

PR #1871 完整报告

radixark/miles

Rename PortCursors to PortAllocator and key it by the node's ip

合并时间: 2026-09-04 13:05

原文链接: <http://prhub.com.cn/radixark/miles/pull/1871>

执行摘要

- 一句话: 将端口分配器内部状态键从节点索引改为节点 IP, 提升身份标识的准确性。
- 推荐动作: 这是一个清晰、内聚的重构 PR, 值得精读。其核心决策——将端口分配状态的键从代理的节点索引改为真实的节点 IP——体现了对系统健壮性的主动改进。虽然逻辑简单, 但它是整个 Rollout 模块规范化、健壮化进程中的必要步骤, 可以作为代码重构中“改善内部表示”的范例学习。

功能与动机

PR 标题明确指出变更动机是“key it by the node's ip”。commit message 进一步解释: 原键 `node_index` 是一个基于 rank 布局的算术代理值, 而节点 IP 是机器的真实身份, 不依赖这种假设。虽然当前行为正确, 但使用节点索引作为键是脆弱且粗放的, 改为 IP 键提升了实现的精确性和可维护性。

实现拆解

1. 核心逻辑重命名与键类型调整: 在 `miles/ray/rollout/addr_allocator.py` 中, 将数据类 `PortCursors` 重命名为 `PortAllocator`, 其内部 `_values` 字典的类型从 `dict[int, int]` 改为 `dict[str, int]`。函数 `allocate_rollout_engine_addr_and_ports_normal` 中的局部变量 `node_port_cursor` 同样调整为 `dict[str, int]`。
2. 节点身份获取方式变更: 在 `allocate_rollout_engine_addr_and_ports_normal` 函数中, 新增一行 `node_ip, _ = ray.get(engine._get_current_node_ip_and_free_port.remote())` 以显式获取节点 IP。内部函数 `get_addr_and_ports` 的参数名从 `node_idx` 改为 `node_ip`, 并在其内部使用该 IP 作为 `node_port_cursor` 字典的键进行存取。函数最终返回更新后的 `PortAllocator(_values=node_port_cursor)`。
3. 依赖方同步更新: 在 `miles/ray/rollout/server_group.py` 的 `ServerGroup.start_engines` 和 `recover` 方法签名中, 将参数 `port_cursors` 重命名为 `port_allocator`, 并更新其内部调用。在 `miles/ray/rollout/rollout_server.py` 和 `miles/ray/rollout/inference_controller.py` 中, 所有实例化 `PortCursors.empty()` 的地方统一替换为 `PortAllocator.empty()`, 并更新相关变量名。
4. 测试配套更新: `tests/fast/ray/rollout/test_addr_allocator.py` 中, 测试类 `TestPortCursors` 和 `TestSharedPortCursorsAcrossGroups` 分别重命名为 `TestPortAllocator` 和 `TestSharedPortAllocatorAcrossGroups`, 其测试用例中涉及 `_values` 字典的断言从使用整数索引改为使用 IP 字符串。`tests/fast/ray/rollout/real_ray/`

下多个测试文件（如 `test_server_group.py`, `test_fault_tolerance.py`）同步更新了导入、实例化调用和 `mock` 对象中的键类型。

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 `PortAllocator`, `allocate_rollout_engine_addr_and_ports_normal`）：核心变更文件。包含 `PortAllocator` 类定义和 `allocate_rollout_engine_addr_and_ports_normal` 函数中节点身份获取与键值调整的核心逻辑。
- `tests/fast/ray/rollout/test_addr_allocator.py`（模块 分配器测试；类别 `test`；类型 `test-coverage`；符号 `TestPortAllocator`, `TestSharedPortAllocatorAcrossGroups`）：核心测试文件。测试类和测试数据同步反映了 `PortAllocator` 的键类型变更，是验证重构正确性的关键。
- `miles/ray/rollout/server_group.py`（模块 服务器组；类别 `source`；类型 `core-logic`；符号 `start_engines`, `recover`）：主要依赖方。其 `start_engines` 和 `recover` 方法的参数和内部变量名同步更新，是连接分配器与服务器组的关键接缝。
- `miles/ray/rollout/rollout_server.py`（模块 服务器启动；类别 `source`；类型 `dependency-wiring`）：入口点文件。在启动服务器和恢复引擎的流程中实例化和传递 `PortAllocator`。

