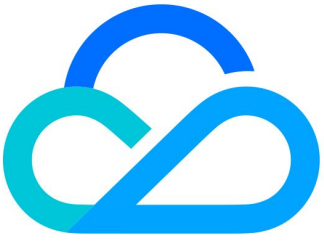
# 高性能计算集群

# 操作指南



腾讯云

【版权声明】



【商标声明】



【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

# 文档目录

操作指南
- 管理高性能计算集群
- 集群自助检测
- GPU 型实例安装 nvidia-fabricmanager 服务
- GPU 型实例安装 TCCL
- GPU 型实例安装 RDMA 毫秒级监控组件
- GPU 型实例监控和告警

# 操作指南
# 管理高性能计算集群

最近更新时间：2025-05-07 16:54:12

## 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 ，在左侧导航栏选择**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**，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 ，在左侧导航栏选择**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，在左侧导航栏选择**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 购买方式 完成扩容操作。

## 删除高性能计算集群

> **注意：**
> 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，在左侧导航栏选择**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。

# 集群自助检测

最近更新时间：2025-01-06 10:56:32

## 概述

高性能计算集群的自助检测（即集群一致性检测）功能提供集群维度的实例检测，您可以检测集群中所有实例的硬件和软件状态。您可通过该功能及时发现并解决集群实例的相关问题。

## 操作场景

以下两种场景推荐使用集群自助检测：

- **集群故障排查**：在日常运维过程中，您可以使用集群自助检测功能，检测集群中所有实例的硬件和软件状态，并根据相应建议对异常情况进行处理。
- **大规模 AI 模型训练环境检测**：集群训练需要保障硬件、GPU 驱动、CUDA、NCCL 和 RDMA 等配置状态的可用性和一致性，自助检测功能提供集群维度的实例检测能力，保障训练正常运行。

## 检测项说明

集群自助检测功能支持诊断硬件和软件配置的一致性并提供诊断报告。

**集群训练环境**

| 检测项 | 检测说明 | 风险等级 | 解决方案 |
| --- | --- | --- | --- |
| 集群训练环境 GPU 型号 | 实例的 GPU 型号 | 异常 | 更换为同种型号 GPU 机型。 |
| 集群训练环境 CPU 型号 | 实例的 CPU 型号 | 异常 | 更换为同种型号 CPU 机型。 |
| 集群训练环境 GPU 卡数 | 实例的 GPU 卡总数 | 异常 | 实例可能存在 GPU 卡异常，请您关注 CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常显卡的硬件维护。 |
| 集群训练环境内核版本 | 实例的操作系统内核版本 | 警告 | 操作系统内核版本不一致，建议更换同操作系统保持内核一致。 |
| 集群训练环境 OS 版本 | 实例的操作系统版本 | 异常 | 建议更换操作系统保持版本一致。 |

| | | | |
|---|---|---|---|
| 集群训练环境 OFED 版本 | 实例的网卡驱动版本 | 异常 | 建议使用 HCC 公有镜像，已安装 OFED 驱动。 |
| 集群训练环境 GPU 驱动版本 | 实例的 GPU 驱动版本 | 异常 | 建议保持 GPU 驱动版本一致，您可以使用 自动安装驱动功能 在创建或重装实例时指定 GPU 驱动版本。 |
| 集群训练环境 CUDA Runtime 版本 | 实例的 CUDA Runtime 版本 | 异常 | 建议保持 CUDA 版本一致，您可以使用 自动安装驱动功能 在创建或重装实例时指定 CUDA 版本。 |
| 集群训练环境 CUDA Driver 版本 | 实例的 CUDA 驱动版本 | 异常 | 建议保持 CUDA 版本一致，您可以使用 自动安装驱动功能 在创建或重装实例时指定 CUDA 版本。 |
| 集群训练环境 GDR 状态 | 实例是否加载 nvidia_peermem 模块。nvidia_peermem 是 NVIDIA 提供的一个内核模块，通常用于支持设备间直接内存访问（peer-to-peer memory access），在高性能计算（HPC）和深度学习应用中用于优化 GPU 间的数据传输效率，对多机多卡训练性能有影响。 | 异常 | 默认自动安装驱动后自动加载 nvidia_peermem 模块，若检测未加载 nvidia_peermem 模块，请确保 GDR 挂载状态，参考如下步骤处理：<br>`modprobe nvidia_peermem && echo "modprobe nvidia_peermem" >>/etc/rc.local` |
| 集群训练环境 NCCL 版本 | 实例的 NCCL 版本。NCCL（NVIDIA Collective Communications Library）是一个 NVIDIA GPU 集合通信库，用于多 GPU 或多个节点通信，是 AI 分布式训练的必选软件。 | 异常 | 建议所有实例 NCCL 版本保持一致，安装方法请参见 NVIDIA NCCL 官方说明 。 |
| 集群训练环境 NCCL-plugin 版本 | 实例的 NCCL-plugin 版本。NCCL-plugin 是一款针对腾讯云星脉网络架构的高性能定制加速 | 异常 | 建议使用统一版本 NCCL-plugin，详情请参见 GPU 型实例安装 TCCL 说明 。 |

| | | | |
|---|---|---|---|
| | 通信库插件，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，详情请参见 GPU 型实例安装 TCCL 说明 。 | | |

### 集群 RDMA

| 检测项 | 检测说明 | 风险等级 | 解决方案 |
|---|---|---|---|
| 集群 RDMA Bonding 状态 | 检查特定的网卡绑定接口（bonding interface）为 up 状态的网卡数量。 | 异常 | 若存在网卡绑定状态未开启，请执行 `ifup bondX` （ **bondX** 是要重启的网卡的名称，例如 **bond0** ）将网卡开启，以便 RDMA 网络正常使用。 |
| 集群 RDMA MTU 配置 | 当前网络接口的最大传输单元（MTU）。MTU 是指在一个网络数据包中，数据部分的最大字节数。不同的网络接口和网络环境可能有不同的最佳 MTU 值。 | 异常 | 若检测 MTU 值异常，建议调整 MTU 值为9100。 |
| 集群 RDMA 网卡速率 | 了解网络接口的速度，以确保网络设备和连接正常工作。 | 异常 | 若检测网卡速率不符合预期，请关注 CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常网卡的维护。 |
| 集群 RDMA rp_filter 状态 | 读取或设置 Linux 系统中某个网络接口的反向路径过滤（Reverse Path Filtering）。 | 异常 | 若检测到 rp_filter 配置异常，请联系腾讯云处理。 |
| 集群 RDMA QoS 配置 | 读取 QoS 配置信息，配置影响数据包的处理优先级和质量，可能具有 0 到 255 之间的数值，用来表示不同的优先级或服务质量，一般 | 异常 | 若检测到您的端口 QoS 配置问题，请执行如下命令修复：<br>`bash /usr/local/qcloud/rdma/dscp.sh` |

| | | | |
|---|---|---|---|
| | 是 160。所有实例配置需要保持一致。 | | |
| 集群 RDMA CRC 异常 | 查询并过滤特定网络接口的统计信息，特别是接收数据包时的 CRC 错误统计。CRC 是一种用于检测错误的校验方法，如果一个数据包的 CRC 校验失败，表明该数据包在传输过程中可能被损坏。 | 警告 | 若该计数持续有增加，请联系腾讯云处理。 |
| 集群 RDMA ARP 双发检查 | 用于配置和管理网络接口绑定（bonding）设置中的广播 ARP（Address Resolution Protocol）检测功能。广播 ARP 检测是网络绑定中的一种链路监控方法。通过发送 ARP 请求并监听响应，来检测链路的可用性。 | 异常 | 若检测到 ARP 配置问题，请联系腾讯云处理。 |
| 集群 RDMA rpg_time_reset 拥塞控制参数 | RoCE（RDMA over Converged Ethernet）网络接口的 RPG（Rate-Parity Group）时间重置相关的设置。RPG 时间重置是 RoCE 网络接口的一种功能，用于在网络链路出现异常情况时执行重置操作，以确保网络的稳定性和可靠性。重置操作可能涉及重新初始化或重新同步网络接口的一些参数或状态。 | 警告 | 拥塞控制参数不一致，如观测到业务训练速度变慢，请联系腾讯云处理。 |
| 集群 RDMA rpg_ai_rate 拥塞控制参数 | 用于配置 RoCE（RDMA over Converged Ethernet）网络接口 | 警告 | 拥塞控制参数不一致，如观测到业务训练速度变慢，请联系腾讯云处理。 |

