

# Huawei Cloud EulerOS ( HCE )

## 常见问题

文档版本 01  
发布日期 2025-12-29



版权所有 © 华为云计算技术有限公司 2025。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

## 商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

## 注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

## 华为云计算技术有限公司

地址：贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址：<https://www.huaweicloud.com/>

## 目录

1 CentOS Linux 停止维护后如何应对? .....	1
2 华为云针对 CentOS EOL 有没有迁移方案? .....	4
3 如何安装 mlnx 驱动? .....	6
4 如何开启 HCE 的 SELinux 功能? .....	9
5 如何关闭 HCE 的 SELinux 功能? .....	11
6 迁移系统后, 如何更改控制台操作系统名称? .....	12
7 HCE、openEuler 和 EulerOS 镜像的主要区别是什么? .....	15
8 如何打开内核 wireguard 模块以及安装 wireguard-tools? .....	16
9 如何将 docker 工具的用户凭证保存方式配置成与社区一致? .....	18
10 free 和 available 说明.....	19
11 系统内存回收机制说明.....	20
12 os 内存预留说明.....	22
13 OOM 相关参数配置与原因排查.....	23
14 IPVS 报错问题说明.....	28
15 中文环境执行 sulogin 命令终端显示乱码说明.....	30
16 ECS 开启 IPv6 后, HCE 系统内无法获取到 IPv6 地址.....	31
17 如何设置自动注销时间 TMOUT? .....	33
18 panic_on_oom 系统参数变更说明.....	35
19 如何添加缺少的字符集.....	36
20 如何解决证书切换导致的安全启动失败问题.....	38
21 如何进行网卡中断绑核.....	40
22 x86_64 环境如何调整 memcpy 默认水位线? .....	42
23 网络服务重启导致 resolv.conf 配置内容发生变更.....	43
24 如何设置 nohup 后台进程在 VNC 会话退出后不被杀死.....	44

**25 系统配置 XPS 的方法及影响说明..... 46**

**26 系统异常重启的原因排查..... 48**

**27 操作系统控制台常见问题..... 50**

27.1 使用操作系统控制台收费吗? .....50

27.2 为什么有些 ECS 虚拟机不支持添加? .....50

27.3 系统常驻组件的资源消耗高吗? ..... 50

27.4 如何检查节点是否支持 eBPF 功能? .....50

27.5 节点插件状态离线问题排查..... 51

**28 IMA 度量机制占用内存过大问题说明.....56**

**29 如何解决 curl 命令报 out of memory? .....57**

# 1 CentOS Linux 停止维护后如何应对?

CentOS官方已计划停止维护CentOS操作系统，华为云上CentOS公共镜像来源于CentOS官方，当CentOS操作系统停止维护后，华为云将会同时停止对该操作系统的支持。本文主要介绍CentOS操作系统停止维护带来的影响，并针对影响提供应对策略。

## 背景信息

2020年12月08日，CentOS官方宣布了停止维护CentOS Linux的计划，并推出了CentOS Stream项目。更多信息，请参见[CentOS官方公告](#)。

CentOS 8系统2021年12月31日已停止维护服务，CentOS 7系统将于2024年06月30日停止维护服务。CentOS官方不再提供CentOS 9及后续版本，不再支持新的软件和补丁更新。CentOS用户现有业务随时面临宕机和安全风险，并无法确保及时恢复。

## 影响

基于CentOS官方的变更计划，对CentOS操作系统的使用者产生的影响如下所述：

- 2021年12月31日以后，CentOS 8的使用者将无法获得包括问题修复和功能更新在内的任何软件维护和支持。
- 2024年06月30日以后，CentOS 7的使用者将无法获得包括问题修复和功能更新在内的任何软件维护和支持。

对于华为云的公共镜像及服务支持存在一定影响：

- 华为云暂不会下线CentOS 8公共镜像，同时已经使用CentOS 8创建的ECS实例运行不会受到影响，但将停止更新镜像。
- 华为云对于CentOS操作系统的服务支持将和CentOS官方日期保持同步。2021年12月31日以后将不再对CentOS 8提供服务支持；对CentOS 7的服务支持将持续至2024年6月30日。

## 应对策略

为了保障使用CentOS系统的业务正常运行，华为云为您提供替换CentOS操作系统的应对策略。替换CentOS操作系统的方式分为两类，切换操作系统和迁移操作系统。

- 将CentOS操作系统切换为[支持切换的操作系统](#)。

如果现有的ECS配置（网卡、磁盘、VPN等配置的类型和数量）都不需要改变，仅需要修改ECS的操作系统镜像，并且您的软件和原操作系统耦合度较低，建议使用系统切换。

- 切换到Huawei Cloud EulerOS具体操作，详见[将操作系统切换为HCE](#)。
- 切换到CentOS Stream或Rocky Linux具体操作详见[切换操作系统](#)。
- 将CentOS操作系统迁移为Huawei Cloud EulerOS操作系统。

如果现有的ECS配置（网卡、磁盘、VPN等配置的类型和数量）都不需要改变，希望保留操作系统软件的配置参数，可以通过操作系统迁移的方式迁移到Huawei Cloud EulerOS。

