

PR #2806 完整报告

radixark/miles

examples: in-process Harbor agent function

合并时间: 2026-09-09 06:14

原文链接: <http://prhub.com.cn/radixark/miles/pull/2806>

执行摘要

- 一句话: 新增进程内 Harbor 代理函数, 让 rollout worker 直接在云端沙箱运行 Harbor 试验。
- 推荐动作: 该 PR 值得精读。它引入了重要的新架构路径, 并展示了良好的设计模式: 将配置构建、试验执行和结果映射职责清晰分离。测试策略 (使用 fake 包进行快速迭代, 同时用真实包进行契约验证) 尤其值得借鉴。核心实现 harbor_agent_function.py 的注释详尽, 解释了每个决策的权衡 (如为什么 abort 策略优于 recover)。审阅者发现的 total_tool_time 缺失和 HARBOR_AGENT_MAX_ITERATIONS=0 问题是实际集成中可能出现的真实 bug, 具有学习价值。

功能与动机

为实现更轻量、直接的 agentic 训练, 需要让 rollout worker 能自身驱动 Harbor 试验, 而不是依赖额外的代理服务器。PR body 明确指出: 'Let a rollout worker run a Harbor trial itself when the task sandbox is on a cloud backend'。这消除了本地 Docker 依赖, 利用云端沙箱的网络可达性, 简化了架构并降低了运维复杂度。

实现拆解

1. 核心入口实现: 新增 examples/experimental/harbor/harbor_agent_function.py, 包含主函数 run() 和配置构建函数 build_trial_config()。run() 是 agentic_tool_call.generate 的入口点, 负责构建配置、运行试验、处理超时并返回结果字典。
2. 绑定与配置: 定义了 HarnessBinding 数据类和 HARNESS_BINDINGS 字典, 为每个支持的 agent (terminus-2, claude-code, opencode 等) 提供特定的 kwargs 和 env 配置逻辑。例如, _terminus_kwargs 确保 API 密钥通过 llm_kwargs 传递, 并设置 response_length_exceeded_policy 为 'abort'。
3. 环境与凭证处理: 通过 _environment_config() 和 _ensure_provider_key() 函数, 将环境变量 (如 HARBOR_ENV_TYPE) 直接映射到 Harbor 的 EnvironmentType, 并解析启动器通过文件路径传递的密钥。
4. 结果映射: trial_result_to_metadata() 函数将 Harbor 的试验结果 (包括奖励、耗时、异常) 转换为训练管道期望的样本元数据格式, 并定义了异常到 exit_status 的映射规则 (如 AgentTimeoutError -> 'TimeLimitExceeded')。

5. 测试配套：新增两个测试文件：test_harbor_agent_function.py 使用 fake harbor 包进行离线单元测试，覆盖配置构建、结果映射和入口行为；test_harbor_contract.py 是针对真实 harbor 包的契约测试，确保构建的配置能被 Harbor 的 Pydantic 模型验证通过。

关键文件：

- examples/experimental/harbor/harbor_agent_function.py (模块 代理函数；类别 source；类型 dependency-wiring；符号 run, build_trial_config, HarnessBinding, HARNESS_BINDINGS)：核心实现文件，定义了进程内 Harbor 试验的完整流程，包括配置构建、试验运行、凭证解析和结果映射。
- tests/fast/examples/experimental/harbor/test_harbor_agent_function.py (模块 代理函数测试；类别 test；类型 test-coverage；符号 FakeTrial, fake_harbor, test_environment_type_is_passed_straight_through, test_terminus_2_binding_aborts_on_truncation_and_carries_sampling_params)：核心单元测试，通过 fake harbor 包验证配置构建、结果映射和入口行为，无需真实依赖。
- tests/fast/examples/experimental/harbor/test_harbor_contract.py (模块 契约测试；类别 test；类型 test-coverage；符号 test_the_pass_through_environment_types_exist, test_every_binding_names_a_real_harbor_agent, test_the_built_config_is_a_valid_trial_config, test_the_result_fields_the_mapping_reads_still_exist)：契约测试，验证构建的配置能被真实的 Harbor Pydantic 模型接受，防止与 Harbor 包的 API 漂移。

关键符号：run, build_trial_config, trial_result_to_metadata, _terminus_kwargs, _claude_code_kwargs, _opencode_kwargs, _ensure_provider_key, _environment_config

