高性能计算集群

操作指南

腾讯云

文档目录

操作指南
管理高性能计算集群

最近更新时间：2023-05-31 16:45:31

操作场景

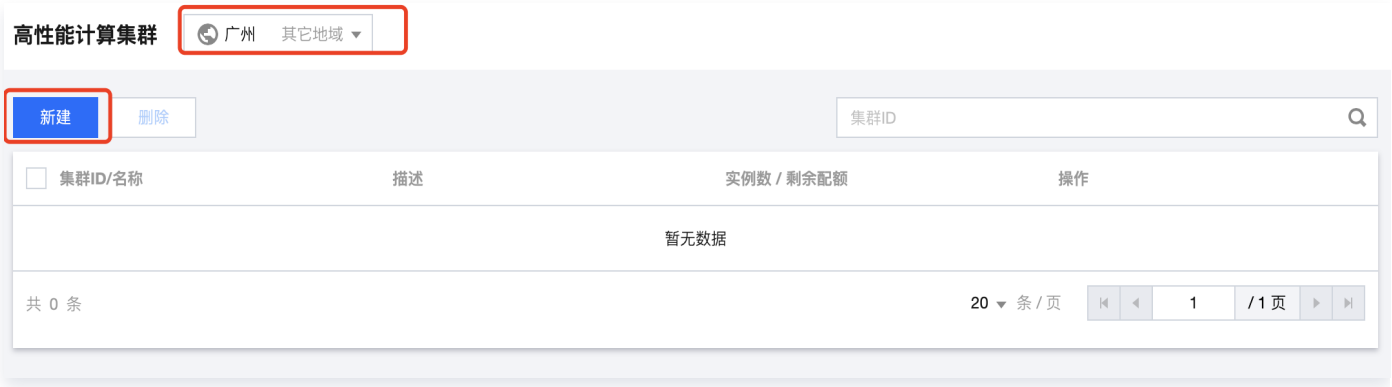
高性能计算集群用于实现高性能计算实例的 RDMA 网络隔离管理。

- 同集群内，实例 RDMA 网络互联互通。
- 跨集群间，实例 RDMA 网络相互隔离。

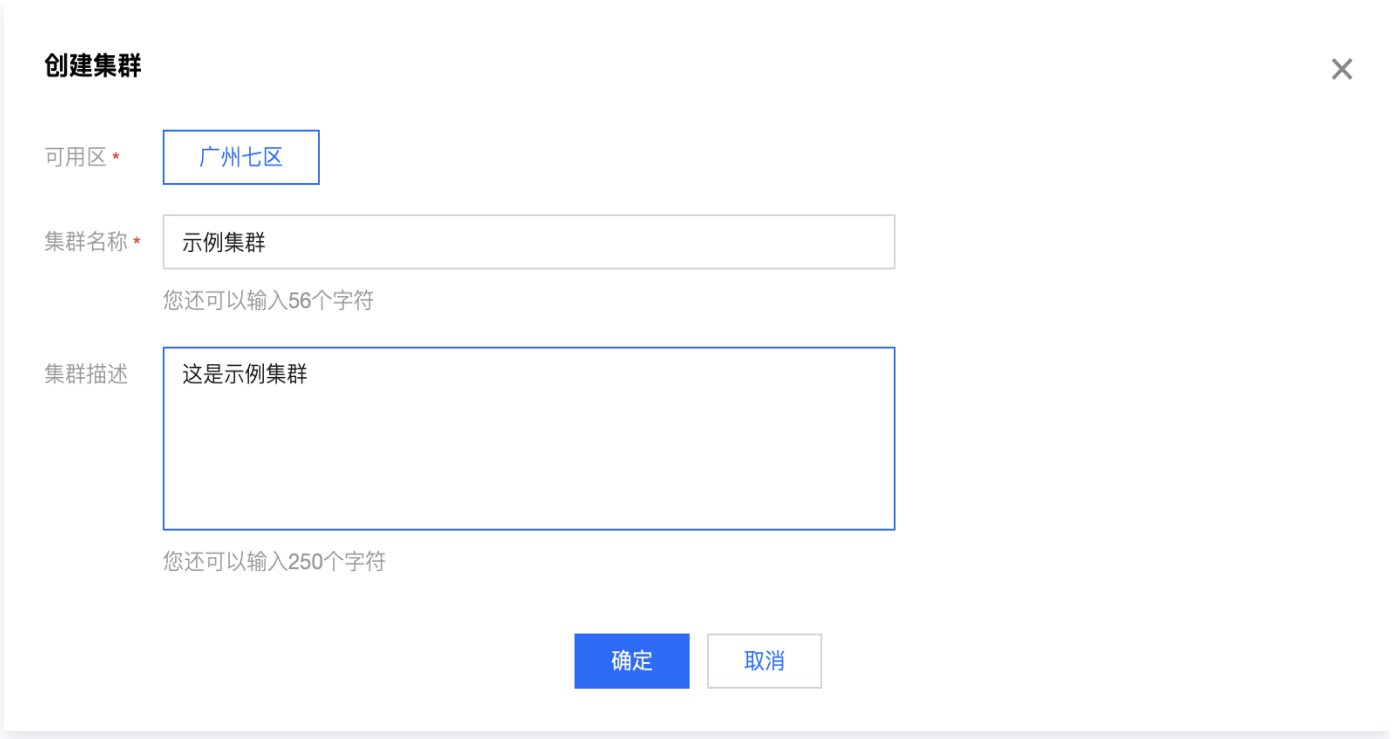
在创建高性能计算**实例**前，您需要首先创建高性能计算**集群**。后续在创建实例时通过选择已有的高性能计算集群，可实现集群内节点高速计算网络互通。
本文介绍高性能计算集群常见的相关操作，例如对集群的创建、修改、扩容、删除等，以下是具体操作步骤：

创建高性能计算集群

1. 登录 云服务器控制台，选择左侧导航栏中的**高性能计算集群**。
2. 在**高性能计算集群**列表页面中，按需选择**地域**。
3. 单击**新建**。