| | | | |
|---|---|---|---|
| | 中 RPG（Rate-Parity Group）的自适应传输速率。自适应传输速率是 RoCE 网络接口的一种功能，允许在传输数据时动态调整传输速率，以根据网络负载和性能需求实现最佳的数据传输效率。 | | |
| 集群 RDMA rpg_byte_reset 拥塞控制参数 | RoCE（RDMA over Converged Ethernet）网络接口的 Rate-Parity Group（RPG）的字节重置设置。 | 警告 | 拥塞控制参数不一致，如观测到业务训练速度变慢，请联系腾讯云处理。 |
| 集群 RDMA rate_reduce_monitor_period拥塞控制参数 | RoCE 网络接口的 Rate-Parity Group（RPG）的速率降低监控周期（Rate Reduce Monitor Period）。 | 警告 | 拥塞控制参数不一致，如观测到业务训练速度变慢，请联系腾讯云处理。 |

**集群单机检测**

| 检测项 | 检测说明 | 风险等级 | 解决方案 |
|---|---|---|---|
| 集群单机检测 GPU 硬件降速 | 检测硬件热量减速功能当前是否处于激活状态 | 异常 | 若检测到实例 GPU 被降速，请您关注 CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常显卡的硬件维护。 |
| 集群单机检测 GPU 软件降速 | 检测软件热量减速功能当前是否处于激活状态 | 异常 | 若检测到实例存在 GPU 运行时 被降速，请您关注CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常显卡的硬件维护。 |
| 集群单机检测 GPU XID 异常 | GPU 是否存在 XID 报错 | 异常 | 若检测到实例存在 GPU XID 错误，请您关注 CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常显卡的硬件维护。 |

| | | | |
|---|---|---|---|
| 集群单机检测 NVLink 激活失败 | 检测 NVLink 激活失败个数 | 异常 | 若检测失败，请确认 nvidia-fabricmanager 服务是否正常。若服务异常，请启动服务再验证是否正常；若服务正常，请重启实例再验证是否正常。 |
| 集群单机检测 NVSwitch 初始化失败 | 检测 NVSwitch 初始化失败 | 异常 | 请重启机器再检测是否可以恢复，若无法恢复请联系腾讯云处理。 |
| 集群单机检测 FabricManager 服务报错 | 检测 FabricManager 服务报错个数 | 异常 | 若实例没有启动 nvidia-fabricmanager 服务，可参见 安装 nvidia-fabricmanager 服务 。 |
| 集群单机检测 NVLink P2P 状态异常 | 检测 NVLink P2P 状态异常 | 异常 | 若检测到实例存在 GPU 异常，请您关注 CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常显卡的硬件维护。 |
| 集群单机检测 PCIe 链路故障 | 检测 PCIe 链路故障个数 | 异常 | 若检测到实例存在 GPU 异常，请您关注 CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常显卡的硬件维护。 |
| 集群单机检测 Row remapper 状态 | 检测 Row remapper 失败的 GPU 故障的个数 | 异常 | 若检测到实例存在 GPU 异常，请您关注 CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常显卡的硬件维护。 |
| 集群单机检测 SRAM UCE 超标的 GPU 故障 | 检测 SRAM UCE 超标的 GPU故障个数 | 异常 | 若检测到实例存在 GPU 异常，请您关注 CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常显卡的硬件维护。 |
| 集群单机检测驱动状态 | 检测驱动状态 | 异常 | 若检测到实例存在 GPU 异常，请您关注 CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常显卡的硬件维护。 |
| 集群单机检测 GPU 初始化状态 | 检测 GPU 初始化状态 | 异常 | 若检测到实例存在 GPU 异常，请您关注 CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常显卡的硬件维护。 |

| | | | |
|---|---|---|---|
| 集群单机检测 InfoROM 状态 | 检测 InfoROM 状态 | 异常 | 若检测到实例存在 GPU 异常，请您关注 CVM 控制台维修任务中是否存在 维修任务并授权腾讯云 进行异常显卡的硬件维护。 |