关键源码片段

examples/experimental/harbor/harbor_agent_function.py

核心实现文件，定义了进程内 Harbor 试验的完整流程，包括配置构建、试验运行、凭证解析和结果映射。

```
# --- harness bindings -----
# 定义每个 harness 需要的连接配置：如何传递 session URL 和 API 密钥。
# 此表是与 harbor-framework/harbor 的同步锚点。
@dataclass(frozen=True)
class HarnessBinding:
    # (session_url, api_key, sampling_params, model, max_seq_len) -> AgentConfig.kwargs
    kwargs: Callable[[str, str, dict[str, Any], str, int | None], dict[str, Any]]
    # (session_url, api_key) -> AgentConfig.env
    env: Callable[[str, str], dict[str, str]]
    # the model name Harbor passes to the harness
    model_name: Callable[[str], str] = lambda model: model

# terminus-2 配置：必须设置 abort 策略以避免与 TITO session server 的 desync。
def _terminus_kwargs(
    session_url: str, api_key: str, sampling_params: dict[str, Any], model: str, max_seq_len: int |
```

```

None
) -> dict[str, Any]:
    kwargs: dict[str, Any] = {
        "parser_name": "xml",
        "interleaved_thinking": True,
        "abort_on_response_length_exceeded": True,
        "llm_call_kwargs": dict(sampling_params),
        "api_base": session_url,
        # Terminus2 has no top-level api_key parameter: the key must ride
        # llm_kwargs (the LLM constructor extras, spread into every litellm
        # call) or litellm falls back to OPENAI_API_KEY in the process env.
        "llm_kwargs": {"api_key": api_key},
        "enable_summarize": _env_flag("HARBOR_TERMINUS_2_ENABLE_SUMMARIZE"),
        "trajectory_config": {"linear_history": _env_flag("HARBOR_TERMINUS_2_LINEAR_HISTORY")
        },
        # "abort" ends the trajectory when one reply is truncated at max_tokens;
        # upstream's "recover" re-sends the truncated reply as a fabricated
        # assistant turn, which desyncs the TITO session server.
        "response_length_exceeded_policy": os.getenv("HARBOR_RESPONSE_LENGTH_POLICY",
        "abort"),
    }
    if max_turns := _env_int("HARBOR_AGENT_MAX_ITERATIONS"):
        kwargs["max_turns"] = max_turns
        kwargs["suppress_max_turns_warning"] = True
    return kwargs

```

```

# OpenCode 需要重命名 provider id 以避免使用 Responses API。
_HARNESS_BINDINGS: dict[str, HarnessBinding] = {
    "terminus-2": HarnessBinding(kwargs=_terminus_kwargs, env=_openai_env),
    "claude-code": HarnessBinding(kwargs=_claude_code_kwargs, env=_claude_code_env),
    "opencode": HarnessBinding(
        kwargs=_opencode_kwargs,
        env=_opencode_env,
        model_name=lambda model: "/" .join(_opencode_provider_model(model)),
    ),
    # ...
}

```

tests/fast/examples/experimental/harbor/test_harbor_agent_function.py

核心单元测试，通过 fake harbor 包验证配置构建、结果映射和入口行为，无需真实依赖。

```

# --- result mapping -----
# 测试 Harbor 异常如何映射到训练管道的 exit_status 词汇。
@pytest.mark.parametrize(
    ("exc_type", "exit_status"),
    [
        ("AgentTimeoutError", "TimeLimitExceeded"),
        ("EnvironmentStartTimeoutError", "TimeLimitExceeded"),
    ]
)

```

```

        ("SingleTurnMaxSeqLenExceededError", "SequenceLengthLimitExceeded"),
        ("RuntimeError", "AgentError"),
    ],
)
def test_harbor_exceptions_map_to_the_exit_status_vocabulary(exc_type, exit_status):
    result = SimpleNamespace(
        exception_info=SimpleNamespace(exception_type=exc_type), verifier_result=None, agent_
        result=None
    )
    out = haf.trial_result_to_metadata(result)
    assert out["reward"] == 0.0
    assert out["exit_status"] == exit_status

# --- entry -----
# 测试主入口函数：配置错误应抛出异常，而不是返回零分。
def test_run_raises_on_a_missing_task_instead_of_scoring_zero(tasks_dir, fake_harbor):
    """A config error fails every sample; raising beats training on silent all-zero rewards."""
    with pytest.raises(FileNotFoundError):
        run_async(haf.run("http://s/sessions/s1", [], {}, {"instance_id": "missing"}))