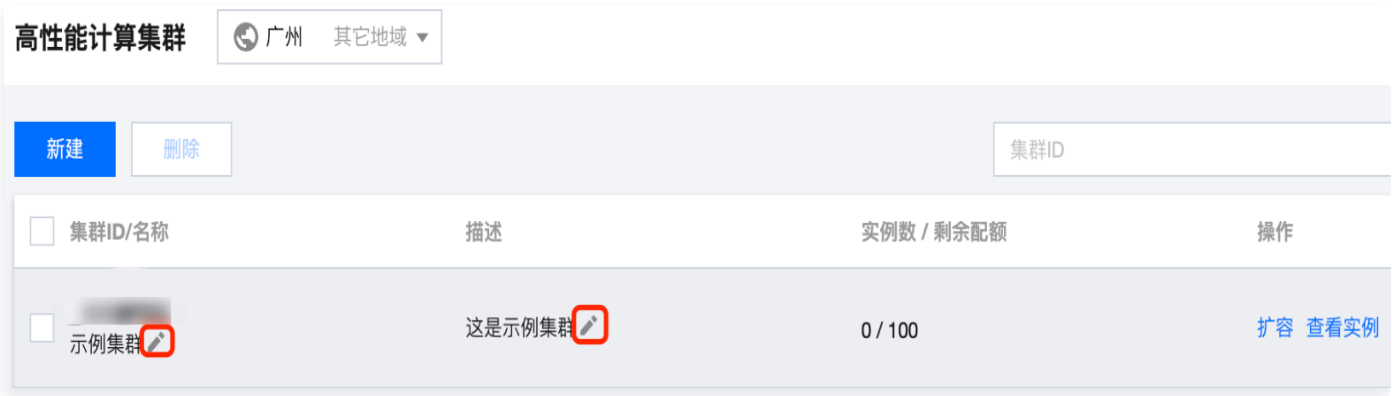
4. 在弹出的**创建集群**窗口中，选择填写**可用区**、**集群名称**、**集群描述**信息。



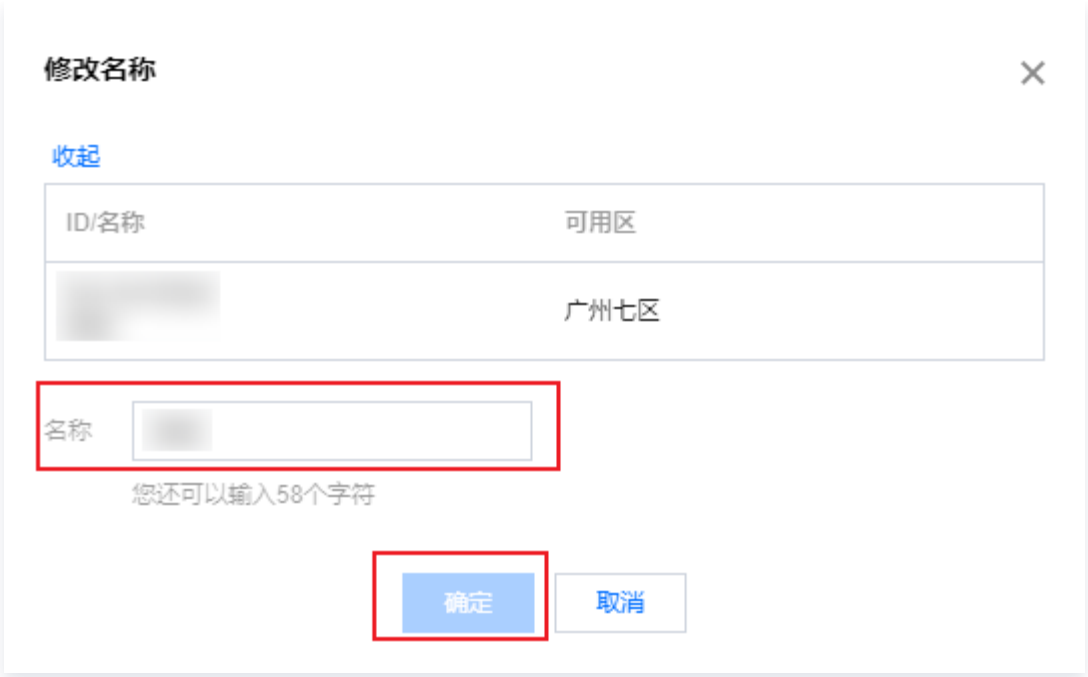
5. 确认信息无误后，单击确定按钮，等待集群创建完成。

修改高性能计算集群信息

1. 登录 云服务器控制台，选择左侧导航栏中的**高性能计算集群**。
2. 在**高性能计算集群**页面，选择需要修改的集群名称或描述右侧的 ，如下图所示。

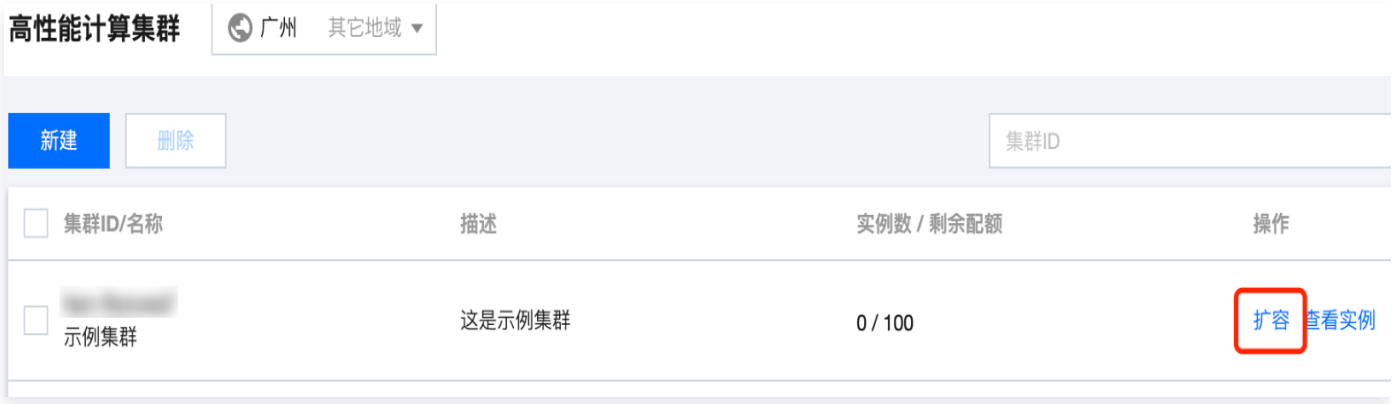


3. 在弹出的**修改名称**或**修改描述**窗口中，输入新的集群名称和集群描述，单击**确定**，完成操作。



扩容高性能计算集群

1. 登录 云服务器控制台，选择左侧导航栏中的**高性能计算集群**。
2. 在**高性能计算集群**页面，选择需要扩容的集群单击**扩容**，进入实例**购买页**。

