

PR #1872 完整报告

radixark/miles

Add PortAllocator.alloc and thread the shared allocator through allocation

合并时间: 2026-09-04 13:06

原文链接: <http://prhub.com.cn/radixark/miles/pull/1872>

执行摘要

- 一句话: 重构 PortAllocator, 引入 alloc 方法, 简化端口分配与状态管理。
- 推荐动作: 建议精读此 PR。这是一个高质量的重构, 展示了如何通过封装状态操作来简化复杂的分配逻辑。值得学习的设计点包括: 1) 将分散的状态管理 (node_port_cursor 和返回新 PortAllocator) 内聚到一个明确的 alloc 方法中; 2) 使用 functools.partial 清晰地创建特定于上下文 (engine 和 node_ip) 的端口分配回调; 3) 配套的测试策略从测试辅助方法 (assign/next_base_port) 转向测试核心行为 (alloc 的各种场景)。

此 PR 是 Issue #1837 这一宏大重构目标中的一个坚实步骤, 其编码质量和测试覆盖度值得借鉴。

功能与动机

作为 Issue #1837 跟踪的重构计划的一部分 (Part of #1837), 旨在简化端口分配器的接口和内部状态管理。原始的分配函数 `allocate_rollout_engine_addr_and_ports_normal` 内部维护一个私有的节点游标字典 `node_port_cursor`, 并在结束时返回一个新的 `PortAllocator` 实例, 调用方 (`ServerGroup.start_engines`) 需要将新实例赋值回去, 增加了状态传递的复杂性。新的 `alloc` 方法将分配和状态更新耦合在一起, 使得共享分配器的管理更加直接和清晰。

实现拆解

1. 核心变更: `PortAllocator.alloc` 方法: 在 `miles/ray/rollout/addr_allocator.py` 中, 为 `PortAllocator` 类新增 `alloc` 方法。该方法封装了通过 Ray actor 获取并验证空闲端口、更新内部游标 `_values` 的全部逻辑。* * 涉及文件: `miles/ray/rollout/addr_allocator.py`。关键符号: `alloc`。* * 具体变更: 移除了旧的 `assign` 和 `next_base_port` 方法, 新增 `alloc(self, *, engine, node_ip: str, consecutive: int = 1) -> int`。方法内部直接调用 `engine._get_current_node_ip_and_free_port.remote` 进行端口探测, 并将结果写入 `self._values[node_ip]`。* * 原因与影响: 将状态变更 (游标推进) 与资源分配 (端口获取) 绑定, 使 `PortAllocator` 成为一个自包含的、可变状态的对象。这移除了 `allocate_rollout_engine_addr_and_ports_normal` 需要返回新分配器的模式, 简化了调用方的逻辑。
2. 更新分配函数接口: 修改 `allocate_rollout_engine_addr_and_ports_normal` 的签名, 移除 `base_port` 参数, 新增必选参数 `port_allocator: PortAllocator`。* * 涉及文件: `miles/ray/rollout/addr_allocator.py`。关键符号: `allocate_rollout_engine_addr_and_ports_normal`。* * 具体变更: 函数内部不再创建

node_port_cursor 字典和 get_addr_and_ports 嵌套函数。改为使用 functools.partial 创建 get_port = functools.partial(port_allocator.alloc, engine=engine, node_ip=node_ip)，直接通过传入的共享分配器分配端口。函数返回值从 (addr_and_ports, PortAllocator) 简化为仅返回 addr_and_ports。 * * 原因与影响 *：使分配函数成为纯过程式调用（除了对传入分配器的原地修改），状态管理职责完全移交给 PortAllocator。

3. 更新调用方：适配新的分配函数接口。 * * 涉及文件：miles/ray/rollout/server_group.py。关键符号：start_engines 方法中的调用。 * * 具体变更：在 ServerGroup.start_engines 中，调用 allocate_rollout_engine_addr_and_ports_normal 时，不再计算 base_port 并传递，而是直接将 port_allocator 参数传递给函数。同时移除了调用后的 port_allocator.assign(next_port_allocator) 赋值操作。 * * 原因与影响 *：与核心变更配套，简化了 ServerGroup 中引擎启动的端口分配流程。

4. 全面更新测试：重构测试用例以验证新的 alloc 方法，并适配所有调用了分配函数的测试夹具。 * * 涉及文件：tests/fast/ray/rollout/test_addr_allocator.py, tests/fast/ray/rollout/real_ray/test_inference_controller.py, tests/fast/ray/rollout/real_ray/conftest.py。 * * 具体变更：

- test_addr_allocator.py: 删除了测试 next_base_port 和 assign 的用例，新增了 test_alloc_advances_the_cursor_of_its_node, test_alloc_starts_from_the_base_port_on_an_unseen_node, test_alloc_consecutive_reserves_a_whole_block, test_alloc_tracks_nodes_independently 四个测试，专门验证 alloc 方法的行为。更新了 TestAllocateNormal 类中所有测试用例，以适配新的函数签名（传入 port_allocator 参数）和返回值。
- test_inference_controller.py 和 conftest.py: 更新了其中的 mock 分配函数 _fake_alloc，使其返回值从元组 (dict, PortAllocator) 改为仅返回 dict，以匹配新的函数接口。
- 原因与影响 *：确保重构后的代码在单元和集成层面行为正确，维持了高测试覆盖率。