# 前提条件

实例需安装自动化助手后才可以使用集群自助检测能力。如未安装，可参见 安装自动化助手客户端 进行安装。

# 相关操作

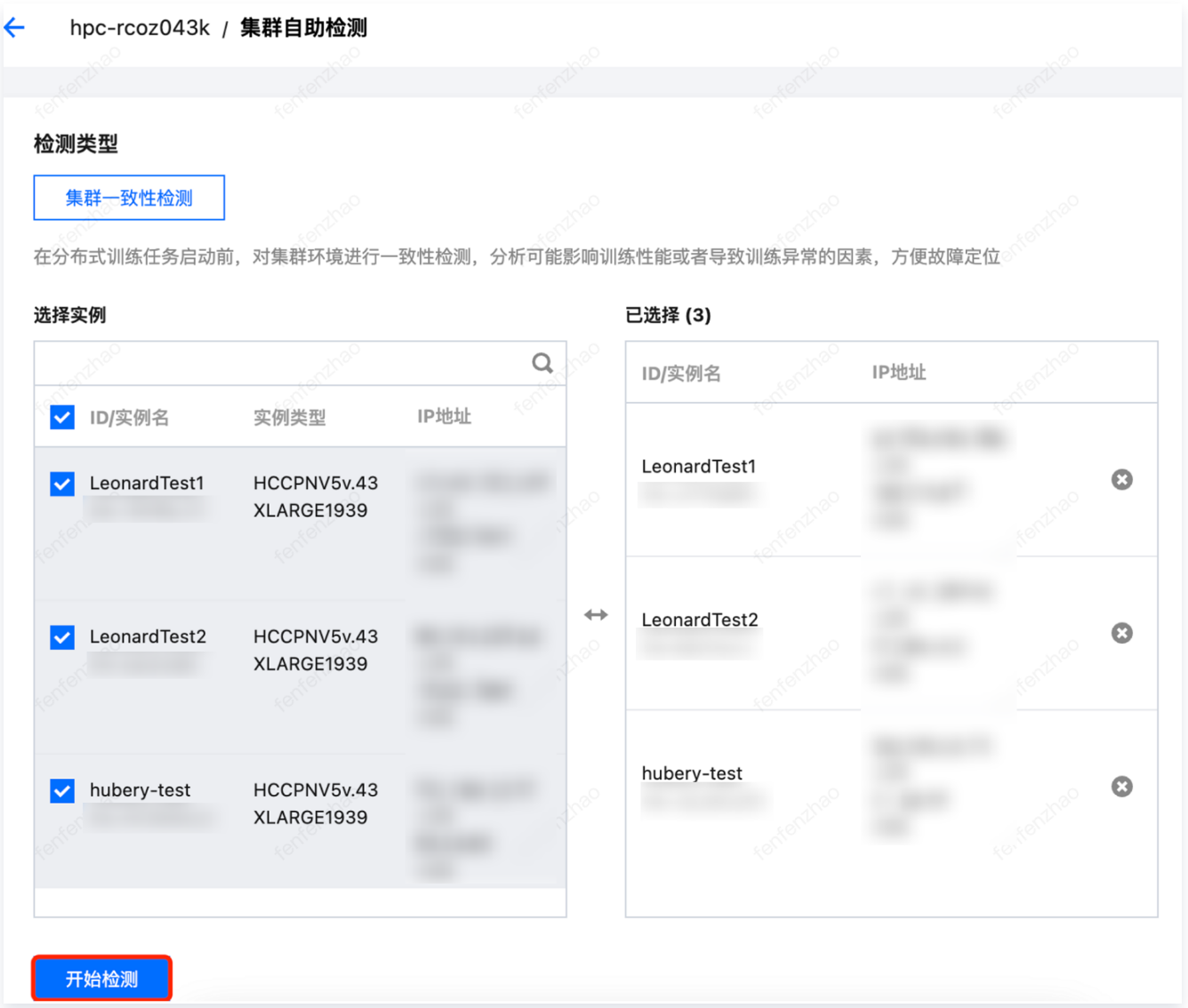
集群自助检测功能支持诊断硬件和软件配置一致性并提供诊断报告。

## 启动检测

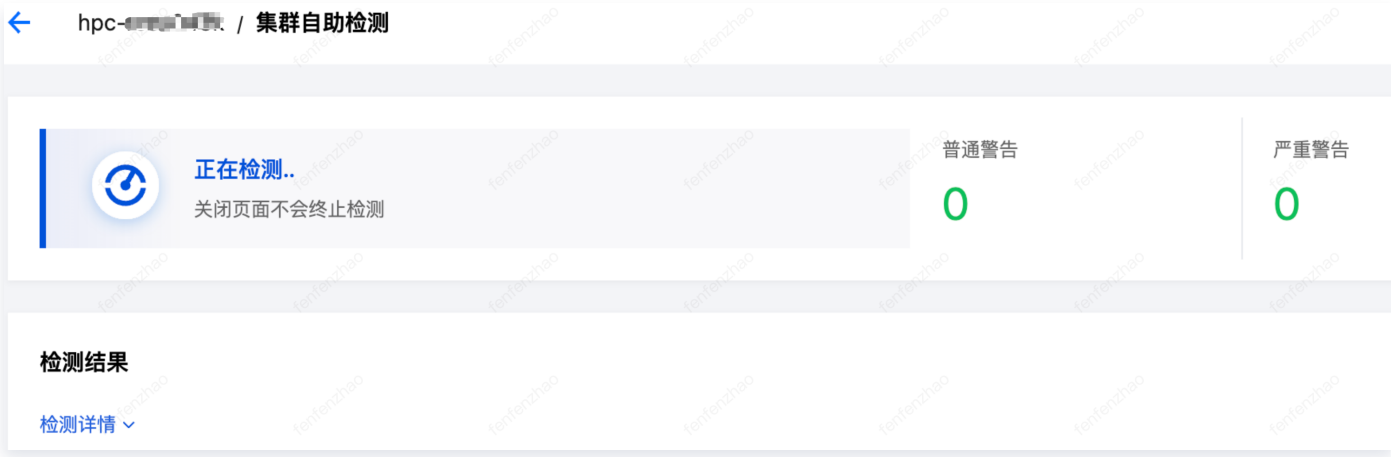
1. 登录 云服务器控制台，在左侧导航栏选择**高性能计算集群**。
2. 在**高性能计算集群**列表页面中，选择集群所在**地域**。
3. 单击**集群 ID**，进入集群详情页面。


4. 在**自助检测**页签，单击**集群自助检测**。


5. 选择本次检测的实例，单击**开始检测**。




开始检测后，需要稍等几分钟。



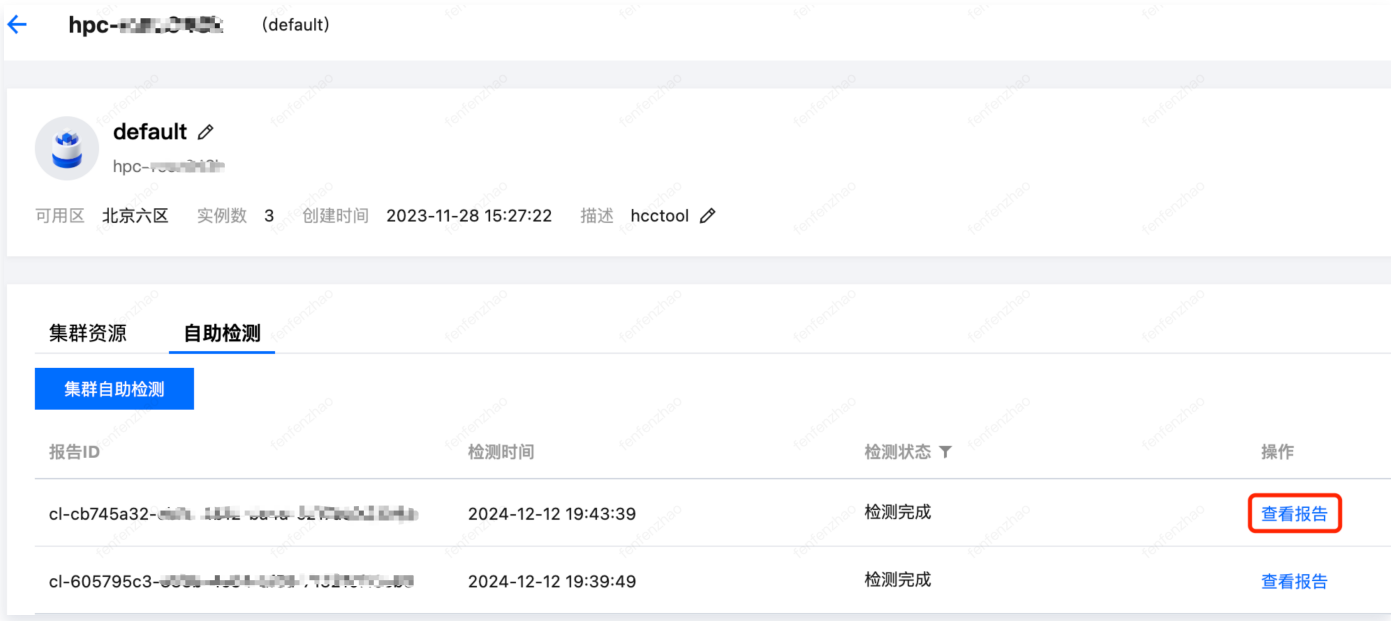

检测完成，页面会显示检测结果。


# 查看历史报告

1. 登录 云服务器控制台，在左侧导航栏选择**高性能计算集群**。
2. 在**高性能计算集群**列表页面中，选择集群所在**地域**。

3. 单击**集群 ID**，进入集群详情页面。


4. 单击**查看报告**，即可查看历史检测报告。



# GPU 型实例安装 nvidia-fabricmanager 服务

最近更新时间：2025-05-06 10:26:22

## 操作场景

高性能计算集群实例搭载了 A100/A800/H800 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。如果您使用该实例，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 **535.216.01** 为例，您可以参考以下步骤进行安装，您也可以根据实际情况替换其他版本。

## 安装 nvidia-fabricmanager 服务

1. 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，替换对应安装包路径，执行命令进行安装。不同镜像和驱动版本匹配不同版本的安装包，更多版本选择可 进入 NV 官网 查看。

> **说明：**
> - NVIDIA GPU 型实例升级 GPU 驱动的同时，还需同步升级 Fabric Manager，否则无法正常使用 GPU 实例。
> - 使用 购买页自动安装驱动 功能将自动安装 Fabric Manager。

### TencentOS 3.1 镜像

```
wget
https://developer.download.nvidia.cn/compute/cuda/repos/rhel8/x86_64
/nvidia-fabric-manager-535.216.01-1.x86_64.rpm
wget
https://developer.download.nvidia.cn/compute/cuda/repos/rhel8/x86_64
/nvidia-fabric-manager-devel-535.216.01-1.x86_64.rpm
rpm -ivh nvidia-fabric-manager-535.216.01-1.x86_64.rpm
rpm -ivh nvidia-fabric-manager-devel-535.216.01-1.x86_64.rpm
```

