为按引擎数 (`num_server_cells`) 计算 `max_connections`。

- `miles/ray/rollout/eval_dispatch.py` (模块 评估调度；类别 `source`；类型 `core-logic`；符号 `EvalDispatcher._export`, `EvalDispatcher.dispatch`, `EvalDispatcher.drain`) : `EvalDispatcher._export` 在 `--offload-train` 下对 `actor` 做 `onload/offload`，对齐 `save_training_model`，使快照导出与 `offload` 训练共存，是 `train-only` 快照 `eval` 能跑通 `--offload-train` 组合的关键。
- `miles/router/config.py` (模块 请求路由；类别 `source`；类型 `core-logic`；符号 `compute_miles_router_config`) : `compute_miles_router_config` 新增 `num_engines` 参数，`max_connections` 从 `sglang_server_concurrency * num_engines` 计算，修复 `train-only` 下 `rollout_num_gpus=0` 导致连接池为 0、所有 `eval` 引擎健康检查超时的问题。
- `miles/utils/http_utils.py` (模块 网络客户端；类别 `source`；类型 `core-logic`；符号 `init_http_client`) : `init_http_client` 在 `snapshot eval` 开启时始终创建客户端，且在没有 `rollout GPU` 时按 `eval fleet` 大小计算并发，保证 `train-only + eval fleet` 的评估请求有可用连接池。
- `tests/e2e/short/test_qwen3_0.6B_sft_snapshot_eval.py` (模块 端到端测试；类别 `test`；类型 `test-coverage`；符号 `_write_datasets`, `prepare`, `execute`) : 新增端到端验证：Qwen3-0.6B 纯 SFT + 1 卡 `eval fleet`，`metric checker` 用 `--ci-metric-checker-expect-num 3` 强制恰好 3 个 `eval` 点，任何引擎或 `eval` 路径上的 `train-only` 短路都会失败，是本次功能唯一的 GPU 级证据。
- `tests/fast/test_train.py` (模块 驱动测试；类别 `test`；类型 `test-coverage`；符号 `TestEvalBeforeTrain`, `TestFinalEval`) : 用 `driver fakes` 锁定 `train.py` 的 `eval` 时序：`fresh run` 先 `eval step 0` 再 `rollout`；`resume run` 评估已完成 `rollout` (`start_rollout_id - 1`) 而非下一个；最后一轮无论 `cadence` 都 `force` 评估。
- `tests/fast/utils/test_arguments.py` (模块 参数测试；类别 `test`；类型 `test-coverage`；符号 `TestSnapshotEvalValidation`) : `TestSnapshotEvalValidation` 覆盖新的参数校验：拒绝 `--load-debug-rollout-data`、`train-only` 必须显式指定 `eval function`、`colocate` 下 `rollout_num_gpus` 仍为 0。

关键符号：`train`, `miles_validate_args`, `_compute_raw_sglang_config`, `_compute_rollout_offset`, `specs_router`, `specs_inference_engine`, `compute_miles_router_config`, `init_http_client`, `EvalDispatcher.dispatch`, `EvalDispatcher._export`, `EvalDispatcher.drain`, `test_fresh_run_evaluates_the_initial_checkpoint_at_step_zero`, `test_resumed_run_evaluates_the_completed_rollout_not_the_next_one`, `test_the_last_rollout_is_always_evaluated_even_off_cadence`

关键源码片段

`train.py`

同步训练主入口的 `eval` 驱动改造：`eval-only`、训练前基线、周期 `eval` 全部从 `rollout_executor.eval.remote` 切换为 `EvalDispatcher.dispatch`，新增 `drain` 收尾，并修正 `eval cadence` 需 `num_rollout` 参数才能让最终 `force` 生效的边界问题。

```

# train.py: train() 中 eval 路径统一接入 EvalDispatcher
# (以下整理 eval 相关主干, 训练循环与 offload 细节用 ... 省略)

# EvalDispatcher 持有 actor 与 rollout executor, 负责 snapshot 导出、
# 异步派发与快照 GC。train_async.py 已用它, 这里让 train.py 复用同一机制
eval_dispatcher = EvalDispatcher(args, actor_model, rollout_executor)

# eval-only 特例: 不训练, 只对 --hf-checkpoint 基线评估一次
if args.num_rollout == 0 and args.eval_interval is not None:
    await inference_controller.prepare_eval()
    await eval_dispatcher.dispatch(0, hf_dir=args.hf_checkpoint)

# 训练前基线评估: 移到训练循环外。fresh run 评估 step 0 (--hf-checkpoint);
# resume run 评估已完成的 checkpoint (start_rollout_id - 1), 而不是下一个
# rollout——下一个 rollout 的 checkpoint 尚不存在 (ckpt_missing),
# 且 staging 模式下会把快照导出到第一个恢复迭代即将覆盖的 step_N 目录
if (
    args.num_rollout > args.start_rollout_id
    and args.eval_interval is not None
    and not args.skip_eval_before_train
):
    await inference_controller.prepare_eval()
    if args.start_rollout_id == 0:
        await eval_dispatcher.dispatch(0, hf_dir=args.hf_checkpoint)
    else:
        await eval_dispatcher.dispatch(args.start_rollout_id - 1)

for rollout_id in range(args.start_rollout_id, args.num_rollout):
    ...
    # 周期 eval: cadence 判断必须带上 num_rollout, 否则最后一轮
    # 永远无法命中检查点, dispatch(force=True) 也就不可达
    if should_run_periodic_action(
        rollout_id, args.eval_interval, num_rollout_per_epoch, args.num_rollout
    ):
        await inference_controller.prepare_eval()
        await eval_dispatcher.dispatch(rollout_id, force=rollout_id == args.num_rollout - 1)
    ...

# 训练结束前排空所有异步 eval 任务, 避免快照评估未完成就拆散资源
await eval_dispatcher.drain()