关键文件：

- miles/ray/rollout/addr_allocator.py（模块 端口分配器；类别 source；类型 core-logic；符号 alloc, allocate_rollout_engine_addr_and_ports_normal）：核心变更文件，引入了 alloc 方法并重构了 allocate_rollout_engine_addr_and_ports_normal 函数的接口与内部逻辑。
- miles/ray/rollout/server_group.py（模块 服务器组；类别 source；类型 core-logic）：核心调用方，start_engines 方法适配了新的分配函数接口，移除了冗余的状态同步代码。
- tests/fast/ray/rollout/test_addr_allocator.py（模块 端口分配测试；类别 test；类型 test-coverage；符号 test_alloc_advances_the_cursor_of_its_node, test_alloc_starts_from_the_base_port_on_an_unseen_node, test_alloc_consecutive_reserves_a_whole_block, test_alloc_tracks_nodes_independently）：测试主体文件，全面重写了 TestPortAllocator 类以覆盖 alloc 方法，并更新了 TestAllocateNormal 类中的所有用例。
- tests/fast/ray/rollout/real_ray/test_inference_controller.py（模块 推理控制器测试；类别 test；类型 test-coverage）：集成测试文件，更新了 mock 分配函数以适配新的返回值类

型。

- tests/fast/ray/rollout/real_ray/conftest.py (模块 测试夹具; 类别 test; 类型 test-coverage) : 测试夹具文件, 更新了 patched_sglang_engine fixture 中的 mock 分配函数。

关键符号: PortAllocator.alloc, allocate_rollout_engine_addr_and_ports_normal, ServerGroup.start_engines

关键源码片段

miles/ray/rollout/addr_allocator.py

核心变更文件, 引入了 alloc 方法并重构了 allocate_rollout_engine_addr_and_ports_normal 函数的接口与内部逻辑。

```
# miles/ray/rollout/addr_allocator.py 中 PortAllocator 类的变更
# 变更前: 通过 assign 复制状态, 通过 next_base_port 获取全局起始点。
# 变更后: alloc 方法封装了通过 Ray 探测端口、更新自身游标的全部逻辑。
# 这使得 PortAllocator 成为一个可变状态的对象, 分配与状态更新原子化。
@dataclass
class PortAllocator:
    _values: dict[str, int]

    @staticmethod
    def empty() -> "PortAllocator":
        return PortAllocator(_values={})

    # 新增的 alloc 方法: 分配端口并原地更新游标
    def alloc(self, *, engine, node_ip: str, consecutive: int = 1) -> int:
        # 使用小端口避免与临时端口 (32768-65535) 冲突
        # Ray 使用 10002-19999, 因此起始点设为 15000 以避免竞争
        start_port = self._values.get(node_ip, 15000)
        # 通过 Ray RPC 获取空闲端口
        _, port = ray.get(
            engine._get_current_node_ip_and_free_port.remote(
                start_port=start_port,
                consecutive=consecutive,
            )
        )
        # 更新该节点的游标, 为下一次分配做好准备
        self._values[node_ip] = port + consecutive
        return port

    # 旧的 assign 和 next_base_port 方法已被移除
```

tests/fast/ray/rollout/test_addr_allocator.py

测试主体文件, 全面重写了 TestPortAllocator 类以覆盖 alloc 方法, 并更新了 TestAllocateNormal 类中的所有用例。

```
# tests/fast/ray/rollout/test_addr_allocator.py 中测试的变更示例
```

```

# 测试重点从验证辅助方法（assign, next_base_port）转向验证核心分配行为（alloc）
class TestPortAllocator:
    # 新增测试：验证 alloc 能正确推进节点的游标
    def test_alloc_advances_the_cursor_of_its_node(self, patch_ray_get):
        """同一节点上的两次分配必须返回不重叠的端口。"""
        cursors = PortAllocator.empty()
        engine = fake_engine(host="10.0.0.1", port_seed=0)
        first = cursors.alloc(engine=engine, node_ip="10.0.0.1")
        second = cursors.alloc(engine=engine, node_ip="10.0.0.1")
        assert second > first
        assert cursors._values["10.0.0.1"] == second + 1

    # 新增测试：验证对新节点从默认基础端口开始分配
    def test_alloc_starts_from_the_base_port_on_an_unseen_node(self, patch_ray_get):
        """没有游标的节点从基础端口（远离 Ray 端口范围）开始分配。"""
        cursors = PortAllocator.empty()
        engine = fake_engine(host="10.0.0.1", port_seed=0)
        assert cursors.alloc(engine=engine, node_ip="10.0.0.1") == 15000

    # 新增测试：验证分配连续端口块
    def test_alloc_consecutive_reserves_a_whole_block(self, patch_ray_get):
        """一次 consecutive=N 的分配必须将节点的游标移动到整个块之后。"""
        cursors = PortAllocator.empty()
        engine = fake_engine(host="10.0.0.1", port_seed=0)
        first = cursors.alloc(engine=engine, node_ip="10.0.0.1", consecutive=5)
        assert cursors._values["10.0.0.1"] == first + 5

    # 新增测试：验证多节点独立跟踪
    def test_alloc_tracks_nodes_independently(self, patch_ray_get):
        """每个节点 IP 拥有自己的游标。"""
        cursors = PortAllocator.empty()
        engine_a = fake_engine(host="10.0.0.1", port_seed=0)
        engine_b = fake_engine(host="10.0.0.2", port_seed=0)
        cursors.alloc(engine=engine_a, node_ip="10.0.0.1")
        cursors.alloc(engine=engine_b, node_ip="10.0.0.2")
        assert set(cursors._values.keys()) == {"10.0.0.1", "10.0.0.2"}