系统迁移详见[将操作系统迁移为HCE](#)。

系统切换和迁移的区别如下表，请根据需要选择合适的替换方式。

表 1-1 系统切换和迁移的区别

区别	系统切换	系统迁移
数据备份	<ul style="list-style-type: none"><li>• 切换操作系统会清除系统盘数据，包括系统盘上的系统分区和所有其它分区。</li><li>• 切换操作系统不影响数据盘数据。</li></ul>	<ul style="list-style-type: none"><li>• 迁移操作系统不会清除系统盘数据，为避免系统软件的数据丢失，建议将其备份。</li><li>• 迁移操作系统不影响数据盘数据。</li></ul>
个性化设置	切换操作系统后，当前操作系统内的个性化设置（如DNS、主机名等）将被重置，需重新配置。	迁移操作系统后，当前操作系统内的个性化设置（如DNS、主机名等）不需重新配置。

表 1-2 支持切换的操作系统

操作系统	概述	适用人群
Huawei Cloud EulerOS	Huawei Cloud EulerOS(简称HCE)是基于openEuler构建的云上操作系统，中文名为华为云欧拉操作系统。HCE打造云原生、高性能、高安全、易迁移等能力，加速用户业务上云，提升用户的应用创新空间，可替代CentOS、EulerOS等公共镜像。	适用于希望使用免费镜像，并延续开源社区镜像使用习惯的个人或企业。
CentOS Stream	CentOS Stream是一个滚动升级的版本，由CentOS官方提供。	适用于希望延续CentOS使用习惯，并希望获得滚动升级的个人或企业。
Rocky Linux	Rocky Linux是一个社区化的企业级操作系统，位于Red Hat Enterprise Linux（RHEL）下游。Rocky Linux与CentOS一样，提供了适用于服务器的稳定版本，旨在作为CentOS完全兼容的替代版本。	适用于希望使用免费镜像，并延续开源社区镜像使用习惯的个人或企业。

操作系统	概述	适用人群
AlmaLinux	AlmaLinux是CloudLinux团队宣布构建的一个稳定版CentOS社区分支。该操作系统实现了与Red Hat Enterprise Linux ( RHEL ) 二进制文件的1:1兼容，并提供了不停机更换操作系统的功能。	适用于希望使用免费镜像，并延续开源社区镜像使用习惯的个人或企业。
Debian、Ubuntu操作系统	Linux的其他发行版操作系统，不同操作系统在使用习惯和应用兼容性上存在一定差异。	适用于可以自行应对操作系统切换成本的个人或企业。

# 2 华为云针对 CentOS EOL 有没有迁移方案?

## 背景信息

CentOS 8系统2021年12月31日已停止维护服务，CentOS 7系统将于2024年06月30日停止维护服务。CentOS不再支持新的软件和补丁更新。CentOS用户现有业务随时面临宕机和安全风险，并无法确保及时恢复。

HCE操作系统从云原生混部竞争力、安全可信、快速迁移、高效运维、专业认证等方面为用户提供专业云服务、解决CentOS停服带来的影响。HCE提供了迁移工具，可将CentOS、EulerOS等操作系统平滑迁移至HCE操作系统。

## 兼容性评估

华为HCE操作系统已具备完整代替CentOS的技术能力，完全自主可控，并基于openEuler开源社区持续自主演进。南向支持6大类400种板卡，基本覆盖主流计算产品。北向100%兼容主流的应用场景（云原生、存储、数据库、大数据、WEB等）。超过5000种应用软件通过兼容性认证，基本能够替代CentOS的各种部署。

为满足CentOS系列到HCE搬迁的准确性和安全性，请您使用兼容性工具对待迁移软件快速进行扫描，获取评估结果。

对于可兼容的应用软件，迁移过程中并不会修改软件配置，迁移完成后无需重新配置；对于部分不兼容的应用软件，评估报告给出相应的规避策略，请在迁移之后进行相应的适配。

## 迁移能力评估

HCE已有成熟的搬迁指导，按照分布式集群应用、主备应用、单机应用三种类型对各种应用进行归类，并制定相应的搬迁方案：

- 针对分布式集群软件，如大数据、分布式存储。CentOS搬迁无需中断业务，基于分布式软件伸缩扩容机制，HCE实现滚动代替，平滑搬迁。
- 针对主备应用，如数据库等。CentOS搬迁无需中断业务，先备后主，基于主备状态同步机制，平滑搬迁。
- 针对单机应用，CentOS搬迁需中断业务，割接式搬迁，该类搬迁方案和原应用重新部署方式等同，属于成熟方案。

