PR #2274 完整报告

radixark/miles

refactor(openenv): move duplicated sandbox helpers into the TB2 recipe

合并时间：2026-08-10 11:05

原文链接：http://prhub.com.cn/radixark/miles/pull/2274

执行摘要

- 一句话：沙箱辅助函数上收共享 recipe，统一多 provider 资源策略
- 推荐动作：值得精读，尤其是三个设计决策：run_with_deadline 对 with 块的论证（线程池 __exit__ 会 join 被放弃的调用）、配置归属原则（描述一个沙箱的 knob 放 create 旁、fan-out knob 放后端）、异常清理的 try/except 保护模式。建议结合栈的 3/3（#2276 Modal 后端）一起看完整演进，并留意后续 PR 是否补充 provider 层单位映射测试。

功能与动机

PR body 明确说明这是 3 步栈的 1/3，独立成立：Daytona 和 E2B 物化各有两份既非 provider 特有、漂移又有后果的副本。task_env_resources 的地板是 recipe 策略（env server 需要与任务并行运行的余量），但 Daytona 副本以 GB 应用（max(2, memory_mb // 1024)）、其他以 MB 应用，同一 task.toml 在不同 provider 上可能被不同地遵守，已经出现实际分歧；run_with_deadline 中“为什么不能是 with 块”的推理也只应存在一份。

实现拆解

1. 上收共享 helper（examples/experimental/openenv/tb2_sandbox_recipe.py）：新增 task_env_resources(task_dir)，从 task.toml [environment] 读取 cpus / memory_mb / storage_mb 并统一应用地板（1 / 2048 / 10240），把“地板是 recipe 策略”固定在单一位置；新增 run_with_deadline(fn, timeout_s)，用单 worker ThreadPoolExecutor 包裹阻塞型 provider 调用，超时抛 TimeoutError 且 shutdown(wait=False) 不等待被放弃的调用；新增模块级常量COMMAND_TIMEOUT_S（读TB2_COMMAND_TIMEOUT_S，默认900），server_cmd 默认参数从字面量 900 改为引用它；并将 _env_src_dir / wait_server_ready 的第三方依赖改为函数内 import，保证离线测试无需安装 tbench2_env 和 requests。
2. 重构 Daytona 物化（tb2_sandbox_daytona.py）：task_resources 删除本地读取 task.toml 的副本，改用 task_env_resources 的 MB 返回值再做 GB 整数除法映射；create_task_sandbox 的 command_timeout_s / ready_timeout_s 默认值改为 COMMAND_TIMEOUT_S 与 _READY_TIMEOUT_S（新环境变量 OPENENV_DAYTONA_READY_TIMEOUT_S），knob 移到 create 旁并写明“描述一个沙箱的配置归属 create、fan-out 配置归属后端”的原则；auto_stop / auto_delete 刻意保持参数而非环境变量，因为它们与心跳节奏耦合。

3. 重构 E2B 物化（tb2_sandbox_e2b.py）：task_build_resources 改用 task_env_resources 后只保留 cpu / memory 字段（E2B 在模板构建时定资源规格）；ensure_task_template 内联的线程池超时逻辑替换为 run_with_deadline 调用；create_task_sandbox 删除 ttl_s 参数——keepalive 线程固定按 _SANDBOX_TTL_S 刷新，外部传值会在第一次心跳被静默覆盖，删除参数消除无效配置；新增 OPENENV_E2B_READY_TIMEOUT_S。
4. 异常路径加固：两个 provider 的失败创建清理（daytona.delete / sandbox.kill）都包上 try/except，清理自身失败不再掩盖 create 失败的真实异常，由 provider TTL / auto-delete 兜底回收。
5. 测试配套：tests/test_tb2_sandbox_recipe.py 新增 test_task_env_resources_floors，用地板下 / 正常两档值验证 floor 只在 recipe 层应用一次；无配置、schema 或部署文件改动，离线测试 61 通过 2 跳过。