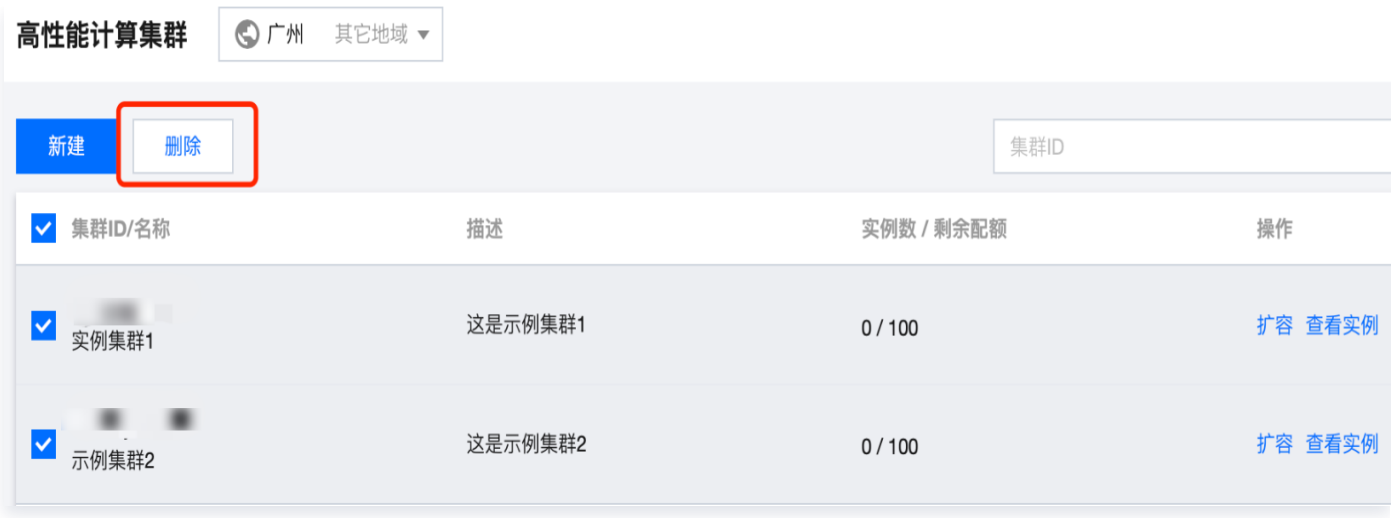


3. 参见 购买高性能计算实例 完成扩容操作。

删除高性能计算集群

> **说明：**
> 若高性能计算集群已部署实例，则该集群无法删除。需销毁集群内全部实例后，才可删除集群。

1. 登录 云服务器控制台，选择左侧导航栏中的高性能计算集群。
2. 在**高性能计算集群**页面，按需勾选一个或多个集群后，单击**删除**。



3. 在弹出的窗口中确认信息，单击**确定**，完成操作。

GPU 型实例安装 nvidia-fabricmanager 服务

最近更新时间：2024-04-17 15:01:02

操作背景

HCCPNV4h 实例搭载了 A100 GPU 并支持 **NvLink & NvSwitch**，需额外安装与驱动版本对应的 nvidia-fabricmanager 服务使 GPU 卡间能够互联。若您使用该实例，请参考本文安装 nvidia-fabricmanager 服务，否则可能无法正常使用 GPU 实例。

操作步骤

本文以驱动版本 **470.103.01** 为例，您可参考以下步骤进行安装，可根据实际情况需要替换 `version` 后的驱动版本。

安装 nvidia-fabricmanager 服务

1. 登录实例，详情请参见 使用标准登录方式登录 Linux 实例 。
2. 不同操作系统版本安装方法不同，请您参考以下方式，执行对应命令进行安装。

CentOS 7.x 镜像

```
version=470.103.01
yum -y install yum-utils
yum-config-manager --add-repo
https://developer.download.nvidia.com/compute/cuda/repos/rhel7/x86_64/cuda-
rhel7.repo
yum install -y nvidia-fabric-manager-${version}-1
```

Ubuntu 18.04 镜像

```
version=470.103.01
main_version=$(echo $version | awk -F '.' '{print $1}')
apt-get updateapt
get -y install nvidia-fabricmanager-${main_version}=${version}-*
```

TencentOS 2.4 镜像

```
version=470.103.01
yum -y install yum-utils
yum-config-manager --add-repo
https://developer.download.nvidia.com/compute/cuda/repos/rhel7/x86_64/cuda-
rhel7.repo
yum install -y nvidia-fabric-manager-${version}-1
```

启动 nvidia-fabricmanager 服务

依次执行以下命令，启动服务。

```
systemctl enable nvidia-fabricmanager
```

```
systemctl start nvidia-fabricmanager
```

查看 nvidia-fabricmanager 服务状态

执行以下命令，查看服务状态。

```
systemctl status nvidia-fabricmanager
```

若输出信息如下，则表示服务安装成功。

```
root@                          # systemctl status nvidia-fabricmanager
● nvidia-fabricmanager.service - NVIDIA fabric manager service
   Loaded: loaded (/lib/systemd/system/nvidia-fabricmanager.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2022-02-22 21:46:13 CST; 39s ago
 Main PID: 45212 (nv-fabricmanage)
    Tasks: 18 (limit: 29491)
   CGroup: /system.slice/nvidia-fabricmanager.service
           └─45212 /usr/bin/nv-fabricmanager -c /usr/share/nvidia/nvswitch/fabricmanager.cfg

Feb 22 21:45:59                systemd[1]: Starting NVIDIA fabric manager service...
Feb 22 21:46:13                nv-fabricmanager[45212]: Successfully configured all the available GPUs and NVSwitches.
Feb 22 21:46:13                systemd[1]: Started NVIDIA fabric manager service.
```

GPU 型实例安装 TCCL 说明

最近更新时间：2024-04-09 15:12:41

TCCL简介

TCCL（Tencent Collective Communication Library）是一款针对腾讯云星脉网络架构的高性能定制加速通信库。主要功能是依托星脉网络硬件架构，为 AI 大模型训练提供更高效的网络通信性能，同时具备网络故障快速感知与自愈的智能运维能力。TCCL 基于开源的 NCCL 代码做了扩展优化，完全兼容 NCCL 的功能与使用方法。TCCL 目前支持主要特性包括：

- 双网口动态聚合优化，发挥 bonding 设备的性能极限。
- 全局 Hash 路由（Global Hash Routing），负载均衡，避免拥塞。
- 拓扑亲和性流量调度，最小化流量绕行。