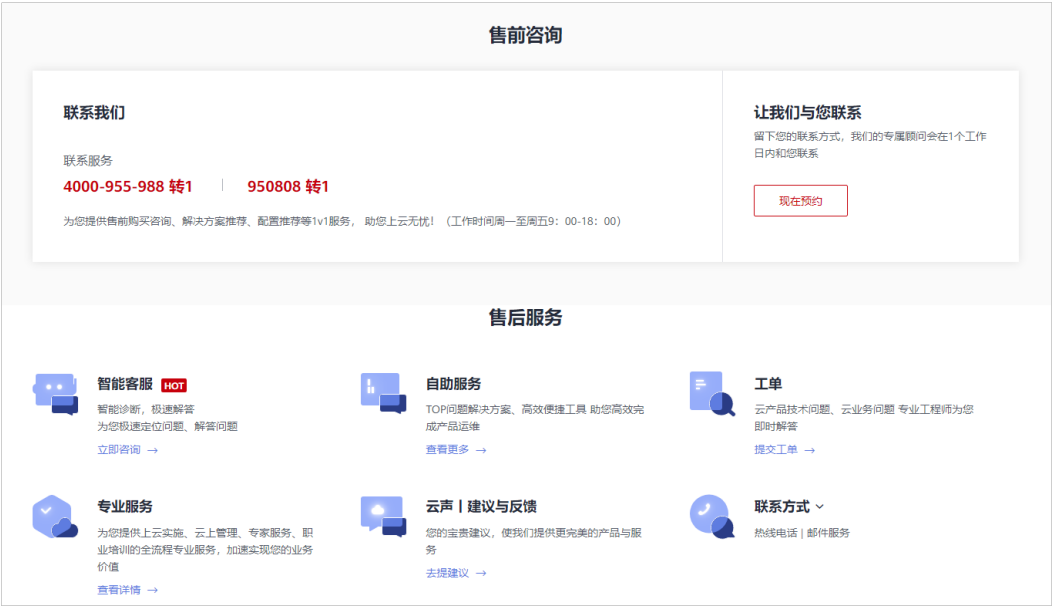
华为云提供[两步切换至HCE操作系统](#)实践，本实践带您体验从CentOS迁移到Huawei Cloud EulerOS的完整过程，开发者可直接体验兼容评估和迁移能力评估。



联系我们

华为云专业的服务团队，致力于为您提供专业的售前购买、咨询服务，及完善的售后技术服务，欢迎[联系我们](#)。

图 2-1 多种技术服务途径



# 3 如何安装 mlnx 驱动?

本节介绍如何在HCE操作系统（包含x86架构和Arm架构系统）安装mlnx驱动。

## 约束与限制

- 只支持在HCE 2.0安装使用。
- CX6网卡驱动为23.10-1.1.9.0-LTS及以上版本。

## 前提条件

已经安装5.10或更高内核版本的HCE系统。

## x86 架构安装 mlnx 驱动

1. 下载CX6网卡驱动安装包[MLNX\\_OFED\\_LINUX-23.10-1.1.9.0-openeuler22.03-x86\\_64.tgz](#)。
2. 解压驱动安装包并进入工作目录。  

```
tar -xf MLNX_OFED_LINUX-23.10-1.1.9.0-openeuler22.03-x86_64.tgz  
cd MLNX_OFED_LINUX-23.10-1.1.9.0-openeuler22.03-x86_64
```
3. 安装CX6网卡驱动软件。  

```
./mlnxofedinstall --basic --without-depcheck --distro OPENEULER22.03 --  
force --kernel 5.10.0-60.18.0.50.oe2203.x86_64 --kernel-sources /lib/  
modules/$(uname -r)/build
```

### 说明

其中，“5.10.0-60.18.0.50.oe2203.x86\_64”是官方MLNX\_OFED包本身编译时的内核版本。

4. 创建链接。  

```
ln -s /lib/modules/5.10.0-60.18.0.50.oe2203.x86_64/extra/mlnx-  
ofa_kernel /lib/modules/$(uname -r)/weak-updates/  
ln -s /lib/modules/5.10.0-60.18.0.50.oe2203.x86_64/extra/kernel-mft /lib/  
modules/$(uname -r)/weak-updates/  
depmod -a
```
5. 执行reboot命令重启系统。

6. 执行`/etc/init.d/openibd status`命令查看驱动安装结果。  
显示如图3-1表示驱动安装成功。

图 3-1 驱动安装成功

```
[root@lnmp ~]# /etc/init.d/openibd status  
  
HCA driver loaded  
  
The following OFED modules are loaded:  
  
rdma_ucm  
rdma_cm  
ib_ipoib  
mlx5_core  
mlx5_ib  
ib_uverbs  
ib_umad  
ib_cm  
ib_core  
mlxfw  
  
[root@lnmp ~]#
```

## Arm 架构安装 mlnx 驱动

1. 下载CX6网卡驱动安装包[MLNX\\_OFED\\_LINUX-23.10-1.1.9.0-openeuler22.03-aarch64.tgz](#)。
2. 解压驱动安装包并进入工作目录。  
`tar -xf MLNX_OFED_LINUX-23.10-1.1.9.0-openeuler22.03-aarch64.tgz`  
`cd MLNX_OFED_LINUX-23.10-1.1.9.0-openeuler22.03-aarch64`
3. 安装CX6网卡驱动软件。  
`./mlnxfedinstall --basic --without-depcheck --distro OPENEULER22.03 --force --kernel 5.10.0-60.18.0.50.oe2203.aarch64 --kernel-sources /lib/modules/$(uname -r)/build`

### 说明

其中，“5.10.0-60.18.0.50.oe2203.aarch64”是官方MLNX\_OFED包本身编译时的内核版本。

4. 执行如下命令创建链接。  
`ln -s /lib/modules/5.10.0-60.18.0.50.oe2203.aarch64/extra/mlnx-ofa_kernel /lib/modules/$(uname -r)/weak-updates/`  
`ln -s /lib/modules/5.10.0-60.18.0.50.oe2203.aarch64/extra/kernel-mft /lib/modules/$(uname -r)/weak-updates/`  
`depmod -a`
5. 执行`reboot`命令重启系统。
6. 执行`/etc/init.d/openibd status`命令查看驱动安装结果。  
显示如图3-2表示驱动安装成功。