```

评论区精华

1. URL 后缀冲突：claude[bot] 发现 ANTHROPIC_BASE_URL 会因 session URL 已带 /v1 后缀而与 SDK 自身添加的 /v1/messages 冲突，导致 404。nblintao 后续修复此问题，使用 session_url.removesuffix("/v1")。
 2. 环境变量值为 0 的处理：claude[bot] 指出 if max_turns := _env_int("HARBOR_AGENT_MAX_ITERATIONS") 的写法会将值为 0 的情况视为 falsy，导致配置被忽略。这是一个潜在的语义错误。
 3. 缺失关键指标：claude[bot] 指出 agent_metrics 中未提供 total_tool_time 字段，会导致上游的吞吐量计算 (non_generation_time) 始终为 0，造成性能指标失真。
 4. Daytona 资源清理：Shi-Dong 询问 Daytona 后端是否需要自动清理参数，nblintao 确认需要，并提到有单独 PR (#2838) 处理此问题。
 5. 设计决策：配置错误（如未知环境类型、缺失任务目录）选择直接抛出异常，而不是返回零分。PR body 解释这比“silent all-zero rewards”更优，能快速失败，避免无意义的训练。
- ANTHROPIC_BASE_URL 的 /v1 后缀冲突 (correctness): 作者回复“fixed”，在后续提交中使用 session_url.removesuffix("/v1") 修复。
 - HARBOR_AGENT_MAX_ITERATIONS=0 的 falsy 语义 (correctness): 未解决。该问题在 PR 评论中被标记为 nit，但未在本次合并中修复。
 - agent_metrics 中缺失 total_tool_time 指标 (performance): 未解决。该问题被标记为 optional，但会影响监控和资源评估的准确性。
 - Daytona 后端是否需要自动清理参数 (question): 作者确认需要，并提到有单独的 PR (#2838) 处理此问题。

风险与影响

- 风险:

1. 集成风险: `total_tool_time` 指标的缺失会影响训练过程中的吞吐量监控和资源评估, 可能掩盖真实的性能问题。
2. 配置错误行为: 未知 `HARBOR_ENV_TYPE` 直接抛异常 (不同于 `agent-server` 的静默回退到 `docker`) 是更安全的设计, 但需要确保所有调用方已知悉此变更。
3. 依赖风险: 该实现严重依赖 Harbor 分支 (`harbor-miles-*`) 的特定 API (如 `response_length_exceeded_policy`、`MaxSeqLenExceededError`)。Harbor 上游变更可能导致不兼容。
4. 状态管理: `FakeTrial` 类使用类级列表 `created` 来追踪试验创建, 这在并行测试中可能导致状态污染, 尽管目前是串行运行。
 - 影响: 用户影响: 为 Miles 用户提供了运行 `agentic` 试验的新选项, 特别适用于云端沙箱环境, 降低了本地部署门槛。系统影响: 在 `rollout worker` 中引入了新的依赖路径 (`harbor` 包), 并扩展了 `agentic_tool_call.generate` 的功能。这是 `agentic` 训练基础设施的重要扩展。团队影响: 该 PR 是多 PR 重构系列的一部分 (#2805, #2804, #2836, #2837), 为 Harbor 集成的完整实现奠定了基础。测试策略 (`fake` 包 + 契约测试) 为类似集成提供了参考。

- 风险标记: 核心路径变更, 外部依赖漂移, 监控指标缺失

关联脉络

- PR #2805 Stacked PR (referenced in body): 本 PR 明确声明 'Stacked on #2805', 表明其依赖于该 PR 的改动。
- PR #2804 Tracking issue (referenced in body): 本 PR 的跟踪 issue, 用于记录整个 Harbor 集成工作的进展。
- PR #2836 Launcher (referenced in body): 本 PR 明确说明 'The launcher is #2836', 该 PR 将包含启动器的实现。
- PR #2837 README/docs (referenced in body): 本 PR 明确说明 'README/docs are #2837', 该 PR 将包含文档更新。
- PR #2876 ci(e2e): one GRPO step with Harbor trials on real e2b sandboxes (manual for now): 关联的 e2e 测试 issue, 验证了本 PR 实现路径的端到端功能, 并报告了本 PR 修复的集成 bug。
- PR #2838 Daytona support (referenced in discussion): 作者在回复 Shi-Dong 时提到, 支持 Daytona 的具体实现位于此单独的 PR 中。