```

评论区精华

本次 PR 的 review 讨论较少，仅有一条 approve 评论。

guapisolo: "Reviewed in stack context. The shared per-node allocator change is sound, with no active findings requiring follow-up before safe merge checkpoint #2176. LGTM."

- 决策结论：审核者认为共享的、基于节点的分配器变更合理，没有需要后续跟进的发现，支持合并。这表明此次重构的架构方向得到了团队的认可。
- 未解决疑虑：评论中未提出任何技术疑虑或争议点。结合 commit message 的清晰描述和全面的测试覆盖，此次重构在技术层面被认为是清晰且安全的。

- 共享端口分配器设计的合理性 (design): 重构的设计方向得到认可, 认为变更合理。

风险与影响

• 风险:

1. 回归风险: 中等。变更了 PortAllocator 的核心 API (alloc vs assign/next_base_port) 和 allocate_rollout_engine_addr_and_ports_normal 的签名与返回值, 属于对现有推理引擎端口分配逻辑的底层重构。尽管测试覆盖全面, 但涉及多节点端口协调的复杂场景, 若测试 mock 不充分或存在边界条件遗漏, 可能在多节点部署中引发端口冲突。* * 具体文件 / 逻辑 *: miles/ray/rollout/addr_allocator.py 中的 alloc 方法和 allocate_rollout_engine_addr_and_ports_normal 函数; miles/ray/rollout/server_group.py 中的调用点。
2. 兼容性风险: 低。所有直接调用 allocate_rollout_engine_addr_and_ports_normal 的地方 (包括源码和测试) 都已同步更新。这是一个内部重构, 不涉及对外部 API 或配置的修改。
3. 性能风险: 极低。重构主要是代码组织层面的优化, 没有改变端口分配的核心算法 (依然是逐个通过 Ray RPC 探测空闲端口)。
4. 设计风险: 低。将 alloc 作为方法放在 PortAllocator 上, 使其成为状态管理的单一职责点, 相比旧方案 (函数内部创建临时状态并返回新对象) 更符合内聚原则。review 确认了设计的合理性。

• 影响:

1. 对用户 / 系统的影响: 直接影响 Miles 推理引擎的启动流程, 具体是 ServerGroup 在分配服务器地址和端口时的行为。由于是内部重构, 对最终用户 (使用 Miles 进行训练和推理) 无感知, 但影响底层组件的健壮性和可维护性。成功的重构可以减少端口管理相关的潜在 bug, 并为后续可能的优化 (如 PR #2501 中进一步调整起始端口) 打下基础。
2. 对团队的影响: 简化了 PortAllocator 和端口分配流程的代码, 降低了理解和维护的复杂度。新的 alloc 接口比旧的 assign/next_base_port 组合更直观, 新成员上手更容易。同时, 清晰的测试用例为相关逻辑提供了良好的文档。
3. 影响范围: 集中在 rollout 模块的端口分配子系统, 涉及核心启动路径。变更文件数不多 (5 个), 但涉及核心逻辑、调用方和多个测试文件, 属于集中且深入的重构。- 风险标记: 核心路径变更, 接口契约变更, 多节点端口协调

关联脉络

- PR #2048 Rename PortAllocator internals and pass the actor positionally: 同一系列重构中的后续 PR, 进一步重命名 PortAllocator 内部状态 (_values 可能改为更明确的名字) 并调整 alloc 方法的参数传递方式 (从关键字参数改为位置参数), 是本 PR 引入 alloc 后的顺滑演进。
- PR #2501 Start dynamic worker ports above Ray's worker range: 后续 PR, 调整了动态端口分配的起始值 (从 15000 改为 20000), 这是在本 PR 重构后的 PortAllocator.alloc 方法中 (硬编码的 15000) 进行的配置调整, 进一步提升了与 Ray 内部端口范围的隔离性。

- PR #1987 Move PortAllocator into miles.utils.workers: 后续 PR, 将 PortAllocator 从 miles.ray.rollout.addr_allocator 移动到更通用的 miles.utils.workers 模块, 是本 PR 完成核心逻辑重构后, 在代码组织层面的进一步归整。