操作场景

本文介绍如何在腾讯云环境中配置 TCCL 加速通信库，实现您在腾讯云RDMA环境中多机多卡通信的性能提升。在大模型训练场景，对比开源的 NCCL 方案，TCCL 预计约可以提升 50% 带宽利用率。

操作步骤

准备环境

1. 创建 GPU 型 HCCPNV4sne 或 GPU 型 HCCPNV4sn 高性能计算集群实例，分别支持 1.6Tbps 和 800Gbps RDMA 网络。
2. 为 GPU 型实例安装 GPU 驱动和 nvidia-fabricmanager 服务 。

> **注意：**
> TCCL 运行软件环境要求 glibc 版本 2.17 以上，CUDA 版本 10.0 以上。

选择安装方式

TCCL目前支持三种使用方式安装，您可以根据需要选择适合业务场景的安装方式使用。

- TCCL通信库 + 编译安装pytorch
- TCCL通信库 + pytorch通信插件
- NCCL插件 + 排序的IP列表

> **说明：**
> 由于当前大模型训练基本都基于 Pytorch 框架，所以主要以 Pytorch 为例进行说明，

TCCL的三种接入方案对比如下表：

安装方式	方法一：编译安装 Pytorch	方法二：安装 Pytorch 通信插件	（推荐）方法三：安装 NCCL 通信插件
使用步骤	• 安装 TCCL • 重新编译安装 Pytorch	• 安装 Pytorch 通信插件 • 修改分布式通信后端	• 安装 NCCL 插件 • 修改启动脚本
优点	对业务代码无入侵	安装方便	安装方便
缺点	需要重新编译安装 Pytorch 对软件环境有要求	需要修改业务代码 对软件环境有要求	集群节点扩充之后，需要更新排序列表
软件环境依赖	对应 NCCL 版本 2.12 要求 glibc 版本 2.17 以上 要求 CUDA 版本 10.0 以上	当前安装包仅支持 Pytorch 1.12 要求 glibc 版本 2.17 以上 要求 CUDA 版本 10.0 以上	安装 NCCL 即可

如果您的机器资源和模型训练场景相对比较固定，推荐使用方法3，兼容不同的NCCL版本和CUDA版本，安装使用方便，不需要修改业务代码或者重新编译pytorch。
如果您的资源需要提供给不同的业务团队，或者经常有扩容的需求，推荐使用前两种方法，不需要算法人员或者调度框架刻意去感知机器的网络拓扑信息。
如果您不希望对业务代码做适配，那么可以使用方法1，只需要重新编译pytorch框架。

配置 TCCL 环境并验证

方法一： 编译安装 Pytorch

由于社区pytorch默认采用静态方式连接NCCL通信库，所以无法通过替换共享库的方式使用TCCL。

1. 安装TCCL

   以 Ubuntu 20.04 为例，您可以使用以下命令安装，安装之后TCCL位于 /opt/tencent/tccl 目录。

```
# 卸载已有tccl版本和nccl插件
dpkg -r tccl && dpkg -r nccl-rdma-sharp-plugins

# 下载安装tccl v1.5版本
wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/tccl/TCCL_1.5-ubuntu.20.04.5_amd64.deb && dpkg -i TCCL_1.5-ubuntu.20.04.5_amd64.deb && rm -f TCCL_1.5-ubuntu.20.04.5_amd64.deb
```

   如果您使用 CentOS 或 TencentOS，参考以下步骤安装：

```
# 卸载已有tccl版本和nccl插件
rpm -e tccl && rpm -e nccl-rdma-sharp-plugins-1.0-1.x86_64

# 下载tccl v1.5版本
wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/tccl/tccl-1.5-1.tl2.x86_64.rpm && rpm -ivh --nodeps --force tccl-1.5-1.tl2.x86_64.rpm && rm -f tccl-1.5-1.tl2.x86_64.rpm
```

2. 重新编译安装 Pytorch

以下为 Pytorch 源码安装示例，详情参考 官网 Pytorch 安装说明 。

```
#!/bin/bash

# 卸载当前版本
pip uninstall -y torch

# 下载pytorch源码
git clone --recursive https://github.com/pytorch/pytorch
cd pytorch

# <!重要> 配置TCCL的安装路径
export USE_SYSTEM_NCCL=1
export NCCL_INCLUDE_DIR="/opt/tencent/tccl/include"
export NCCL_LIB_DIR="/opt/tencent/tccl/lib"

# 参考官网添加其他编译选项

# 安装开发环境
python setup.py develop
```

3. 配置TCCL环境变量

```
export NCCL_DEBUG=INFO
export NCCL_SOCKET_IFNAME=eth0
export NCCL_IB_GID_INDEX=3
export NCCL_IB_DISABLE=0
export NCCL_IB_HCA=mlx5_bond_0,mlx5_bond_1,mlx5_bond_2,mlx5_bond_3,mlx5_bond_4,mlx5_bond_5,mlx5_bond_6,mlx5_bond_7
export NCCL_NET_GDR_LEVEL=2
export NCCL_IB_QPS_PER_CONNECTION=4
export NCCL_IB_TC=160
export NCCL_IB_TIMEOUT=22
```

```
export NCCL_PXN_DISABLE=0
export TCCL_TOPO_AFFINITY=4
```

> ⚠ **注意：**
> 需要通过TCCL_TOPO_AFFINITY=4开启网络拓扑感知特性。

4. 验证 Pytorch

运行单机多卡或者多机多卡训练过程中有如下打印（export NCCL_DEBUG=INFO）：

```
vm-3-17-centos:74350:74350 [0] NCCL INFO Bootstrap : Using eth0:10.100.3.17<0>
vm-3-17-centos:74350:74350 [0] NCCL INFO NET/Plugin : No plugin found (libnccl-net.so), using internal implementation
vm-3-17-centos:74350:74350 [0] NCCL INFO NET/IB : Using [0]mlx5_bond_0:1/RoCE [RO]; OOB eth0:10.100.3.17<0>
vm-3-17-centos:74350:74350 [0] NCCL INFO Using network IB
NCCL version 2.12.12_TCCL_v1.5+cuda11.6
vm-3-17-centos:74352:74352 [2] NCCL INFO Bootstrap : Using eth0:10.100.3.17<0>
vm-3-17-centos:74352:74352 [2] NCCL INFO NET/Plugin : No plugin found (libnccl-net.so), using internal implementation
vm-3-17-centos:74352:74352 [2] NCCL INFO NET/IB : Using [0]mlx5_bond_0:1/RoCE [RO]; OOB eth0:10.100.3.17<0>
vm-3-17-centos:74352:74352 [2] NCCL INFO Using network IB
```