图 3-2 驱动安装成功

```
[root@lnmp ~]# /etc/init.d/openibd status  
  
HCA driver loaded  
  
The following OFED modules are loaded:  
  
rdma_ucm  
rdma_cm  
ib_ipoib  
mlx5_core  
mlx5_ib  
ib_uverbs  
ib_umad  
ib_cm  
ib_core  
mlxfw  
  
[root@lnmp ~]#
```

# 4 如何开启 HCE 的 SELinux 功能?

HCE默认关闭SELinux功能。如果业务需要开启SELinux 功能，请参照本节指导操作。

## ⚠ 注意

请按照本节指导开启SELinux功能，勿直接通过/etc/selinux/config开启SELinux功能，否则可能会出现无法登录的问题。

## 操作步骤

### 1. 修改配置文件：

对于EFI启动的机器：修改/boot/efi/EFI/hce/grub.cfg文件，删除selinux=0。

对于BIOS启动的机器：修改/boot/grub2/grub.cfg文件，删除selinux=0。

### 📖 说明

如果selinux=0不存在，则忽略。

### 2. 执行touch /.autorelabel命令。

/.autorelabel文件将触发OS在启动过程中对磁盘上所有文件relabel重新打SELinux标签，该过程可能需要持续几分钟。relabel完成后OS将自动重启一次并生效，同时自动删除/.autorelabel文件确保下次不会再重复执行relabel动作。

### 3. 打开配置文件/etc/selinux/config，设置SELINUX=permissive，并执行reboot重启操作系统。

图 4-1 设置 SELINUX=permissive

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of three values:
#   targeted - Targeted processes are protected.
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

### 4. 再次打开配置文件/etc/selinux/config，设置SELINUX=enforcing，并执行reboot重启操作系统。

图 4-2 设置 SELINUX=enforcing

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

5. 重启后执行**getenforce**命令查看SELinux状态。  
显示Enforcing表示SELinux已经开启。

图 4-3 查看 SELinux 状态

```
[root@localhost ~]#
[root@localhost ~]# getenforce
Enforcing
```

# 5 如何关闭 HCE 的 SELinux 功能?

1. 执行**getenforce**查看SELinux状态，显示Enforcing表示SELinux已经开启。

图 5-1 SELinux 已开启示例

```
[root@localhost ~]# getenforce
Enforcing
[root@localhost ~]#
```

2. 打开配置文件/etc/selinux/config，设置SELINUX=disabled，保存退出。然后执行**reboot**重启操作系统。

图 5-2 设置 SELINUX 为 disabled 示例

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. 重启后执行**getenforce**命令查看SELinux状态，显示Disabled表示SELinux已经关闭。

图 5-3 SELinux 已关闭示例

```
[root@localhost ~]# getenforce
Disabled
[root@localhost ~]#
```

# 6 迁移系统后，如何更改控制台操作系统名称？

## 问题背景

原操作系统（例如CentOS 7.9）迁移为Huawei Cloud EulerOS后，控制台仍然显示原操作系统名称CentOS 7.9而不是Huawei Cloud EulerOS。

您可通过创建私有镜像、再切换到此私有镜像的方式，将控制台操作系统名称更改为Huawei Cloud EulerOS。

图 6-1 操作系统更名



## 操作步骤

1. 登录[ECS控制台](#)。
2. 在待迁移系统的弹性云服务器的“操作”列下，选择“更多 > 镜像 > 创建镜像”。
3. 在“创建私有镜像”页面，配置如下镜像信息。
  - 区域：服务器所在区域，请保持此默认配置。
  - 创建方式：创建私有镜像，请保持此默认配置。
  - 镜像类型：系统盘镜像，请保持此默认配置。
  - 镜像源：迁移系统的弹性云服务器，请保持此默认配置。
  - 名称：填写便于识别的镜像名称。
  - 协议：阅读并勾选协议。



图 6-2 配置镜像信息

目前镜像服务已进入商业化阶段，私有镜像会收取一定的存储费用。详细计费标准可参考[镜像服务计费标准](#)

镜像类型和来源

\* 区域

华东-上海一

\* 创建方式

创建私有镜像

导入私有镜像

\* 镜像类型

系统盘镜像

整机镜像

数据盘镜像

\* 选择镜像源

云服务器

裸金属服务器

不同区域的云服务产品之间内网互不相通；请就近选择靠近您业务的区域，可减少网络时延，提高访问速度。

当前关机或开机状态的弹性云服务器才可以用来自建私有镜像。

创建镜像前，请确保弹性云服务器已完成相关配置。[了解更多](#)

请勿在创建镜像过程中对所选择的弹性云服务器及其关联资源进行其他操作。

所有状态

ID

fc62cfee-7f39-49de-985 X

Q

C

名称	操作系统	运行状态	私有IP地址	创建时间
<div>Servers-to-be-migrated</div>	CentOS 7.9 64bit	运行中		2023/09/14 11:16:25 GM...

