PR #7386 完整报告

verl-project/verl

[ray, doc] fix: make Slurm Ray network interface configurable

合并时间：2026-08-13 17:40

原文链接：http://prhub.com.cn/verl-project/verl/pull/7386

执行摘要

- 一句话：Slurm 示例新增 RAY_NETWORK_INTERFACE 配置，支持绑定指定网卡
- 推荐动作：值得快速浏览，特别是需要在 Slurm 多网卡集群上部署 verl 的团队。本 PR 的设计体现了「配置化优先、默认行为不变」的原则，通过 srun 动态解析每节点 IP 并显式绑定，确保了正确性和可诊断性。对于示例代码审查者，可关注 srun --overlap 的兼容性和错误处理路径。

功能与动机

PR 作者指出：On multi-NIC clusters, hostname resolution may select a management network instead of the high-speed interconnect. 在多网卡集群上，hostname --ip-address 可能解析到管理网卡地址，导致 Ray actor 注册到慢速网络，影响训练性能。本 PR 通过 RAY_NETWORK_INTERFACE 变量让用户显式选择网卡，并独立解析每个节点上该接口的 IPv4 地址，确保 Ray 绑定到高速互联网络。

实现拆解

1. 在 examples/tutorial/slurm/ray_on_slurm.slurm 开头新增环境变量声明 RAY_NETWORK_INTERFACE=${RAY_NETWORK_INTERFACE:-}，默认空值不影响原有 hostname --ip-address 解析路径。
2. 定义函数 get_node_ipv4()：接收节点名，通过 srun --overlap --nodes=1 --ntasks=1 -w 在该节点执行 ip -4 -o addr show dev 解析指定网卡的 IPv4 地址，用 awk 提取 CIDR 前缀部分；若为空则打印错误并返回 1，避免静默使用错误网卡。
3. 修改 head 节点 IP 获取逻辑：若设置了 RAY_NETWORK_INTERFACE，则调用 get_node_ipv4 并将结果用于后续 ray start --node-ip-address，否则保留原 hostname --ip-address 路径。
4. 修改 worker 节点启动循环：为每个 worker 按同样方式解析 IP，并通过数组参数 worker_node_ip_args 将 --node-ip-address 传给 ray start；未设置变量时保持原输出和启动命令。
5. 更新 docs/start/multinode.rst，增加多网卡集群使用说明、RAY_NETWORK_INTERFACE 用法示例，并强调 Gloo 和 NCCL 的接口设置需要单独配置。测试方面：作者使用 bash -n、git diff --check、pre-commit 校验，并用 mocked srun 输出测试成功与失败路径，最终在四节点集群上验证，设置 RAY_NETWORK_INTERFACE=ib

0 后 Ray 节点注册到 10.245.* 高带宽网段，训练任务成功完成。

关键文件：

- examples/tutorial/slurm/ray_on_slurm.slurm（模块 示例脚本；类别 other；类型 core-logic；符号 get_node_ipv4）：核心变更文件，新增 RAY_NETWORK_INTERFACE 环境变量和 get_node_ipv4 函数，修改 head/worker 启动逻辑以支持显式绑定网卡 IP，是多网卡问题解决方案的具体实现。
- docs/start/multinode.rst（模块 文档；类别 docs；类型 documentation）：同步更新多节点文档，说明 RAY_NETWORK_INTERFACE 的用法，并明确 Ray 与 Gloo/NCCL 网卡设置相互独立，帮助用户在真实集群中正确配置。