5. 验证nccl-tests

运行 nccl-tests 之前需要 export 对应的 TCCL路径：

```
export LD_LIBRARY_PATH=/opt/tencent/tccl/lib:$LD_LIBRARY_PATH
```

6. 软件版本支持

目前 TCCL 对应 NCCL 版本 2.12 ，要求 glibc 版本 2.17 以上，CUDA 版本 10.0 以上。其他 CUDA 版本支持请联系您的售前经理获取支持。

方法二：安装 Pytorch 通信插件

Pytorch支持通过插件的方式接入第三方通信后端，所以在不重新编译 Pytorch 的前提下，用户可以使用 TCCL 通信后端，API 与 NCCL 完全兼容。详情可参考 Pytorch 现有通信后端介绍 。

1. 安装 Pytorch 通信插件

```
# 卸载现有的tccl和NCCL插件
dpkg -r tccl && dpkg -r nccl-rdma-sharp-plugins

# 卸载torch_tccl
pip uninstall -y torch-tccl

# 安装torch_tccl 0.0.2版本
```

```
wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/tccl/torch_tccl-0.0.2_pt1.12-py3-none-any.whl && pip install torch_tccl-0.0.2_pt1.12-py3-none-any.whl && rm -f torch_tccl-0.0.2_pt1.12-py3-none-any.whl
```

2. 修改业务代码

```
import torch_tccl

    #args.dist_backend = "nccl"
    args.dist_backend = "tccl"
    torch.distributed.init_process_group(
        backend=args.dist_backend,
        init_method=args.dist_url,
        world_size=args.world_size, rank=args.rank
    )
```

3. 配置TCCL环境变量

```
export NCCL_DEBUG=INFO
export NCCL_SOCKET_IFNAME=eth0
export NCCL_IB_GID_INDEX=3
export NCCL_IB_DISABLE=0
export
NCCL_IB_HCA=mlx5_bond_0,mlx5_bond_1,mlx5_bond_2,mlx5_bond_3,mlx5_bond_4,mlx5_bond_5,mlx5_bond_6,mlx5_bond_7
export NCCL_NET_GDR_LEVEL=2
export NCCL_IB_QPS_PER_CONNECTION=4
export NCCL_IB_TC=160
export NCCL_IB_TIMEOUT=22
export NCCL_PXN_DISABLE=0
export TCCL_TOPO_AFFINITY=4
```

> **注意：**
> 需要通过 TCCL_TOPO_AFFINITY=4 开启网络拓扑感知特性。

4. 验证 Pytorch

您在执行分布式训练业务时，出现如下提示可确认通信后端被正确加载。

```
vm-3-17-centos:35915:35915 [0] NCCL INFO NET/Plugin : No plugin found (libnccl-net.so), using internal implementation
vm-3-17-centos:35915:35915 [0] NCCL INFO NET/IB : Using [0]mlx5_bond_0:1/RoCE [RO]; OOB eth0:10.100.3.17<0>
vm-3-17-centos:35915:35915 [0] NCCL INFO Using network IB
NCCL version 2.12.12_TCCL_v1.5+cuda11.6
vm-3-17-centos:35919:35919 [4] NCCL INFO Bootstrap : Using eth0:10.100.3.17<0>
vm-3-17-centos:35919:35919 [4] NCCL INFO NET/Plugin : No plugin found (libnccl-net.so), using internal implementation
vm-3-17-centos:35921:35921 [6] NCCL INFO Bootstrap : Using eth0:10.100.3.17<0>
vm-3-17-centos:35920:35920 [5] NCCL INFO Bootstrap : Using eth0:10.100.3.17<0>
```

5. 软件版本限制

   当前安装包仅支持Pytorch 1.12，其他 Pytorch 和 CUDA 版本支持请联系您的售前经理获取支持。

> **说明：**
> 如果运行nccl-tests或者其他需要动态链接通信库的场景，请使用方法一安装 TCCL。

（推荐）方法三：安装 NCCL 插件

如果您已经安装了 NCCL ，也可以使用 NCCL 插件的方式使用 TCCL 加速能力。

1. 安装 NCCL 插件

   以 Ubuntu 20.04 为例，您可以使用以下命令安装插件。

```
# 卸载现有的 tccl 和 nccl 插件
dpkg -r tccl && dpkg -r nccl-rdma-sharp-plugins

# 下载安装 nccl 1.2 插件
wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/nccl/nccl-rdma-sharp-plugins_1.2_amd64.deb && dpkg -i nccl-rdma-sharp-plugins_1.2_amd64.deb

# 请确保集群内使用的 nccl 插件版本一致，以下为 nccl 1.0 版本下载安装命令，推荐使用稳定性更优的 nccl 1.2 版本
# wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/nccl/nccl-rdma-sharp-plugins_1.0_amd64.deb && dpkg -i nccl-rdma-sharp-plugins_1.0_amd64.deb && rm -f nccl-rdma-sharp-plugins_1.0_amd64.deb
```

   如果您使用 CentOS 或 TencentOS，参考以下步骤安装：

```
# 卸载现有的 nccl 插件
rpm -e nccl-rdma-sharp-plugins-1.0-1.x86_64
```

```
# 下载安装 nccl 1.2 插件
wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/nccl/nccl-rdma-sharp-plugins-1.2-1.x86_64.rpm && rpm -ivh --nodeps --force nccl-rdma-sharp-plugins-1.2-1.x86_64.rpm

# 请确保集群内使用 nccl 插件版本一致，以下为 nccl 1.0 版本下载安装命令，推荐使用稳定性更优的 nccl 1.2 版本
# wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/nccl/nccl-rdma-sharp-plugins-1.0-1.x86_64.rpm && rpm -ivh --nodeps --force nccl-rdma-sharp-plugins-1.0-1.x86_64.rpm && rm -f nccl-rdma-sharp-plugins-1.0-1.x86_64.rpm
```

2. 获取拓扑排序的 IP 列表

   NCCL 插件不需要依赖文件可提供 bonding 口动态聚合和全局 hash 路由两种优化。如果需要支持网络拓扑的亲和性感知，用户可以通过排序的IP列表来实现。