### Ubuntu 22.04 镜像

```
wget
https://developer.download.nvidia.cn/compute/cuda/repos/ubuntu2204/x
86_64/nvidia-fabricmanager-535_535.216.01-1_amd64.deb
wget
https://developer.download.nvidia.cn/compute/cuda/repos/ubuntu2204/x
86_64/nvidia-fabricmanager-dev-535_535.216.01-1_amd64.deb
sudo dpkg -i nvidia-fabricmanager-535_535.216.01-1_amd64.deb
sudo dpkg -i nvidia-fabricmanager-dev-535_535.216.01-1_amd64.deb
```

### CentOS 7.x 镜像

```
wget
https://developer.download.nvidia.cn/compute/cuda/repos/rhel7/x86_64
/nvidia-fabric-manager-535.216.01-1.x86_64.rpm
wget
https://developer.download.nvidia.cn/compute/cuda/repos/rhel7/x86_64
/nvidia-fabric-manager-devel-535.216.01-1.x86_64.rpm
rpm -ivh nvidia-fabric-manager-535.216.01-1.x86_64.rpm
rpm -ivh nvidia-fabric-manager-devel-535.216.01-1.x86_64.rpm
```

## 启动 nvidia-fabricmanager 服务

**依次执行以下命令，启动服务。**

```
systemctl enable nvidia-fabricmanager
```

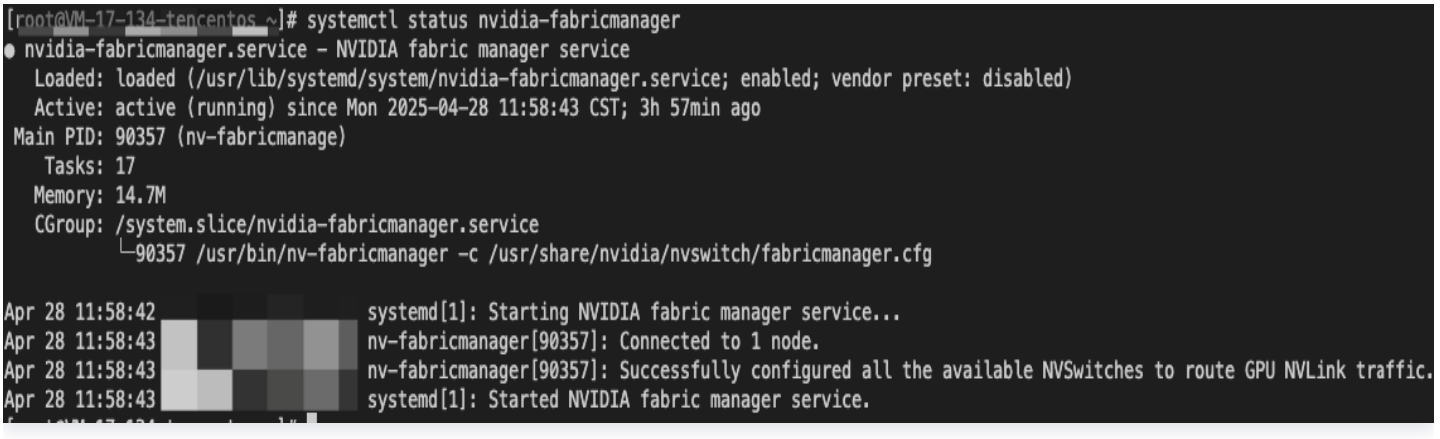
```
systemctl start nvidia-fabricmanager
```

## 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。**



# GPU 型实例安装 TCCL

最近更新时间：2024-12-16 15:43:02

## 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。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。

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

TCCL 的三种接入方案对比如下表：

| 安装方式 | （推荐）方法一：安装 NCCL 通信插件 | 方法二：编译安装 Pytorch | 方法三：安装 Pytorch 通信插件 |
| --- | --- | --- | --- |

| | | | |
|---|---|---|---|
| 适用场景 | 资源和模型训练场景相对比较固定，推荐使用该方法，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。 | • 资源需要提供给不同的业务团队，或者经常有扩容的需求，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。<br>• 不希望对业务代码做适配，只需要重新编译 Pytorch 框架。 | 资源需要提供给不同的业务团队，或者经常有扩容的需求，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。 |
| 使用步骤 | • 安装 NCCL 插件<br>• 修改启动脚本 | • 安装 TCCL<br>• 重新编译安装 Pytorch | • 安装 Pytorch 通信插件<br>• 修改分布式通信后端 |
| 优点 | 安装方便 | 对业务代码无入侵 | 安装方便 |
| 缺点 | 集群节点扩充之后，需要更新排序列表 | 需要重新编译安装 Pytorch<br>对软件环境有要求 | 需要修改业务代码<br>对软件环境有要求 |
| 软件环境依赖 | 安装 NCCL 即可 | • NCCL 版本 2.12<br>• glibc 版本 2.17 以上<br>• CUDA 版本 10.0 以上 | • Pytorch 仅支持 1.12<br>• glibc 版本 2.17 以上<br>• CUDA 版本 10.0 以上 |

## 配置 TCCL 环境并验证

### （推荐）方法一：安装 NCCL 通信插件

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 路由两种优化能力。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（例如对于 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
```

### 方法二：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，详情请参见 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_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
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. 运行 Pytorch 单机多卡或者多机多卡训练过程中有如下打印，说明安装成功：

```
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. 如果运行 nccl-tests，需要执行以下命令设置 TCCL 路径：

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

### 方法三：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。详情可参见 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_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
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. 在执行 Pytorch 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>
```

> **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。

# GPU 型实例安装 RDMA 毫秒级监控组件

最近更新时间：2025-01-06 10:56:32

## 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
```

## 配置云产品监控

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 | InstanceId | 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 | InstanceId | 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 | InstanceId | 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 | InstanceId | 10s、60s、300s、3600s、86400s |
| RxHpbwP90 | 毫秒级_RDMA 网卡接收带宽90百分位值 | 10秒内从小到大RDMA 网卡接收带宽的毫秒级统计粒度前90百分位数 | Mbps | InstanceId | 10s、60s、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 | InstanceId | 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 | InstanceId | 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 | InstanceId | 10s、60s、300s、3600s |
| TxHpbwP50 | 毫秒级_RDMA 网卡 | 10秒内从小到大RDMA 网卡发送带宽毫 | Mb | InstanceId | 10s、60s、300s、3600s |

| | | | | | |
|---|---|---|---|---|---|
| | 发送带宽50百分位 | 秒级统计粒度前50百分位数 | ps | | |
| 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 | InstanceId | 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。