关键文件：

- examples/experimental/openenv/tb2_sandbox_recipe.py（模块 构建配方；类别 source；类型 core-logic；符号 task_env_resources, run_with_deadline, server_cmd, COMMAND_TIMEOUT_S）：共享 recipe 层，新增 task_env_resources、run_with_deadline 与 COMMAND_TIMEOUT_S，是本次去重的唯一事实来源
- examples/experimental/openenv/tb2_sandbox_daytona.py（模块 沙箱后端；类别 source；类型 core-logic；符号 task_resources, create_task_sandbox）：Daytona 物化改用共享 floor 并做 GB 映射，清理路径加保护；修复了与其他 provider 的资源解释漂移
- examples/experimental/openenv/tb2_sandbox_e2b.py（模块 沙箱后端；类别 source；类型 dependency-wiring；符号 task_build_resources, ensure_task_template, create_task_sandbox）：E2B 物化复用共享 helper，删除 ttl_s 参数并新增 ready timeout 环境变量，是 API 行为变化最集中处
- examples/experimental/openenv/tests/test_tb2_sandbox_recipe.py（模块 构建配方；类别 test；类型 test-coverage；符号 test_task_env_resources_floors）：新增 test_task_env_resources_floors，把 floor 策略锁定在 recipe 层，防止未来 provider 再次漂移

关键符号：task_env_resources, run_with_deadline, server_cmd, task_resources, task_build_resources, ensure_task_template, create_task_sandbox, test_task_env_resources_floors

关键源码片段

examples/experimental/openenv/tb2_sandbox_recipe.py

共享 recipe 层，新增 task_env_resources、run_with_deadline 与 COMMAND_TIMEOUT_S，是本次去重的唯一事实来源

```
# tb2_sandbox_recipe.py —— 共享层
# 沙箱内单条命令的执行超时：以 TB2_COMMAND_TIMEOUT_S 为唯一来源
#  （这是 tbench2_env 的契约变量，命名对齐服务器侧），默认 900 秒。
# 归属 recipe 而非各 provider：只有这里构建 server 命令行，
# 所有后端自然拿到同一值，不再维护三个可能漂移的 getenv 副本。
```

```
COMMAND_TIMEOUT_S = int(os.getenv("TB2_COMMAND_TIMEOUT_S", "900"))


def task_env_resources(task_dir: Path) -> tuple[int, int, int]:
    """从 task.toml 的 [environment] 读取 CPU/内存/磁盘并应用地板值。

    地板（1 CPU / 2048 MB 内存 / 10240 MB 磁盘）是 recipe 策略——
    环境服务器需要与任务并行运行的余量——只在这里应用一次；
    各 provider 自行映射单位、丢弃自己不用的字段，
    避免同一份 task.toml 在不同 provider 上被不同对待。
    """
    env_cfg = read_task_config(task_dir).get("environment", {})
    return (
        max(1, int(env_cfg.get("cpus", 1))),
        max(2048, int(env_cfg.get("memory_mb", 2048))),
        max(10240, int(env_cfg.get("storage_mb", 10240))),
    )


def run_with_deadline(fn, timeout_s: float):
    """在单 worker 线程上执行 *fn*，超过 *timeout_s* 抛 TimeoutError。

    用于没有自身超时、跨多次请求阻塞的 provider 调用（如 E2B 模板构建）。
    刻意不用 with ThreadPoolExecutor 块：其 __exit__ 会 join worker，
    等于等完超时本要放弃的那个调用。超时后 provider 侧工作仍可能继续，
    回收它产出的东西是调用方的 provider 特有事务；
    但本调用方不再持有锁和信号量槽位。
    """
    pool = concurrent.futures.ThreadPoolExecutor(max_workers=1)
    try:
        return pool.submit(fn).result(timeout=timeout_s)
    finally:
        pool.shutdown(wait=False) # 不 join 仍在运行的 provider 调用
```