关键符号：`PortAllocator.empty`, `PortAllocator.assign`, `PortAllocator.next_base_port`, `allocate_rollout_engine_addr_and_ports_normal`, `ServerGroup.start_engines`, `ServerGroup.recover`, `RolloutServer.recover`, `InferenceController.start_cell`

关键源码片段

`miles/ray/rollout/addr_allocator.py`

核心变更文件。包含 `PortAllocator` 类定义和 `allocate_rollout_engine_addr_and_ports_normal` 函数中节点身份获取与键值调整的核心逻辑。

```
from dataclasses import dataclass

@dataclass
class PortAllocator:
    # 内部状态：从节点索引 (int) 改为节点 IP (str)，以更准确地标识机器身份。
    _values: dict[str, int]

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

    def assign(self, other: "PortAllocator"):
        self._values = other._values.copy()

    def next_base_port(self) -> int:
        # 返回所有已知节点中已分配的最大端口值加一，作为新的起始端口。
        return max(self._values.values()) if self._values else 15000
```

```

# 在 allocate_rollout_engine_addr_and_ports_normal 函数内部:
# 1. 字典类型调整为 dict[str, int]
node_port_cursor: dict[str, int] = {}

# 2. 在遍历引擎时, 显式获取节点 IP 作为后续操作的键
node_ip, _ = ray.get(engine._get_current_node_ip_and_free_port.remote())
get_addr, get_port = get_addr_and_ports(engine, node_ip) # 参数名也相应改为 node_ip

# 3. 内部闭包使用传入的 node_ip 作为键
start_port = node_port_cursor.get(node_ip, base_port)
# ... 在 port() 闭包中更新 ...
node_port_cursor[node_ip] = start_port

# 4. 返回时使用新的 PortAllocator
return addr_and_ports, PortAllocator(_values=node_port_cursor)

```

tests/fast/ray/rollout/test_addr_allocator.py

核心测试文件。测试类和测试数据同步反映了 **PortAllocator** 的键类型变更, 是验证重构正确性的关键。

```

# 测试类重命名并更新断言
class TestPortAllocator: # 原 TestPortCursors
    def test_next_base_port_returns_max_value(self):
        # 测试数据使用 IP 字符串作为键
        c = PortAllocator(_values={"10.0.0.1": 17000, "10.0.0.2": 16500, "10.0.0.3": 18000})
        assert c.next_base_port() == 18000

# 测试断言中的键类型变更
class TestAllocateNormal:
    def test_single_node_8_cards_tp1(self, patch_ray_get):
        # ... 前面逻辑 ...
        # 断言: 键集合现在是 IP 字符串, 而非节点索引
        assert isinstance(cursors, PortAllocator)
        assert set(cursors._values.keys()) == {"10.0.0.1"} # 原为 {0}
        assert cursors._values["10.0.0.1"] >= max(_all_ports(addr_and_ports)) + 1 # 原为 cursors._values[0]

class TestSharedPortAllocatorAcrossGroups: # 原 TestSharedPortCursorsAcrossGroups
    def test_sequential_groups_share_cursor_and_avoid_overlap(self, patch_ray_get):
        cursors = PortAllocator.empty() # 原 PortCursors.empty()
        # ... 测试逻辑 ...