## 查看本地监控

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
```

图中部分参数含义解释如下：

- Device：RDMA 网卡的名称。
- Transmitted 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。

# GPU 型实例监控和告警

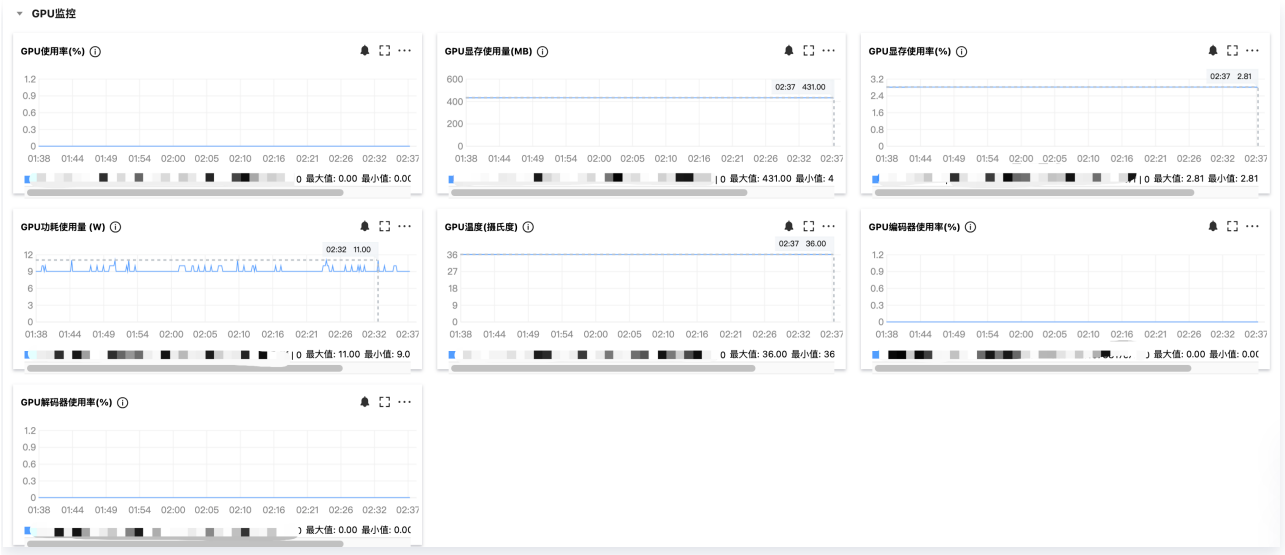
最近更新时间：2025-04-16 15:16:32

监控与告警是保证高性能计算集群 GPU 型实例高可靠性、高可用性和高性能运行的重要部分。创建实例时，默认免费开通腾讯云可观测平台。您可以通过 云服务器控制台 查看监控指标，详细说明请参见 云服务器监控内容 。
NVIDIA GPU 系列实例另外提供了监控 GPU 使用率，显存使用量，功耗以及温度等参数的能力。
您也可以在 腾讯云可观测平台 分析监控指标和实施告警，更多详细内容可参见 腾讯云可观测平台告警管理 。

> **说明：**
> 监控功能是通过在 GPU 型实例上部署安装相关 GPU 驱动 、nvidia-fabricmanager 服务 和 云服务器监控组件 来实现的，公共镜像默认包含云服务器监控组件，只需安装 GPU 驱动。如果您使用自定义镜像，需手动安装云服务器监控组件和 GPU 驱动。

## 在控制台查看 GPU 监控指标

单击 GPU 列表中的 监控图标， 访问 控制台 GPU 实例的监控页面，查看 GPU 监控，移动鼠标到指标曲线上将显示对应 GPU 设备的 BDF（Bus、Device、Function，设备唯一地址）和监控数据。如下图所示：




参数说明：

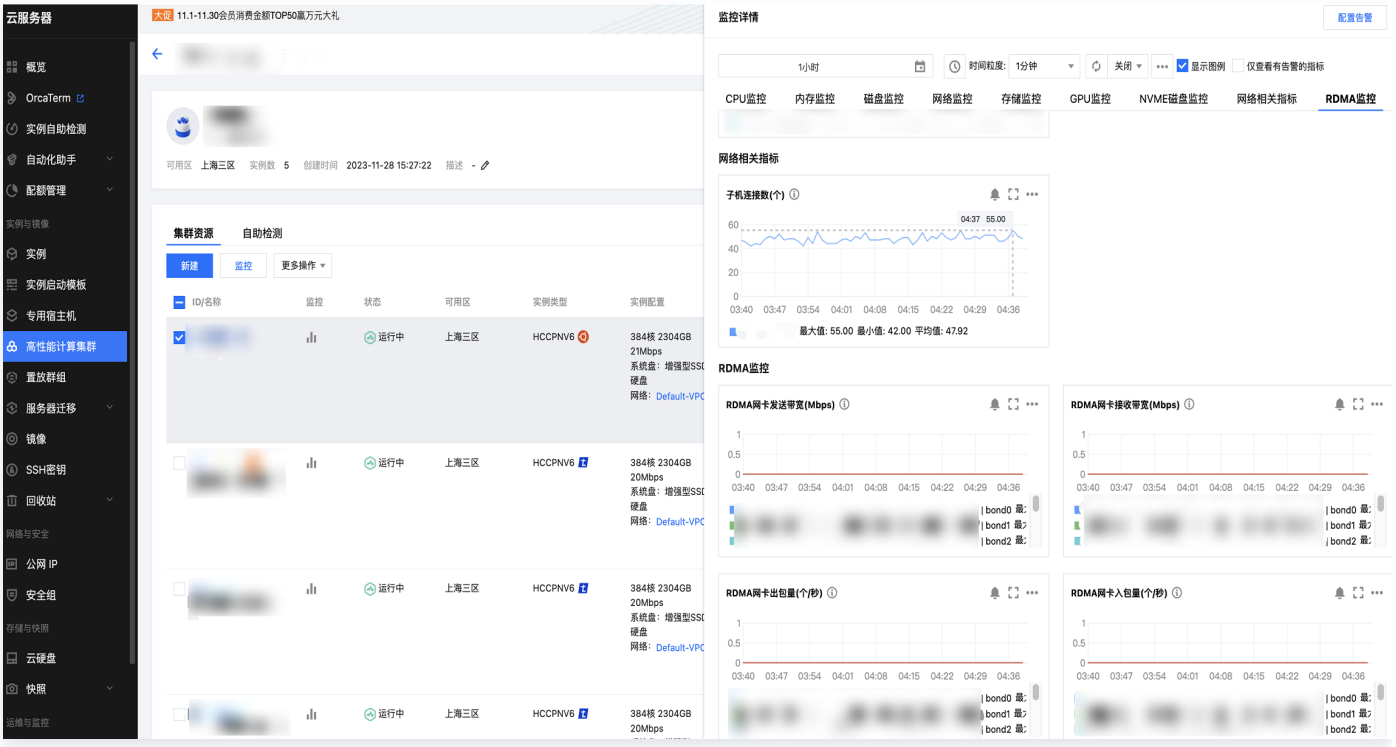
| 指标名称 | 含义 | 单位 | 维度 |
| --- | --- | --- | --- |
| GPU 使用率 | 评估负载所消耗的计算能力，非空闲状态百分比 | % | per-GPU |
| GPU 显存使用量 | 评估负载对显存占用 | MB | per-GPU |
| GPU 显存使用率 | 评估负载对显存占用百分比 | % | per-GPU |

| | | | |
|---|---|---|---|
| GPU 功耗使用量 | 评估 GPU 耗电情况 | W | per-GPU |
| GPU 温度 | 评估 GPU 散热状态 | 摄氏度 | per-GPU |
| GPU 编码器使用率 | 评估编码器使用百分比 | % | per-GPU |
| GPU 解码器使用率 | 评估解码器使用百分比 | % | per-GPU |

## 在控制台查看 RDMA 监控指标

1. 在 高性能计算集群 中选择单击**集群 ID** 或**查看详情**，可看到集群的 GPU 服务器实例详情。


2. 单击 GPU 实例 列表中的 监控图标， 访问 腾讯云控制台 GPU 实例的监控页面，查看 RDMA监控（仅支持分钟级以上粒度），移动鼠标到指标曲线上将显示实例对应 bond 的监控数据。如下图所示，当前支持RDMA网卡发送带宽，RDMA 网卡接收带宽，RDMA 网卡出包量，RDMA 网卡入包量显示。



参数说明：

| 指标中文名 | 含义 | 单位 | 维度 |
| --- | --- | --- | --- |
| RDMA 网卡接收带宽 | RDMA 网卡接收带宽 | MBit/s | InstanceId |
| RDMA 网卡发送带宽 | RDMA 网卡发送带宽 | MBit/s | InstanceId |
| RDMA 网卡入包量 | RDMA 网卡入包量 | 个/秒 | InstanceId |
| RDMA 网卡出包量 | RDMA 网卡出包量 | 个/秒 | InstanceId |

> **说明：**
> 腾讯云也提供 RDMA 网络毫秒级监控的能力，需要安装毫秒级监控组件，实现在腾讯云 RDMA 环境中毫秒级的性能监控。操作步骤可参见 GPU 型实例安装 RDMA 毫秒级监控组件 。

## 在腾讯云可观测平台查看监控指标

腾讯云可观测平台 支持分析更丰富的 GPU 监控指标。

1. 登录 腾讯云可观测平台，左侧导航栏中选择 **Dashboard**，进入 Dashboard 列表页。
2. 在 Dashboard 列表中，单击 新建 Dashboard，在新的 Dashboard 选择**新建图表**。