当前选择：Servers-to-be-migrated | 操作系统：CentOS 7.9 64bit | 系统盘：通用型SSD | 40 GiB

购买弹性云服务器

配置信息

加密

未加密

\* 名称

\* 企业项目

--请选择--

C

标签

如果您需要使用同一标签标识多种云资源，即所有服务均可在标签输入框下拉选择同一标签，建议在TMS中创建预定义标签。[查看预定义标签](#)

标签键

标签值

您还可以添加10个标签。

描述

0/1,024

协议

☐ 我已经阅读并同意《[镜像制作承诺书](#)》和《[镜像免责声明](#)》

立即创建

4. 单击“立即创建”创建私有镜像。

5. 确认镜像信息，单击“提交”。

6. 返回弹性云服务器控制台，在待切换操作系统的弹性云服务器的“操作”列下，选择“更多 > 镜像 > 切换操作系统”。

7. 在“切换操作系统”界面配置如下参数。

勾选立即关机。

镜像：选择私有镜像。

登录凭证：选择“使用镜像密码”。

文档版本 01 (2025-12-29)

版权所有 © 华为云计算技术有限公司

13

图 6-3 配置参数

切换操作系统

切换操作系统提供以用户选择的镜像进行重装系统的功能。

1、切换操作系统不影响数据盘数据，但是系统盘的所有分区数据和创建的所有快照会被删除，请做好数据备份。

2、部分操作系统不支持挂载SCSI磁盘，切换操作系统后，可能会导致原弹性云服务器上挂载的SCSI磁盘不可用。查看支持列表

3、切换操作系统成功后，云服务器会自动开机；当前操作系统内的个性化设置（如DNS、主机名等）将被重置，需重新配置。

收起说明

当前配置

云服务器名称	IP地址	规格	镜像	系统盘
Servers-to-be-migrated	<div><div></div><div></div><div></div><div></div><div></div></div> (私有)	1vCPUs   1GiB  ...	CentOS 7.9 64bit (64-bit)	40GiB

☐ 立即关机 (切换操作系统前需先将云服务器关机)

镜像

公共镜像

私有镜像

共享镜像

市场镜像

Huawei Cloud EulerOS(40GiB)

新建私有镜像

☐ 磁盘加密 

?

登录凭证

密码

密钥对

使用镜像密码

创建后设置

保留所选镜像的密码。为了保证您的正常使用，请确保所选镜像中已经设置了密码。

确定

取消

8. 单击“确定”并根据界面提示完成验证。

9. 阅读并勾选声明，单击“确定”。  
系统切换后，控制台系统名称即更改为Huawei Cloud EulerOS。

文档版本 01 (2025-12-29)

版权所有 © 华为云计算技术有限公司

14

7

HCE、openEuler 和 EulerOS 镜像的主要区别是什么?

HCE、openEuler和EulerOS镜像均为华为自研镜像，主要区别与联系如下[表7-1](#)所示：

表 7-1 HCE、openEuler 和 EulerOS 镜像的区别与联系

镜像类型	描述
HCE	HCE是基于openEuler开发的一款商业发行版镜像，可替代CentOS、EulerOS等操作系统，并提供专业的维护保障能力，镜像目前免费对用户使用。
openEuler	openEuler是一款开源镜像，您可以免费使用，但是不提供商业维护保障能力。openEuler最初由华为研发，但是在2021年11月9日正式捐赠给开放原子开源基金会，openEuler的技术支持由开源社区提供。
EulerOS	EulerOS是基于开源技术的企业级Linux操作系统软件，具备高安全性、高可扩展性、高性能等技术特性，能够满足客户IT基础设施和云计算服务等多业务场景需求。

# 8 如何打开内核 wireguard 模块以及安装 wireguard-tools?

## 说明

wireguard-tools工具来源于社区，如果您在使用中遇到问题，可通过<https://github.com/WireGuard/wireguard-tools/pulls>获取帮助。

## 打开内核 wireguard 模块

您通过命令**modprobe wireguard**打开内核wireguard模块。

## 安装 wireguard-tools

**步骤1** 执行以下命令安装依赖。

```
dnf install gcc make
```

**步骤2** 执行以下命令下载wireguard-tools源码包。

```
wget https://git.zx2c4.com/wireguard-tools/snapshot/wireguard-tools-1.0.20210914.tar.xz
```

**步骤3** 执行以下命令解压上述源码包。

```
tar -xf wireguard-tools-1.0.20210914.tar.xz
```

**步骤4** 进入wireguard-tools-1.0.20210914/src目录，依次执行以下命令编译安装。

```
make  
make install
```

**步骤5** 验证安装是否成功。

可以执行**wg -h**和**wg-quick -h**命令验证是否安装成功，如图所示。

图 8-1 验证是否安装成功

```
[root@localhost ~]# wg -h
Usage: wg <cmd> [<args>]

Available subcommands:
  show: Shows the current configuration and device information
  showconf: Shows the current configuration of a given WireGuard interface, for use with 'setconf'
  set: Change the current configuration, add peers, remove peers, or change peers
  setconf: Applies a configuration file to a WireGuard interface
  addconf: Appends a configuration file to a WireGuard interface
  syncconf: Synchronizes a configuration file to a WireGuard interface
  genkey: Generates a new private key and writes it to stdout
  genpsk: Generates a new preshared key and writes it to stdout
  pubkey: Reads a private key from stdin and writes a public key to stdout
You may pass '--help' to any of these subcommands to view usage.
[root@localhost ~]# wg-quick -h
Usage: wg-quick [ up | down | save | strip ] [ CONFIG_FILE | INTERFACE ]

CONFIG_FILE is a configuration file, whose filename is the interface name
followed by '.conf'. Otherwise, INTERFACE is an interface name, with
configuration found at /etc/wireguard/INTERFACE.conf. It is to be readable
by wg(8)'s 'setconf' sub-command, with the exception of the following additions
to the [Interface] section, which are handled by wg-quick:

- Address: may be specified one or more times and contains one or more
  IP addresses (with an optional CIDR mask) to be set for the interface.
- DNS: an optional DNS server to use while the device is up.
- MTU: an optional MTU for the interface; if unspecified, auto-calculated.
- Table: an optional routing table to which routes will be added; if
  unspecified or 'auto', the default table is used. If 'off', no routes
  are added.
- PreUp, PostUp, PreDown, PostDown: script snippets which will be executed
  by bash(1) at the corresponding phases of the link, most commonly used
  to configure DNS. The string '%i' is expanded to INTERFACE.
- SaveConfig: if set to 'true', the configuration is saved from the current
  state of the interface upon shutdown.

See wg-quick(8) for more info and examples.
[root@localhost ~]#
```