关键符号：get_node_ipv4

关键源码片段

examples/tutorial/slurm/ray_on_slurm.slurm

核心变更文件，新增 RAY_NETWORK_INTERFACE 环境变量和 get_node_ipv4 函数，修改 head/worker 启动逻辑以支持显式绑定网卡 IP，是多网卡问题解决方案的具体实现。

```
# 可选：在多网卡集群上，用 RAY_NETWORK_INTERFACE 指定 Ray 使用的网卡
# 例如： RAY_NETWORK_INTERFACE=ib0 sbatch examples/tutorial/slurm/ray_on_slurm.slurm
RAY_NETWORK_INTERFACE=${RAY_NETWORK_INTERFACE:-}

# 在指定节点上解析所选网卡的 IPv4 地址，失败时打印错误并返回非零
# 使用 srun --overlap 独立在每个节点执行，避免依赖单一节点的解析结果
get_node_ipv4() {
    local node_name=$1
    local node_ip

    node_ip=$(srun --overlap --nodes=1 --ntasks=1 -w "$node_name" \
        ip -4 -o addr show dev "$RAY_NETWORK_INTERFACE" scope global |
        awk 'NR == 1 {split($4, address, "/"); print address[1]}')

    if [[ -z "$node_ip" ]]; then
        echo "Cannot find an IPv4 address for interface $RAY_NETWORK_INTERFACE on $node_
        name." >&2
        return 1
    fi

    printf '%s\n' "$node_ip"
}

# 获取 head 节点 IP：设置变量时走 get_node_ipv4，否则保持原有 hostname 解析
if [[ -n "$RAY_NETWORK_INTERFACE" ]]; then
    head_node_ip=$(get_node_ipv4 "$head_node") || exit 1
else
    head_node_ip=$(srun --nodes=1 --ntasks=1 -w "$head_node" hostname --ip-address)
fi
```

```
# worker 节点启动：同样在设置变量时为每个节点解析 IP 并传给 ray start
for ((i = 1; i <= worker_num; i++)); do
    node_i=${nodes_array[$i]}
    worker_node_ip_args=()
    if [[ -n "$RAY_NETWORK_INTERFACE" ]]; then
        node_i_ip=$(get_node_ipv4 "$node_i") || exit 1
        worker_node_ip_args=(--node-ip-address="$node_i_ip")
        echo "Starting WORKER $i at $node_i ($node_i_ip)"
    else
        echo "Starting WORKER $i at $node_i"
    fi
    srun --nodes=1 --ntasks=1 -w "$node_i" \
        apptainer run --nv --bind $verl_workdir $apptainer_image_path \
        ray start --address "$ip_head" "${worker_node_ip_args[@]}" \
        --num-cpus "${SLURM_CPUS_PER_TASK}" --num-gpus "${SLURM_GPUS_PER_NODE}" --
        block &
    sleep 5
done
```

评论区精华

本 PR 无实质性 review 评论。CLAassistant 检查通过，maintainer wuxibin89 直接批准（state: APPROVED）。作者在 PR 描述中详细说明了测试过程，包括模拟 srun 失败路径和真实四节点集群验证，属于低风险示例改动。

- 暂无高价值评论线程

风险与影响

- 风险：本 PR 仅修改示例脚本和文档，不影响 verl 核心库代码，风险低。但存在以下具体风险点：
  - get_node_ipv4 依赖 srun --overlap 选项，较老版本的 Slurm 可能不支持；且节点镜像需包含 ip 命令。
  - 若指定网卡在某个节点上无 IPv4 地址，脚本会提前退出，此时可能已有部分 worker 启动，造成残留进程；这是显式失败优于静默错误的取舍，但用户需保证网卡在所有节点上一致存在。
  - worker 启动循环中过早退出可能留下未清理的 head 或 worker 进程，属于示例脚本范畴，需用户自行注意。
  - 文档中新增的用法示例仅针对 Slurm 环境，其他集群管理器不受影响。
  - 影响：影响范围限于使用 examples/tutorial/slurm/ray_on_slurm.slurm 启动 verl 训练的用户，尤其是多网卡 Slurm 集群。默认行为完全不变，未设置 RAY_NETWORK_INTERFACE 时脚本与原版一致，因此对现有用户无破坏性影响。该改动为多网卡场景提供了显式配置手段，有助于提升 Ray 通信性能。对团队而言，该示例可作为后续部署最佳实践的参考。
  - 风险标记：示例脚本变更，依赖 srun 选项，多节点接口一致性，默认行为不变

关联脉络

- 暂无明显关联 PR