### 查看 GPU 监控指标

在**指标**处选择 GPU / 云服务器 / GPU 监控，单击您关注的**指标**，自定义监控面板进行多实例展示，如下图所示：

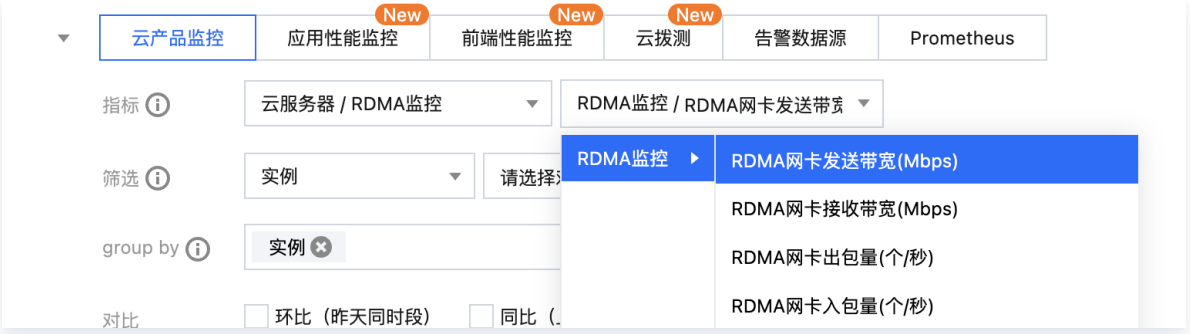

参数详情可参见 腾讯云可观测平台 GPU 云服务器监控指标，提供以下监控指标：

| 指标英文名 | 指标中文名 | 指标说明 | 单位 | 维度 |
| --- | --- | --- | --- | --- |

| | | | | |
|---|---|---|---|---|
| Gpumem usage | GPU 显存使用率 | GPU 显存使用率 | % | per-GPU |
| GpuMem Used | GPU 显存使用量 | 评估负载对显存占用 | MB | per-GPU |
| Gpupowdraw | GPU 功耗使用量 | GPU 功耗使用量 | W | per-GPU |
| Gpupowusage | GPU 功耗使用率 | GPU 功耗使用率 | % | per-GPU |
| Gputemp | GPU 温度 | 评估 GPU 散热状态 | 摄氏度 | per-GPU |
| Gpuutil | GPU 使用率 | 评估负载所消耗的计算能力，非空闲状态百分比 | % | per-GPU |
| GpuEncUtil | GPU 编码器使用率 | GPU 编码器使用率 | % | per-GPU |
| GpuDecUtil | GPU 解码器使用率 | GPU 解码器使用率 | % | per-GPU |

## 查看 RDMA 监控指标

**在指标处选择 云服务器 / RDMA 监控，单击您关注的指标，自定义监控面板进行多实例展示，如下图所示：**




**参数详情可参见 腾讯云可观测平台 RDMA 监控指标，提供以下监控指标：**

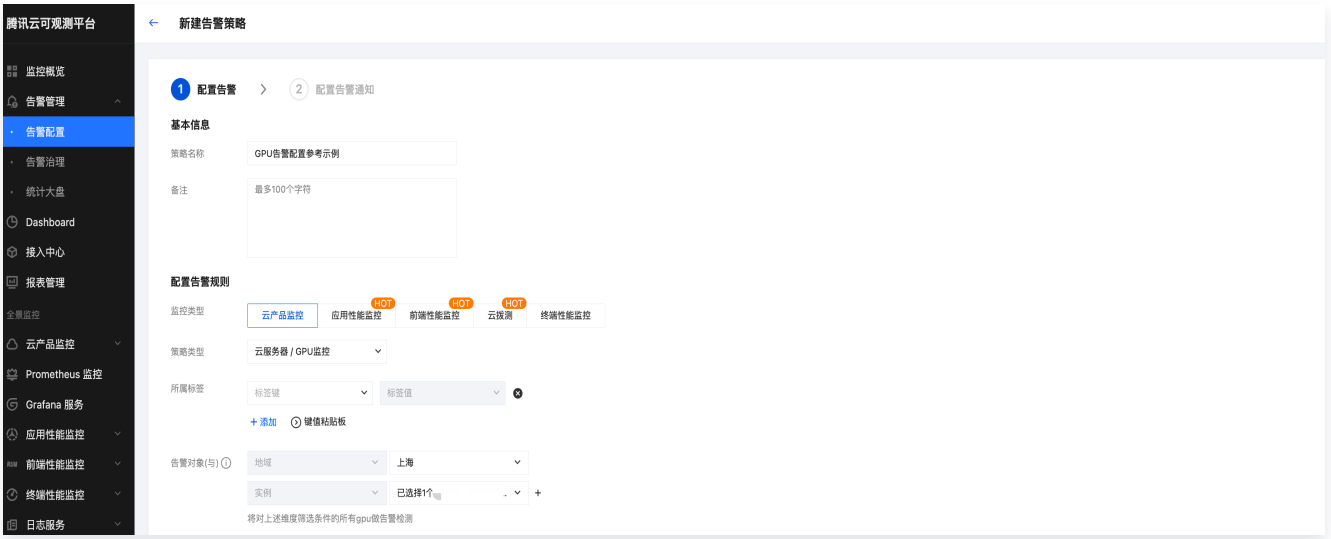
| 指标英文名 | 指标中文名 | 指标说明 | 单位 | 维度 |
|---|---|---|---|---|
| RdmaIntraffic | RDMA 网卡接收带宽 | RDMA 网卡接收带宽 | MBit/s | InstanceId |

| | | | | |
|---|---|---|---|---|
| RdmaOuttraffic | RDMA 网卡发送带宽 | RDMA 网卡发送带宽 | MBit/s | InstanceId |
| RdmaInpkt | RDMA 网卡入包量 | RDMA 网卡入包量 | 个/秒 | InstanceId |
| RdmaOutpkt | RDMA 网卡出包量 | RDMA 网卡出包量 | 个/秒 | InstanceId |
| CnpCount | CNP 统计量 | 拥塞通知报文统计 | 个/秒 | InstanceId |
| EcnCount | ECN 统计量 | 显示拥塞通知统计 | 个/秒 | InstanceId |
| RdmaPktDiscard | 端测丢包量 | 端测丢包量 | 个/秒 | InstanceId |
| RdmaOutOfSequence | 接收方乱序错误量 | 接收方乱序错误量 | 个/秒 | InstanceId |
| RdmaTimeoutCount | 发送方超时错误量 | 发送方超时错误量 | 个/秒 | InstanceId |
| TxPfcCount | TX PFC 统计量 | TX PFC 统计量 | 个/秒 | InstanceId |
| RxPfcCount | RX PFC 统计量 | RX PFC 统计量 | 个/秒 | InstanceId |

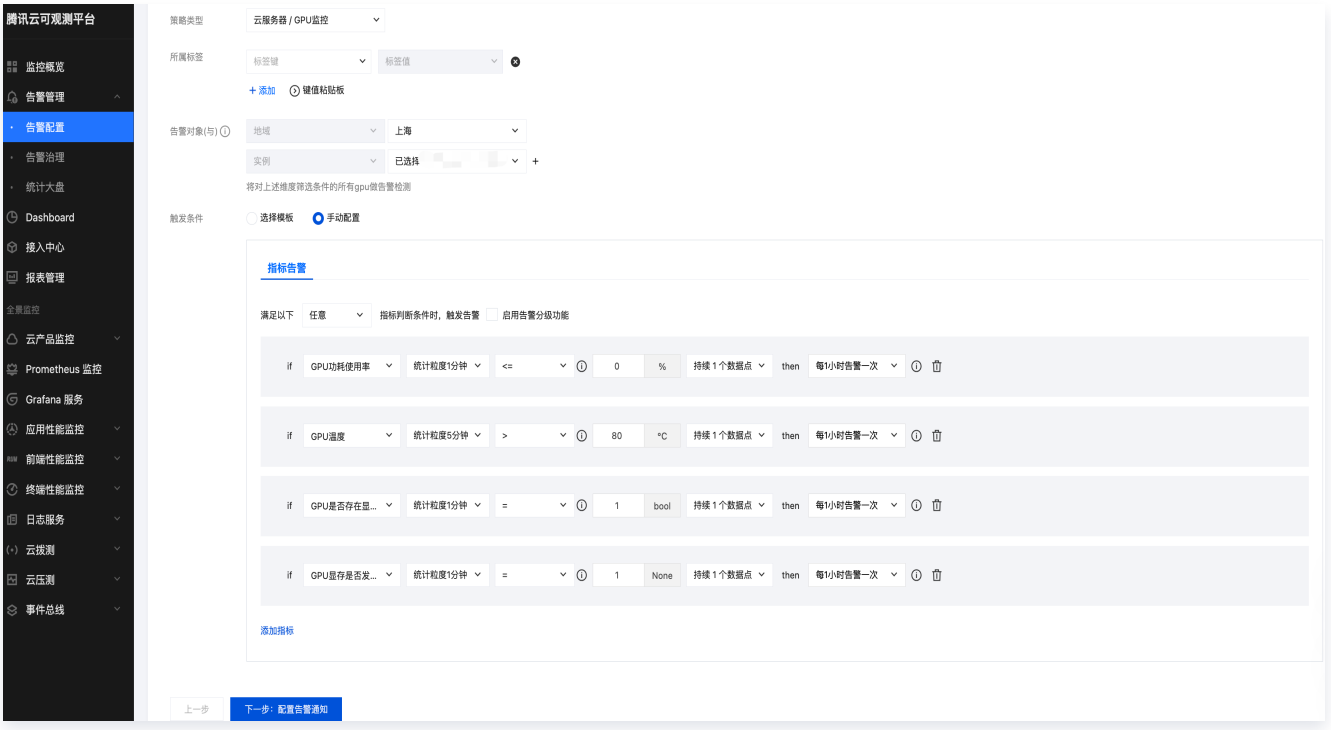
## 监控指标告警配置

### 配置 GPU 监控指标告警

1. 登录 腾讯云可观测平台，在左侧导航栏中，选择**告警管理 > 告警配置**。
2. 单击 新建告警策略，在监控类型选择**云产品监控**，策略类型中选择**云服务器 / GPU 监控**，选择您希望接收告警的 GPU 实例对象，触发条件选择**手动配置**。




3. GPU 云服务器监控支持以下指标告警：GPU 内存使用率、GPU 功耗使用率、GPU 使用率、GPU 温度、GPU 是否存在显存页需隔离、GPU 显存是否发生 UCE 等。您可以参考下图进行配置告警。告警通知的配置可参见 新建通知模板，支持通过多种渠道进行通知。