examples/experimental/openenv/tb2_sandbox_daytona.py

Daytona 物化改用共享 floor 并做 GB 映射，清理路径加保护；修复了与其他 provider 的资源解释漂移

```
# tb2_sandbox_daytona.py —— provider 物化
def task_resources(task_dir: Path):
    from daytona import Resources

    # Daytona 按整 GB 计资源；recipe 的地板（2048 MB / 10240 MB）
    # 保证整数除法不会把最小值舍成 0。
    cpus, memory_mb, storage_mb = task_env_resources(task_dir)
    return Resources(cpu=cpus, memory=memory_mb // 1024, disk=storage_mb // 1024)


def create_task_sandbox(
```

```
    daytona,
    task_dir: Path,
    *,
    command_timeout_s: int = COMMAND_TIMEOUT_S, # 默认值统一来自 recipe
    create_timeout_s: float = 1800.0,
    ready_timeout_s: float = _READY_TIMEOUT_S, # OPENENV_DAYTONA_READY_TIMEOUT_S
    auto_stop_minutes: int = 30, # 刻意保持参数而非环境变量：
    auto_delete_minutes: int = 120, # 它们是 Daytona 自有区间，与心跳节奏耦合
):
    from daytona import CreateSandboxFromImageParams

    params = CreateSandboxFromImageParams(
        image=build_task_image(task_dir),
        resources=task_resources(task_dir),
        auto_stop_interval=auto_stop_minutes,
        auto_delete_interval=auto_delete_minutes,
        labels=sandbox_labels(task_dir),
    )
    sandbox = daytona.create(params, timeout=create_timeout_s)
    try:
        cmd = server_cmd(command_timeout_s, default_task_id=task_dir.name)
        # Daytona 不执行镜像 CMD，需显式 nohup 启动环境服务器并记录 PID
        sandbox.process.exec(
            f"nohup bash -c {shlex.quote(cmd)} > /tmp/openenv-server.log 2>&1 &"
            " echo $! > /tmp/openenv-server.pid",
            timeout=10,
        )
        url = sandbox.create_signed_preview_url(8000, expires_in_seconds=86400).url
        wait_server_ready(url, timeout_s=ready_timeout_s)
        _start_keepalive(sandbox, task_dir.name)
        return sandbox, url
    except Exception:
        # 清理失败不得掩盖 create 失败的真实异常；
        # 即便 delete 也失败，auto-stop/auto-delete TTL 仍会兜底回收。
        try:
            daytona.delete(sandbox)
        except Exception:
            pass
        raise
```

评论区精华

Review 活动极少：Shi-Dong 直接 APPROVED，评论仅一句 “LGTM!”，无实质交锋。本 PR 最有价值的“讨论”其实在 PR body 里：run_with_deadline 为何不能写成 with ThreadPoolExecutor 块——__exit__ 会 join worker，等完超时本要放弃的调用；以及“描述一个沙箱的 knob 应放在 create 旁边，fan-out knob 留在后端”的配置归属原则。

- 重构整体审批 (other): 直接合并

风险与影响

- 风险：
  1. API 破坏（影响面小）：tb2_sandbox_e2b.create_task_sandbox 删除了 ttl_s 参数，显式传参的调用方会直接 TypeError；该目录为 experimental，影响有限，但迁移时需确认调用方。
  2. 环境变量影响面扩大（有意的）：设置 TB2_COMMAND_TIMEOUT_S 后统一影响所有 provider 的 server 命令超时；不设置时默认值 900 / 300 与原来一致，行为兼容。COMMAND_TIMEOUT_S 在 import 时读取，依赖"rollout worker 每次是新进程"的进程模型。
  3. 行为修复：Daytona 的 GB 地板与其他 provider 的 MB 地板已出现语义漂移，统一后同一 task.toml 在各 provider 上遵守一致，这是本 PR 的核心收益。
  4. 测试缺口：新测试只覆盖 recipe 层 floor，provider 层的 GB 整数除法映射没有新增单测，主要靠既有离线测试兜底。 - 影响：影响范围限定在 examples/experimental/openenv 下的 TB2 沙箱后端（Daytona / E2B / 共享 recipe），不触及训练主路径、ppo、rollout 等核心模块。对用户而言，显式传 ttl_s 的 E2B 调用方需要迁移；对系统而言，统一策略降低了多 provider 维护成本，为 #2276 Modal 后端复用 helper 铺路。对团队而言，本 PR 确立了两个可复用的设计原则：provider 无关策略只放 recipe 一处、清理失败不得掩盖主异常。 - 风险标记：API 参数删除，环境变量影响面扩大，provider 映射缺测试

关联脉络

- PR #2275 (2/3 of the stack): PR body 明确声明为本 PR 所在 3 步栈的第 2 步，承接本次共享 helper 上收
- PR #2276 (3/3 of the stack: Modal sandbox backend): 栈的最后一环，新增 Modal 沙箱后端并复用本 PR 上收的 helper；对 provider 的实机验证在 3/3 完成
- PR #2237 fix(openenv): build E2B task templates as root: 同改 tb2_sandbox_e2b.py 的历史修复，展示 openenv 沙箱后端在同一功能线的持续演进