----结束

# 9 如何将 docker 工具的用户凭证保存方式配置成与社区一致?

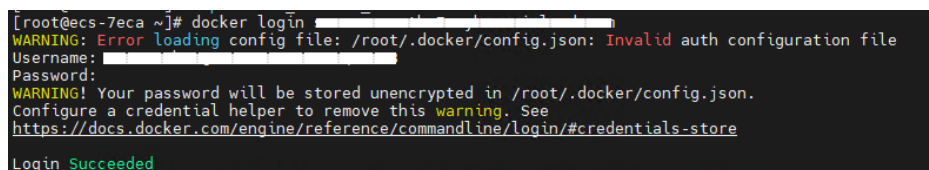
## 问题背景

社区版本的docker工具，使用docker login命令登录成功后，会将用户的用户名、密码等数据以base64的格式保存在用户配置文件，存在较大安全隐患，所以HCE 2.0提供的docker工具，将默认的保存方式改为了加密保存。部分社区工具暂时不支持该安全特性，需要手动将保存方式改为社区的保存方案。

## 如何将凭证保存方式修改为社区方案

1. 配置环境变量  
export USE\_DECRYPT\_AUTH=true
2. 使用docker login命令重新登录  
docker login

图 9-1 使用 docker login 命令重新登录



```
[root@ecs-7eca ~]# docker login
WARNING: Error loading config file: /root/.docker/config.json: Invalid auth configuration file
Username:
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
```

3. 验证完成后，建议将环境变量配置保存在持久文件中（如 ~/.bash\_profile，/etc/profile等），以便重启后生效。  
echo "export USE\_DECRYPT\_AUTH=true" >> ~/.bash\_profile

# 10 free 和 available 说明

---

当我们在cat /proc/meminfo的时候，经常会关注MemFree和MemAvailable字段，下面对其进行说明。

1. MemFree：表示当前完全没有被使用的物理内存。
  - 它来自伙伴系统（buddy allocator）中未分配的页面数量。
  - 不包括用作 page cache、slab 缓存、buffer 的内存。
2. MemAvailable：内核估算的应用程序还能用的内存（考虑了回收缓存的可能性），包含：
  - MemFree。
  - 大部分的 page cache（可回收的）。
  - 可回收的 slab 内存。
  - 还会考虑 min\_free\_kbytes 等保留内存阈值。

---

## 注意

MemFree 小 ≠ 系统要 OOM，只要有足够的可回收内存和 swap，MemFree 少也不会 OOM，看 OOM 风险，要看 MemAvailable。

---

# 11 系统内存回收机制说明

## Linux 系统启动内存回收的时机

1. kswapd 后台回收（异步）。
  - 内核的守护线程 kswapd 会不断检查内存水位：
    - high：正常，不回收。
    - low：唤醒 kswapd，尝试回收一些页。
    - min：即使 kswapd 也没来得及回收，系统非常紧张。
  - 特点：后台执行，不阻塞进程，属于“温和的内存回收”。
2. direct reclaim 前台回收（同步）。
  - 当某个进程分配内存时发现系统内存不够，会直接触发回收。
  - 特点：这个回收过程是同步的，会阻塞进程执行。
3. OOM
  - 如果直接内存回收后，空闲的物理内存仍然无法满足此次物理内存的申请，那么内核就会触发OOM机制（kill进程或者panic）。

## Linux 系统内存回收策略

1. 优先回收 page cache（文件页）。
  - 如果是 clean page：直接丢掉 → 成本最低。
  - 如果是 dirty page：写回磁盘再丢掉。
2. 回收 slab 缓存（dentry/inode 等内核对象）。
  - 内核自身的对象缓存（slab/slub 分配器管理的内存），这些缓存大多可以回收（比如 dentry cache、inode cache）。
  - 内核会通过 shrinker 机制（shrink\_slab()）来释放部分。
3. 回收匿名页（堆/栈）→ swap。
  - 匿名页 = 进程的堆、栈、malloc 出来的内存。这类内存没有磁盘后备存储，如果要回收，就必须写到 swap 分区。内核会把不活跃的匿名页写到 swap 分区，再释放物理页。
  - 代价比回收缓存要高得多。
4. 最后 OOM Kill。