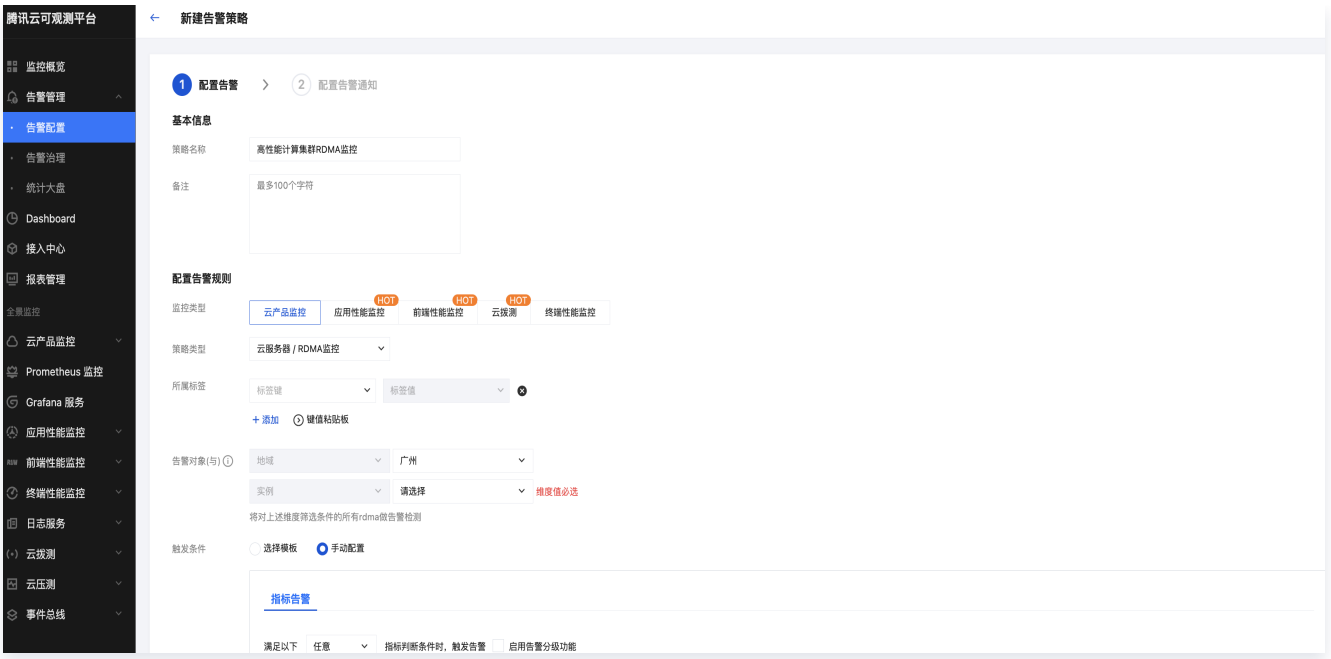

常用告警指标参考如下：

| 指标名称 | 建议告警阈值 | 描述 | 处理建议 |
| --- | --- | --- | --- |

| | | | |
|---|---|---|---|
| GPU 功耗使用率 | <=0 | 功耗小于0时可能功率出现 Unknown Error 了，会影响 GPU 的正常使用。 | 执行 `nvidia-smi` 命令查看 GPU 的功率是否有 ERR 或 nvidia-smi -i <target gpu> -q \|grep "Power Draw" 是否为 Unknown Error，若存在该现象则尝试重启机器恢复及更新驱动观察。若重启无法恢复 提交工单 联系腾讯云支持。 |
| GPU 温度 | 持续5分钟>80 | 当 GPU 温度过高时可能会导致 GPU SlowDown，影响业务性能。 | 可能负载过高导致 GPU 温度过高，可尝试重启实例恢复，若无法恢复 提交工单 联系腾讯云支持。 |
| GPU 是否存在显存页需隔离 | =1 | 安培以下架构 GPU 出现了 ECC ERROR，应用进程被 kill，GPU卡处于 pending 状态。 | 执行 `nvidia-smi -i <target gpu> -q -d PAGE_RETIREMENT` 命令查看是否有 GPU 卡处于 pending 状态，重置 GPU 卡或重启实例恢复。若重启无法恢复 提交工单 联系腾讯云支持。 |
| GPU 显存是否发生 UCE | =1 | 安培及以上架构 GPU 出现了 ECC ERROR，应用进程被 kill，GPU卡处于 pending 状态。 | 执行 `nvidia-smi -i <target gpu> -q -d ROW_REMAPPER` 命令查看是否有 GPU 卡处于 Pending 状态，重置 GPU 卡或重启实例恢复。若重启无法恢复 提交工单 联系腾讯云支持。 |
| GPU 内存使用率 | 仅保持观察 | - | 评估负载对显存占用。 |
| GPU 使用率 | 仅保持观察 | - | 评估负载对 GPU 流处理器占用。 |

## 配置 RDMA 监控指标告警

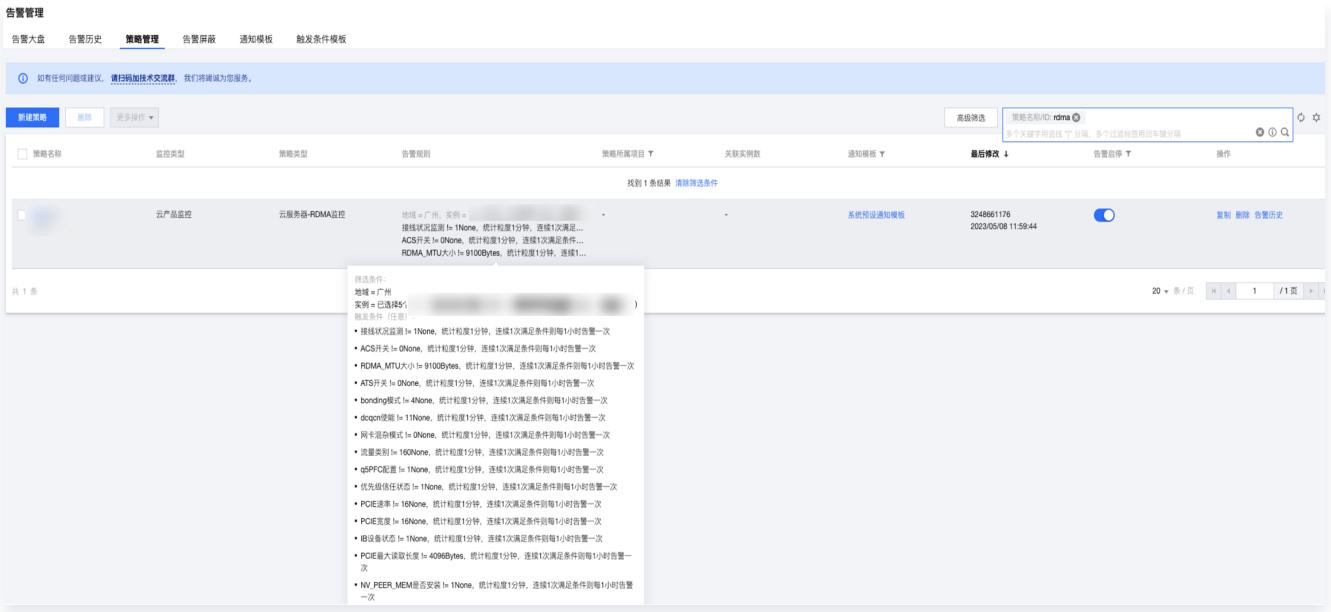
1. 登录 腾讯云可观测平台，左侧导航栏中选择**告警配置**，新建告警策略，监控类型选择**云产品监控**，策略类型选择云服务器 /RDMA 监控，选择告警对象。


腾讯云可观测平台
新建告警策略
监控概览
告警管理
告警配置
告警治理
统计大盘
Dashboard
接入中心
报表管理
全景监控
云产品监控
Prometheus 监控
Grafana 服务
应用性能监控
前端性能监控
终端性能监控
日志服务
云拨测
云压测
事件总线
1 配置告警
2 配置告警通知
基本信息
策略名称
高性能计算集群RDMA监控
备注
最多100个字符
配置告警规则
监控类型
云产品监控
应用性能监控
前端性能监控
云拨测
终端性能监控
HOT
策略类型
云服务器 / RDMA监控
所属标签
标签键
标签值
+ 添加
键值粘贴板
告警对象(与)
地域
广州
实例
请选择
维度值必选
将对上述维度筛选条件的所有rdma做告警检测
触发条件
选择模板
手动配置
指标告警
满足以下
任意
指标判断条件时，触发告警
启用告警分级功能

**指标告警参考如下配置：**


2. 告警通知可参见 新建通知模板 配置，支持多渠道通知。

配置完成后策略查看截图如下：




告警示例截图如下：


高性能计算集群 GPU 型实例 RDMA 告警处理建议如下：

| 监测指标 | 指标名 | 错误描述 | 正确配置值 | 处理策略 | 客户更正方式 | 设备 |
| --- | --- | --- | --- | --- | --- | --- |
| 接线状况监测 | link_detected | 链路down | 1（1代表端口up） | 客户尝试软件恢复，如无法恢复，授权运维维修 | ifconfig $ethname up | eth |
| ACS开关 | acs | ACS开关配置错误 | 0（0代表关闭ACS） | 客户改正配置（需要重启机器） | bash /etc/acsctl_online.sh disable_acsctl | eth |