```

miles/utils/arguments.py

引擎启动决策的新判据源与 snapshot eval 校验重构: --debug-train-only 下强制 rollout_num_gpus=0 并派生 starts_inference_engines; 快照 eval 从完全禁止 debug 模式改为显式拒绝 debug_rollout_only/load-debug-rollout-data, 并要求 train-only 显式指定 eval function。

```

# miles/utils/arguments.py: miles_validate_args 中引擎启动决策与 snapshot-eval 校验

```

```

# colocate 规范化会把 actor 的 GPU 数写回 rollout_num_gpus,
# 因此 train-only 的归零必须放在它之后
if args.colocate:
    if args.rollout_num_gpus != args.actor_num_gpus_per_node * args.actor_num_nodes:
        args.rollout_num_gpus = args.actor_num_gpus_per_node * args.actor_num_nodes

# 引擎启动决策: train-only 不开 rollout 引擎; 若配置了 eval fleet
# (--eval-num-gpus > 0) 则仍会启动评估引擎。该派生字段取代散落在
# sglang_config / specs / controller 里的 debug_train_only 短路,
# 成为各处读取的单一事实来源
if args.debug_train_only:
    args.rollout_num_gpus = 0
args.starts_inference_engines = not args.debug_train_only or args.eval_num_gpus > 0

...

# snapshot eval 参数校验: 从“禁止 debug 模式”放宽为显式排除无法共存的
# 调试组合, 并强制 train-only 自己提供可评估 snapshot 的 eval 函数
if args.eval_uses_snapshots:
    ...
    assert not args.debug_rollout_only, "Snapshot eval is not supported with debug_rollout_
    only."
    assert args.load_debug_rollout_data is None, (
        "Snapshot eval is not supported with --load-debug-rollout-data: "
        "no rollout functions are loaded."
    )
    # SFT 训练函数只消费 message 列表, 不能评估 HF snapshot,
    # 所以 train-only 下必须显式指定 fleet-aware 的 --eval-function-path
    if args.debug_train_only:
        assert args.eval_function_path != args.rollout_function_path, (
            "Snapshot eval during --debug-train-only requires an explicit "
            "--eval-function-path; the training rollout function cannot evaluate snapshots."
        )

```

miles/backends/sglang_utils/sglang_config.py

引擎配置生成的核心: 以 starts_inference_engines 取代 debug_train_only 短路, train-only 无 eval fleet 时不建任何模型、有 eval fleet 时只建 eval 模型; rollout offset 仅在无推理引擎时为 0, 否则 eval fleet 排在 actor bundles 之后。

miles/backends/sglang_utils/sglang_config.py: 引擎配置按派生字段生成

```

def _compute_raw_sglang_config(args) -> _RawSglangConfig:
    # 没有任何引擎需要启动: 纯 train-only 且未配置 eval fleet,
    # 直接返回空配置, 全程不落 sglang 进程
    if not args.starts_inference_engines:
        return _RawSglangConfig(models=[])

    eval_num_gpus = args.eval_num_gpus
    rollout_num_gpus = args.rollout_num_gpus or 0

```

```

# 显式 YAML 配置时, GPU 总数必须等于 rollout 加 eval 的 GPU 数;
# 归零的 rollout_num_gpus 用 0 参与求和, 保持校验依然成立
if getattr(args, "sglang_config", None) is not None:
    config = _RawSglangConfig.from_yaml(args.sglang_config)
    expected = rollout_num_gpus + eval_num_gpus
    actual = config.total_num_gpus
    assert actual == expected, "YAML 配置的 GPU 总数必须等于 rollout 加 eval 的 GPU 数"
    ...

# train-only + eval fleet: 不建任何 rollout 模型, 只保留 eval 模型配置;
# prefill 服务器分支只在有 rollout 引擎时才有意义
if rollout_num_gpus == 0:
    config = _RawSglangConfig(models=[])
elif args.prefill_num_servers is not None:
    config = _RawSglangConfig.from_prefill_num_servers(args)
else:
    config = _RawSglangConfig(...)
...
return config

```

```

def _compute_rollout_offset(args) -> int:
    """Offset (PG bundle 槽位) 表示 rollout GPU 从哪个位置开始占位。"""
    # 无任何推理引擎时不需要为 rollout 预留位置; debug_rollout_only 与
    # colocate 同样返回 0。train-only + eval fleet 时此处不为 0,
    # eval bundle 会排在 actor bundle 之后而不是抢占前段
    if args.debug_rollout_only or args.colocate or not args.starts_inference_engines:
        return 0
    if getattr(args, "critic_train_only", False):
        return args.critic_num_nodes * args.critic_num_gpus_per_node
    ...