```

miles/ray/rollout/server_group.py

主要依赖方。其 **start_engines** 和 **recover** 方法的参数和内部变量名同步更新, 是连接分配器与服务器组的关键接缝。

```

from miles.ray.rollout.addr_allocator import (
    PortAllocator, # 从 PortCursors 重命名导入
    allocate_rollout_engine_addr_and_ports_external,

```

```

    allocate_rollout_engine_addr_and_ports_normal,
)

class ServerGroup:
    def start_engines(
        self, port_allocator: PortAllocator, start_cell_indices: list[int] | None = None #
        参数名和类型更新
    ) -> tuple[list, list[int]]:
        """创建 Ray actors, 分配端口, 并启动 ``engine.init()``。
        就地修改 ``port_allocator`` 以推进到任何新分配端口之后。"""
        # ... 内部逻辑 ...
        base_port = port_allocator.next_base_port() # 变量名更新
        addr_and_ports, next_port_allocator = allocate_rollout_engine_addr_and_ports_normal(
            # ... 其他参数 ...
            base_port=base_port,
        )
        port_allocator.assign(next_port_allocator) # 方法调用和变量名更新

    async def recover(self, port_allocator: PortAllocator, filter_cell_indices: list[int] | None =
    None):
        # ... 内部逻辑 ...
        handles, new_engine_indices = self.start_engines(port_allocator, start_cell_indices=filter_
        cell_indices) # 变量名更新

```

评论区精华

审核者 guapisolo 在批准评论中确认: “This cleanly migrates port-allocation identity from group-local indices to node IPs; #1872 completes the shared allocator mechanics before the #2176 safe-merge checkpoint, and the behavior remains present through the checked chain tip.” 这表明该变更是更大重构序列（#1837 栈）中的一环，旨在为后续更复杂的共享分配器机制（#1872）做准备，并确保行为在当前链路中保持一致。无其他评论或争议。

- PR #1871 作为 #1837 重构序列的一环被批准 (design): 该变更被认可为更大重构方案（#1837 栈）中的一个干净、正确的步骤，旨在改善端口分配的身份标识，并为后续 PR（#1872）完善共享分配器机制做准备。行为在当前链路中保持一致。

风险与影响

- 风险:
 1. 逻辑正确性风险: 虽然 `get_addr_and_ports` 函数原本也通过 `engine._get_current_node_ip_and_free_port.remote()` 获取地址，但新逻辑在函数外显式获取了一次 IP，并将其作为键。需确保两次获取的 IP 一致性，但考虑到这是同一个 Actor 的同一次调用，理论上应返回相同 IP。
 2. 依赖方遗漏风险: 本次变更涉及 11 个文件，主要是测试文件。所有 PortCursors 的使用点均已更新为 PortAllocator，且测试数据中的键也从 int 改为 str。风险较低，但需通过 CI 验证无语法或导入错误。

3. 端口分配行为不变：核心分配算法（如 `next_base_port` 取最大值逻辑）未变，仅键的标识含义改变。因此，只要所有使用方统一，端口分配结果应与之前完全一致，回归风险低。
- 影响：此变更属于内部重构，对最终用户和系统外部行为无直接影响。它提升了代码的内部质量和可维护性，使端口分配器的身份标识更准确，为未来支持更灵活的节点拓扑或并行恢复奠定了更坚实的基础。影响范围限于 Miles 的 Ray Rollout 模块内部，特别是端口分配和服务组管理逻辑。
 - 风险标记：键类型变更，依赖方更新

关联脉络

- PR #2048 Rename PortAllocator internals and pass the actor positionally: 同仓库近期 PR，同样在重构 PortAllocator 模块（原 `addr_allocator.py`）。它进一步调整了 `alloc` 方法签名，是同一代码质量提升脉络的延续。
- PR #2050 Stop checking actor liveness while waiting for a tcp port: 同仓库近期 PR，同样涉及 `miles/utils/http_utils.py` 和 `miles/ray/rollout/router_manager.py`，简化了端口就绪探测接口。与本 PR 共同服务于 Rollout 模块端口管理逻辑的清晰化与健壮化。
- PR #2049 Add a worker provider abstraction over RayWorkerManager: 同仓库近期 PR，引入了工作者提供者抽象层，封装了包括地址查询在内的底层逻辑。与本 PR 的端口分配器重构共同体现了模块化、抽象化的设计方向。
- PR #1987 Move PortAllocator into `miles.utils.workers`: 更早期的 PR，将 PortAllocator 模块从 `miles/ray/rollout/` 移动到了更通用的 `miles/utils/workers/` 目录下，为其成为通用工具模块做了组织上的准备。本 PR 的重命名是在该移动之后进行的语义优化。