   IP 排序可以按照如下方式完成：

   - 准备 IP 列表文件

     VPC IP 地址通过 `ifconfig eth0` 获取，每行1个节点 IP，格式如下：

     ```
     root@VM-125-10-tencentos:/workspace# cat ip_eth0.txt
     172.16.177.28
     172.16.176.11
     172.16.177.25
     172.16.177.12
     ```

   - 执行排序

     ```
     wget https://taco-1251783334.cos.ap-shanghai.myqcloud.com/tccl/get_rdma_order_by_ip.sh && bash get_rdma_order_by_ip.sh ip_eth0.txt
     ```

     > ⚠ 注意：
     > - 所有节点都安装了 curl 工具（比如对于 Ubuntu，通过 `apt install curl` 安装）。
     > - 执行脚本的节点可以 ssh 免密访问其他所有节点。

   - 查看排序后的 IP 列表文件

     ```
     root@VM-125-10-tencentos:/workspace# cat hostfile.txt
     172.16.176.11
     172.16.177.12
     ```

```
172.16.177.25
172.16.177.28
```

3. 配置 TCCL 环境变量

```
export NCCL_DEBUG=INFO
export NCCL_SOCKET_IFNAME=eth0
export NCCL_IB_GID_INDEX=3
export NCCL_IB_DISABLE=0
export
NCCL_IB_HCA=mlx5_bond_0,mlx5_bond_1,mlx5_bond_2,mlx5_bond_3,mlx5_b
ond_4,mlx5_bond_5,mlx5_bond_6,mlx5_bond_7
export NCCL_NET_GDR_LEVEL=2
export NCCL_IB_QPS_PER_CONNECTION=4
export NCCL_IB_TC=160
export NCCL_IB_TIMEOUT=22
export NCCL_PXN_DISABLE=0

# 机器IP手动排序之后，就不需要添加如下变量了
# export TCCL_TOPO_AFFINITY=4
```

4. 修改启动脚本

您需要在启动分布式训练时修改启动脚本。例如，如果使用 deepspeed launcher 启动训练进程，拿到排序后的 IP 列表之后，将对应的 IP 列表写入 hostfile，再启动训练进程。

```
root@vm-3-17-centos:/workspace/ptm/gpt# cat hostfile
172.16.176.11 slots=8
172.16.177.12 slots=8
172.16.177.25 slots=8
172.16.177.28 slots=8

deepspeed --hostfile ./hostfile --master_addr 172.16.176.11 train.py
```

如果使用 torchrun 启动训练进程，通过 `--node_rank` 指定对应的节点顺序，

```
// on 172.16.176.11
torchrun --nnodes=4 --nproc_per_node=8 --node_rank=0 --
master_addr=172.16.176.11 train.py ...
// on 172.16.176.12
torchrun --nnodes=4 --nproc_per_node=8 --node_rank=1 --
master_addr=172.16.176.11 train.py ...
// on 172.16.176.25
```

```
torchrun --nnodes=4 --nproc_per_node=8 --node_rank=2 --
master_addr=172.16.176.11 train.py ...
// on 172.16.176.28
torchrun --nnodes=4 --nproc_per_node=8 --node_rank=3 --
master_addr=172.16.176.11 train.py ...
```

如果使用 mpirun 启动训练进程，按照顺序排列 IP 即可。

```
mpirun \
-np 64 \
-H 172.16.176.11:8,172.16.177.12:8,172.16.177.25:8,172.16.177.28:8 \
--allow-run-as-root \
-bind-to none -map-by slot \
-x NCCL_DEBUG=INFO
-x NCCL_IB_GID_INDEX=3 \
-x NCCL_IB_DISABLE=0 \
-x NCCL_SOCKET_IFNAME=eth0 \
-x
NCCL_IB_HCA=mlx5_bond_0,mlx5_bond_1,mlx5_bond_2,mlx5_bond_3,mlx5_b
ond_4,mlx5_bond_5,mlx5_bond_6,mlx5_bond_7 \
-x NCCL_NET_GDR_LEVEL=2 \
-x NCCL_IB_QPS_PER_CONNECTION=4 \
-x NCCL_IB_TC=160 \
-x NCCL_IB_TIMEOUT=22 \
-x NCCL_PXN_DISABLE=0 \
-x LD_LIBRARY_PATH -x PATH \
-mca coll_hcoll_enable 0 \
-mca pml ob1 \
-mca btl_tcp_if_include eth0 \
-mca btl ^openib \
all_reduce_perf -b 1G -e 1G -n 1000 -g 1
```

GPU 型实例安装 RDMA 毫秒级监控组件

最近更新时间：2024-04-09 15:12:41

功能简介

高性能计算集群具备在 RDMA 网络环境下实现毫秒级监控的能力，这使得您能够实时监测和分析瞬时的网络数据，帮助您深入分析网络流量模式，进行网络优化和性能提升，为业务提供有力支持。

操作场景

本文介绍如何在腾讯云高性能计算集群环境中安装毫秒级监控组件，实现您在腾讯云 RDMA 环境中毫秒级的性能监控。腾讯云提供两种监控数据的查看方式，您可以选择在云产品监控上查看毫秒级监控的统计数据或在实例本地查看保存的监控日志。

> ⚠ **注意：**
> RDMA 毫秒级监控启动后约会占用小于 0.05 个核资源，可根据业务需要判断是否使用。

操作步骤

准备环境

1. 创建 GPU 型 HCCPNV4sne、GPU 型 HCCPNV4sn 或 GPU 型 HCCPNV5v 高性能计算集群实例，镜像建议选择 TencentOS Server 2.4 (TK4)。
2. 为 GPU 型实例安装 GPU 驱动和 nvidia-fabricmanager 服务。

安装验证

1. 在 TencentOS Server 2.4 (TK4) 环境下，您可以使用以下命令安装：

```
# 卸载已有增强型监控软件包
rpm -e rdma_monitor-1.0-1.tl2.x86_64
# 下载并安装毫秒级监控组件，
# 安装好软件包后，会自动注册系统服务来启动增强型监控并保活，无需手动启动
wget http://mirrors.tencentyun.com/install/GPU/rdma_monitor-1.0-1.tl2.x86_64.rpm
&& rpm -ivh rdma_monitor-1.0-1.tl2.x86_64.rpm
```