- 如果 page cache 已经回收、swap 已经写满，还是没办法满足内存需求，Linux 会触发 OOM Killer。

# 12 os 内存预留说明

---

kdump机制需要一个“备用内核”来在系统崩溃时接管。为了能让备用内核（crash kernel）顺利启动，需要在系统启动时就预留一块物理内存给它。这个参数就是告诉内核：预留多少内存给 crash kernel 使用。

如果启动参数配置为crashkernel=auto，系统就会根据机器的物理内存大小，自动决定预留多少 crash kernel。不同架构，配置crashkernel=auto时会分配不同的内存大小，x86和arm会根据规格自动分配对应的内存。

# 13 OOM 相关参数配置与原因排查

## OOM 相关概念

OOM ( Out Of Memory, 简称OOM ) 指系统内存已用完, 在linux系统中, 如果内存用完会导致系统无法正常工作, 触发系统panic或者OOM killer。

OOM killer是linux内核的一个机制, 该机制会监控那些占用内存过大的进程, 尤其是短时间内消耗大量内存的进程, 在系统的内存即将不够用时结束这些进程从而保障系统的整体可用性。

## OOM 相关参数

表 13-1 OOM 相关参数

参数名称	参数说明	取值	修改方式
panic_on_oom	<p>panic_on_oom参数是控制系统遇到OOM时如何反应的。当系统遇到OOM的时候, 通常会有两种选择:</p> <ul style="list-style-type: none"><li>• 触发系统panic, 可能会出现频繁宕机的情况。</li><li>• 选择一个或者几个进程, 触发OOM killer, 结束选中的进程, 释放内存, 让系统保持整体可用。</li></ul>	<p>可以通过以下命令查看参数取值:</p> <p><b>cat /proc/sys/vm/panic_on_oom</b>或者</p> <p><b>sysctl -a   grep panic_on_oom</b></p> <ul style="list-style-type: none"><li>• 值为0: 内存不足时, 触发OOM killer。</li><li>• 值为1: 内存不足时, 根据具体情况可能发生kernel panic, 也可能触发OOM killer。</li><li>• 值为2: 内存不足时, 强制触发系统panic, 导致系统重启。</li></ul> <p><b>说明</b> HCE中参数默认值为0。</p>	<p>例如将参数设置为0, 可用以下两种方式:</p> <ul style="list-style-type: none"><li>• 临时配置, 立即生效, 但重启后恢复成默认值。 <b>sysctl -w vm.panic_on_oom=0</b></li><li>• 持久化配置, 系统重启仍生效。执行命令<b>vim /etc/sysctl.conf</b>, 在该文件中添加一行<b>vm.panic_on_oom =0</b>, 再执行命令<b>sysctl -p</b>或重启系统后生效。</li></ul>

参数名称	参数说明	取值	修改方式
oom_kill_allocating_task	<p>当系统选择触发OOM killer，试图结束某些进程时，oom_kill_allocating_task参数会控制选择哪些进程，有以下两种选择：</p> <ul style="list-style-type: none"> <li>触发OOM的进程。</li> <li>oom_score得分最高的进程。</li> </ul>	<p>可以通过以下命令查看参数取值：</p> <pre>cat /proc/sys/vm/oom_kill_allocating_task或者sysctl -a   grep oom_kill_allocating_task</pre> <ul style="list-style-type: none"> <li>值为0：选择oom_score得分最高的进程。</li> <li>值为非0：选择触发OOM的进程。</li> </ul> <p><b>说明</b> HCE中参数默认值为0。</p>	<p>例如将该参数设置成1，可用以下两种方式：</p> <ul style="list-style-type: none"> <li>临时配置，立即生效，但重启后恢复成默认值。 <b>sysctl -w vm.oom_kill_allocating_task=1</b></li> <li>持久化配置，系统重启仍生效。执行命令<b>vim /etc/sysctl.conf</b>，在该文件中添加一行<b>vm.oom_kill_allocating_task=1</b>，再执行命令<b>sysctl -p</b>或重启系统后生效。</li> </ul>
oom_score	<p>指进程的得分，主要由两部分组成：</p> <ul style="list-style-type: none"> <li>系统打分，主要是根据该进程的内存使用情况由系统自动计算。</li> <li>用户打分，也就是oom_score_adj，可以自定义。</li> </ul>	<p>可以通过调整oom_score_adj的值进而调整一个进程最终的得分。通过以下命令查看参数取值：</p> <pre>cat /proc/进程id/oom_score_adj</pre> <ul style="list-style-type: none"> <li>值为0：不调整oom_score。</li> <li>值为负值：在实际打分值上减去一个折扣。</li> <li>值为正值：增加该进程的oom_score。</li> </ul> <p><b>说明</b> oom_score_adj的取值范围是-1000~1000。 若设定成OOM_SCORE_ADJ_MIN或-1000，则表示禁止OOM killer结束该进程。</p>	<p>例如将进程id为2939的进程oom_score_adj参数值设置为1000，可用以下命令：</p> <pre>echo 1000 &gt; /proc/2939/oom_score_adj</pre>