| RDMA_MTU大小 | active_mtu | RDMA卡的MTU配置错误（影响性能） | 9100 | 客户更正配置即可 | ifconfig $ethname mtu 9100 | bond |

| | | | | | | |
|---|---|---|---|---|---|---|
| ATS 开关 | ats_enabled | ATS 开关配置错误 | 0（0代表关闭 ATS） | 客户改正配置（需要重启机器） | // 关闭 ATS<br>for i in `lspci -d 15b3: \| awk '{print $1}'`; do echo $i; mlxconfig -d $i -y s ATS_ENABLED=0; done<br><br>// 重启之后确认状态<br>for i in `lspci -d 15b3: \| awk '{print $1}'`; do echo $i; mlxconfig -d $i q \| grep ATS_ENABLED; done | eth |
| bonding 模式 | bonding_mode | bonding 模式配置错误 | 4（4代表双发模式） | 客户更正配置即可 | cd /usr/local/qcloud/rdma/; sh set_bonding.sh; sh dscp.sh | bond |
| dcqcn 使能 | dcqcn_enable | dcqcn 未使能 | 11（两个1分别代表 rp 和 np 的状态） | 客户更正配置即可 | echo 1 > /sys/class/net/$ethname/ecn/roce_rp/enable/5<br>echo 1 > /sys/class/net/$ethname/ecn/roce_np/enable/5 | eth |
| 网卡混杂模式 | eth_promisc | 网卡误配为混杂模式 | 0（0代表非混杂模式） | 客户更正配置即可 | ifconfig $ethname -promisc | eth |
| 流量类别 | traffic_class | 流量类别配置错误 | 160 | 客户更正配置即可 | echo 160 > /sys/class/infiniband/$RDMA_name/tc/1/traffic_class | bond |
| q5 PFC 配置 | q5_pfc_enabled | PFC 未使能，存在 QOS | 1（1代表 PFC 使能） | 客户更正配置即可 | mlnx_qos -i $ethname -f 0,0,0,0,0,1,0,0 | eth |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | ERROR | | | | |
| 优先级信任状态 | prio_trust_state | 优先级信任状态配置错误 | 1（1代表dscp） | 客户更正配置即可 | mlnx_qos -i $ethname --trust=dscp | eth |
| pcie速率 | max_link_speed | PCIE GEN配置错误 | 16 | 客户更正配置即可 | 尝试重启实例恢复，若无法恢复提交工单腾讯云支持 | eth |
| pcie宽度 | max_link_width | PCIE width配置错误 | 16 | 客户更正配置即可 | 尝试重启实例恢复，若无法恢复提交工单腾讯云支持 | eth |
| IB设备状态 | link_state | bond口下两个eth口全部down | 1（1代表bond口up） | 客户尝试软件恢复，如无法恢复，授权运维维修 | ifconfig $ethname up | bond |
| MRSS PCIE最大读取长度 | mrss | MRSS配置错误 | 4096 | 客户更正配置即可 | lspci -D -nn \| grep 15b3 \|awk -F' ' '{print $1}' \|xargs -I {} setpci -s {} 68.w=5936 | eth |
| NV_MEM_PEER是否安装 | nv_peer_mem_state | nvidia_peermem模块未加载 | 1（1代表模块已加载） | 客户加载模块即可 | modprobe nvidia_peermem | 整机 |