2. 使用以下命令，验证是否安装成功：

```
ps -aux | grep monitor_server
```

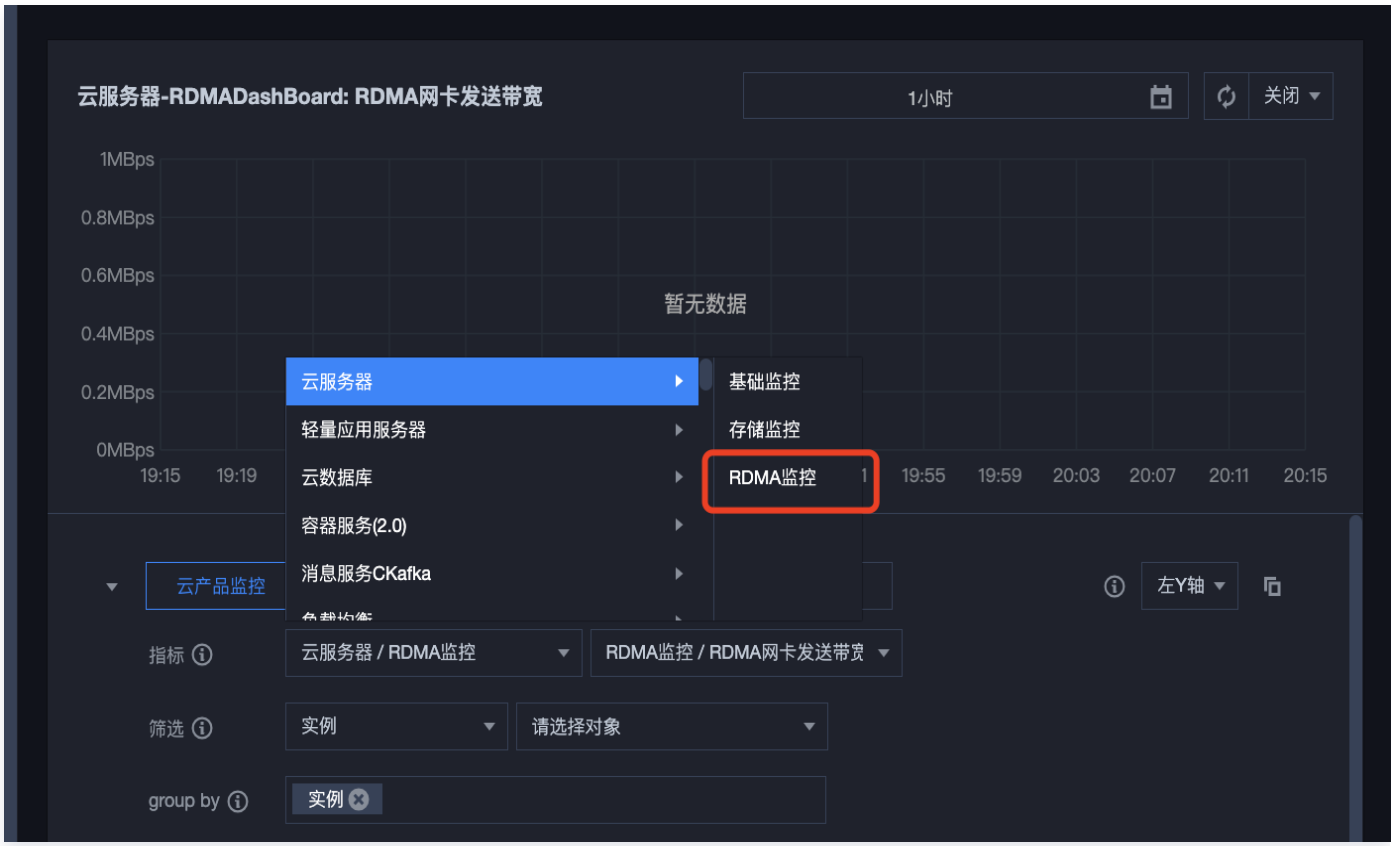
执行命令，如果红字所示字段，代表增强型监控成功安装启动。

```
[root@VM-16-2-tencentos ~]# ps -aux | grep monitor_server
root        3719  0.0  0.0 112824  2260 pts/5    S+   15:26   0:00 grep --color=auto monitor_server
root      230127  1.8  0.0 1587160 25080 pts/0   Sl+  15:20   0:06 monitor_server -m 3
```