```

评论区精华

唯一实质性 review 评论来自 yueming-yuan (P2 级别), 在 resume eval 语义上提出了正确性问题:

Use the completed rollout's version for resume eval — train.py:

1. On resume, start_rollout_id is the loaded checkpoint's rollout plus one. Dispatching that next ID has two consequences: checkpoint-reuse mode requests a nonexistent checkpoint, producing ckpt_missing; staging mode exports under the same step_N directory that the first resumed training iteration may subsequently overwrite. With --eval-interval 1, I reproduced both pending evaluations reading the post-training weights, losing the intended pre-training measurement. Use the loaded checkpoint's completed rollout ID.

该问题在 commit 17f47bb 中修复: 训练前基线评估改为 `dispatch(args.start_rollout_id - 1)`, 并由 `tests/fast/test_train.py::test_resumed_run_evaluates_the_completed_rollout_not_t`

he_next_one 锁定行为。yueming-yuan 最终 APPROVED, 评价 "LGTHuman, with one codex comment"。

- Resume 时 snapshot eval 应使用已完成 rollout 的版本 (P2) (correctness): commit 17f47bb 修复为评估已完成 rollout (start_rollout_id - 1), 并用 test_resumed_run_evaluates_the_completed_rollout_not_the_next_one 锁定行为: eval:2 必须先于 prepare_rollout:3, 且 prepare_rollout:3 之前不得出现 eval:3。

风险与影响

- 风险:
 1. 核心训练入口变更 (train.py) : 所有走同步 train.py 的训练 (不限于 debug-train-only) 的 eval 路径都从 rollout_executor.eval.remote 切换为 EvalDispatcher.dispatch, 异步语义 (force、drain、快照导出时序) 与旧路径不同, 普通 RL 训练也有回归暴露面; fast 测试覆盖了 cadence 与 resume, 但多引擎 GPU 场景仍需观察。
 2. resume eval 语义变化: resume 时从此前准备评估 start_rollout_id 改为 start_rollout_id - 1, 依赖旧行为的用户会看到评估点编号变化; 在 staging 模式下旧行为会覆盖即将重写的快照目录, 因此这是必要修复, 但属于行为变更。
 3. router 连接数计算变化 (miles/router/config.py + miles/ray/specs/inference.py) : max_connections 从 sglang_server_concurrency * rollout_num_gpus // rollout_num_gpus_per_engine 改为 sglang_server_concurrency * num_engines (model_cfg.num_server_cells), 影响所有使用 --use-miles-router 的运行, 不只 train-only; 需要验证与 rollout 数量不一致的配置下连接池是否符合预期。
 4. 新派生字段的一致性依赖: starts_inference_engines 依赖 miles_validate_args 先于 specs/config 执行; 任何绕过该校验的入口或测试 fixture 都会拿到未派生字段 (commit b90cb4a 就修了一个漏掉的 fixture), 后续新增入口需注意。
 5. 参数校验收紧: --debug-train-only 现在强制 rollout_num_gpus=0 (含 colocate 场景), 快照 eval 拒绝 --load-debug-rollout-data, 这些组合此前可能“静默可用”, 现在会直接断言失败。- 影响: 功能上, 纯 SFT (--debug-train-only) 用户现在可配置 --eval-num-gpus 获得与 async 训练一致的 snapshot export、异步调度与快照 GC; 使用上, train-only 下必须显式提供 fleet-aware 的 --eval-function-path, 且 eval prompt 需要预渲染列 (--eval-input-key), --apply-chat-template 不能用于 SFT 训练数据的 eval 提示。系统层面, train.py 与 train_async.py 的 eval 路径收敛到同一个 EvalDispatcher, 未来维护单一路径; train.py 的 eval-before-train 语义 (含 resume) 被修正并有 fast 测试锁定。影响范围包括所有走 train.py 的同步训练 (eval 路径) 以及所有使用 miles router 的运行 (连接数计算变化)。- 风险标记: 核心训练入口变更, resume eval 语义变化, router 连接数计算变化, 新派生字段一致性依赖, eval 路径行为差异

关联脉络

- PR #2882 (被本 PR 取代的先前方案, PR body 标注 Supersedes #2882) : PR body 明确本 PR 取代 #2882 (新增 train-only 专用 eval 路径的先前尝试), 本 PR 改为复用

EvalDispatcher 机制，避免平行实现。

- PR #3129 fix: guard connection allocations during offloaded weight sync: 同属 offload 权重同步边界：本 PR 让 EvalDispatcher._export 在 --offload-train 下 onload/offload actor (对齐 save_training_model) , 与 #3129 对 actor.py 中 offload 权重同步阶段连接分配的内存守卫相邻，可关注同一路径上的内存语义一致性。