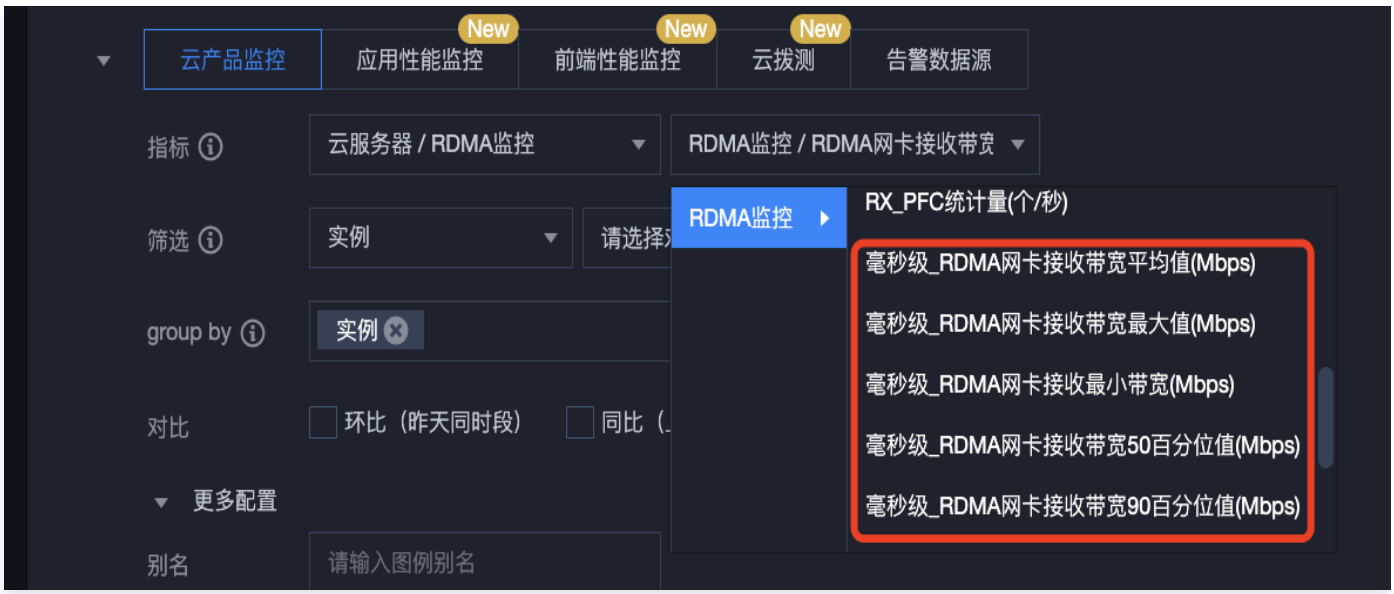
云产品监控查看

RDMA 毫秒级监控可在云产品监控查看统计数据，您可以在云产品监控-dashboard 中配置您需要的监控指标，操作步骤如下：

1. 新建 dashboard，指标选择 **云服务器-RDMA 监控**：



2. 选择您需要监控的 RDMA 毫秒级统计指标。



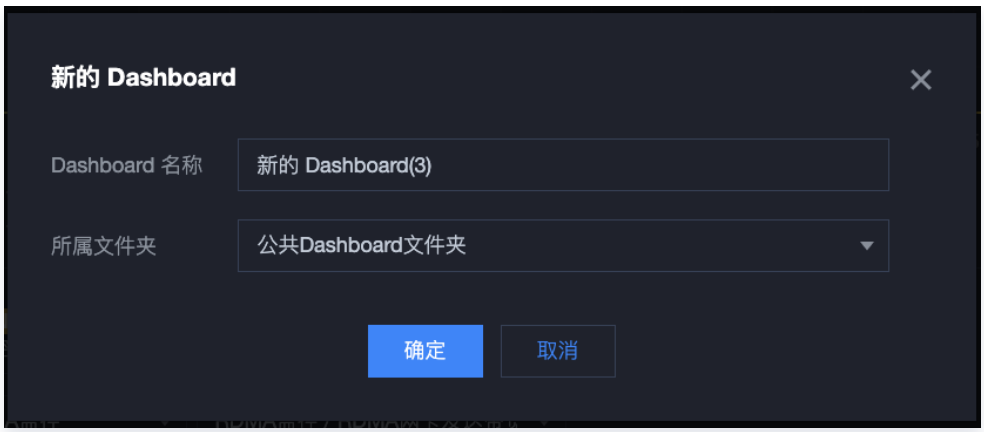
云产品监控支持查看以下统计数据，您可以根据需要在云产品监控 dashboard 配置。

指标英文名	指标中文名	指标说明（非必填）	单位	维度	统计粒度
RxHpbwAvg	毫秒级_RDMA 网卡接收带宽平均值	10秒内 RDMA 网卡接收带宽的毫秒级统计粒度平均值	Mbps	Instanceld	10s、60s、300s、3600s
RxHpbwMax	毫秒级_RDMA 网卡接收带宽最大值	10秒内 RDMA 网卡接收带宽的毫秒级统计粒度最大值	Mbps	Instanceld	10s、60s、300s、3600s
RxHpbwMin	毫秒级_RDMA 网卡接收带宽最小值	10秒内 RDMA 网卡接收带宽的毫秒级统计粒度最小值	Mbps	Instanceld	10s、60s、300s、3600s
RxHpbwP50	毫秒级_RDMA 网卡接收带宽50百分位值	10秒内从小到大 RDMA 网卡接收带宽的毫秒级统计粒度前50百分位数	Mbps	Instanceld	10s、60s、300s、3600s、86400s
RxHpbwP90	毫秒级	10秒内从小到	M	Instan	10s、60s、

	_RDMA 网卡接收带宽90百分位值	大 RDMA 网卡接收带宽的毫秒级统计粒度前90百分位数	bps	celd	300s、3600s
TxHpbwAvg	毫秒级_RDMA 网卡发送带宽平均值	10秒内 RDMA 网卡发送带宽的毫秒级统计粒度平均值	Mbps	Instanceld	10s、60s、300s、3600s
TxHpbwMax	毫秒级_RDMA 网卡发送带宽最大值	10秒内 RDMA 网卡发送带宽的毫秒级统计粒度最大值	Mbps	Instanceld	10s、60s、300s、3600s
TxHpbwMin	毫秒级_RDMA 网卡发送带宽最小值	10秒内 RDMA 网卡发送带宽的毫秒级统计粒度最小值	Mbps	Instanceld	10s、60s、300s、3600s
TxHpbwP50	毫秒级_RDMA 网卡发送带宽50百分位	10秒内从小到大 RDMA 网卡发送带宽毫秒级统计粒度前50百分位数	Mbps	Instanceld	10s、60s、300s、3600s
TxHpbwP90	毫秒级_RDMA 网卡发送带宽90百分位	10秒内从小到大 RDMA 网卡发送带宽毫秒级统计粒度前90百分位数	Mbps	Instanceld	10s、60s、300s、3600s

3. 选择需要监控的高性能计算集群实例 ID。

4. 单击**确定**即可快速创建 Dashboard。



本地监控查看

RDMA 毫秒级监控可查看最小 10ms 粒度级别的带宽数据监控，但云产品监控只支持最小粒度为10s的数据上报。如果用户想获取更精确的网卡监控数据，可以使用如下命令，保存毫秒级的数据在本地查看。

```
# monitor_client 随增强型监控已自动安装，/tmp/monitor.log 为自定义的数据保存路径，文件大小会持续增长，注意管理存储空间
monitor_client -r -p raw > /tmp/monitor.log
# -r 持续取最近10s的数据
# -p 打印选择
#  -p summary，默认值，打印统计信息
#  -p raw，打印原始数据点
#  -p all，打印统计信息和原始数据点
# 您可以使用 monitor_client -h 查看更多参数说明
```

查看记录的监控数据，您可以根据需要分析监控记录，监控记录的格式如下：

```
Device: bond3, Transmitted data points: 1000, Timestamp: 1697616573
Data Point 0: 0
Data Point 1: 0
Data Point 2: 0
Data Point 3: 0
Data Point 4: 0
Data Point 5: 0
Data Point 6: 0
Data Point 7: 0
Data Point 8: 0
Data Point 9: 0
Data Point 10: 0
Data Point 11: 0
Data Point 12: 0
Data Point 13: 0
Data Point 14: 0
Data Point 15: 0
Data Point 16: 0
```

说明：

图中部分参数含义解释如下：

- Device：RDMA 网卡的名称。
- Received data points：接收侧10s内采集到的数据点数，这里是10s内采集了1000个点，也就是每10ms采集一次数据点，每个点的数据为对应10ms的接收带宽。
- Timestamp：采集时的时间戳。
- Data Point n：自时间戳n*10ms后采集到的接收带宽。每个点的采样时间与前后的点均间隔10ms。