参数名称	参数说明	取值	修改方式
oom_dump_tasks	<p>oom_dump_tasks参数控制OOM发生时是否记录系统的进程信息和OOM killer信息。</p> <p>例如dump系统中所有的用户空间进程关于内存方面的一些信息，包括：进程标识信息、该进程使用的内存信息、该进程的页表信息等，这些信息有助于了解出现OOM的原因。</p>	<p>可以通过以下命令查看参数取值：</p> <p><b>cat /proc/sys/vm/oom_dump_tasks</b>或者<b>sysctl -a   grep oom_dump_tasks</b></p> <ul style="list-style-type: none"><li>值为0：OOM发生时不会打印相关信息。</li><li>值为非0：以下三种情况会调用dump_tasks打印系统中所有task的内存状况。<ul style="list-style-type: none"><li>由于OOM导致kernel panic。</li><li>没有找到需要结束的进程。</li><li>找到进程并将其结束的时候。</li></ul></li></ul> <p><b>说明</b> HCE中参数默认值为1。</p>	<p>例如将该参数设置成0，可用以下两种方式：</p> <ul style="list-style-type: none"><li>临时配置，立即生效，但重启后恢复成默认值。<b>sysctl -w vm.oom_dump_tasks=0</b></li><li>持久化配置，系统重启仍生效。执行命令<b>vim /etc/sysctl.conf</b>，在该文件中添加一行<b>vm.oom_dump_tasks=0</b>，再执行命令<b>sysctl -p</b>或重启系统后生效。</li></ul>

## 触发 OOM killer 示例

- 您可以参考[表13-1](#)设置HCE系统参数，示例配置如下：

```
[root@localhost ~]# cat /proc/sys/vm/panic_on_oom
0
[root@localhost ~]# cat /proc/sys/vm/oom_kill_allocating_task
0
[root@localhost ~]# cat /proc/sys/vm/oom_dump_tasks
1
```

- panic\_on\_oom=0，表示在发生系统OOM的时候触发OOM killer。
- oom\_kill\_allocating\_task=0，表示触发OOM killer的时候优先选择结束得分高的进程。
- oom\_dump\_tasks=1，表示系统发生OOM的时候记录系统的进程信息和OOM killer信息。

- 启动测试进程。

在系统中同时启动三个相同的测试进程（test、test1、test2），不断申请新的内存，并将test1的oom\_score\_adj设置成最大的1000，表示OOM killer优先结束该进程，直至内存耗尽触发系统OOM。

```
[root@localhost ~]# ps -ef | grep test
root    2938   2783  0 19:08 pts/2    00:00:00 ./test
root    2939   2822  0 19:08 pts/3    00:00:00 ./test1
root    2940   2918  0 19:08 pts/5    00:00:00 ./test2
[root@localhost ~]# echo 1000 > /proc/2939/oom_score_adj
[root@localhost ~]# cat /proc/2939/oom_score_adj
1000
```

### 3. 查看OOM信息。

经过一段时间后系统发生OOM并触发OOM killer，同时在/var/log/messages中打印系统所有进程的内存等信息并结束了test1进程：

### 图 13-1 查看进程信息

```
[root@localhost ~]# ps -ef | grep test
root      2938      2783    0 19:08 pts/2        00:00:06 ./test
root      2940      2918    0 19:08 pts/5        00:00:06 ./test2
root      3383      2861    0 19:41 pts/4        00:00:00 grep --color=auto test
[root@localhost ~]#
```

### 图 13-2 oom 时的错误日志

```

mar-19 15:15:14 localhost kernel: [5865.355570] kernel fault[002] notification starting on CPU 15
mar-19 15:15:14 localhost kernel: [5865.355598] kernel fault[002] notification finished on CPU 15
mar-19 15:15:14 localhost kernel: [5865.355600] local host 0ms of mknod[000][GPP nHdSsdE MOVABLE], orders=
mar-19 15:15:14 localhost kernel: [5865.355596] "cpu" PPU - "OpenStack.com" top module: "loaded tainted v=0" 5.10.0-136.12.0.el8_4.r1466.73.xzrctoe.x86_64 #41/2014
mar-19 15:15:14 localhost kernel: [5865.355596] Hardware name: OpenStack Foundation OpenStack Nova, BIOS rel=i.112.1.0-gasC85B-202703913-150422-zxcrtosx86_64 04/01/2014
mar-19 15:15:14 localhost kernel: [5865.355597] all trace
mar-19 15:15:14 localhost kernel: [5865.355570] dump_stack+0x5f/0x60
mar-19 15:15:14 localhost kernel: [5865.355570] Out of core debug info part.0=>0x4y0x131
mar-19 15:15:14 localhost kernel: [5865.355570] ? printk=>0x3/0x3
mar-19 15:15:14 localhost kernel: [5865.355575] out of memory.cold+0x6c/0x60
mar-19 15:15:14 localhost kernel: [5865.355570] alloc_pages+0x6e/0x200
mar-19 15:15:14 localhost kernel: [5865.355580] pagecache_get_page+0x1c3/0x560
mar-19 15:15:14 localhost kernel: [5865.355582] filemap_fault+0x3a2/0x700
mar-19 15:15:14 localhost kernel: [5865.355580] ext4_filemap_fault+0x2c/0x250 [ext4]
mar-19 15:15:14 localhost kernel: [5865.356070] do_fault+0x37/0x1e0
mar-19 15:15:14 localhost kernel: [5865.356071] __do_fault+0x432/0xf0
mar-19 15:15:14 localhost kernel: [5865.356073] do_fault+0x57/0x150
mar-19 15:15:14 localhost kernel: [5865.356074] handle_mm_fault+0x4d4/0xb70
mar-19 15:15:14 localhost kernel: [5865.356076] handle_mm_fault+0xb/0x19
mar-19 15:15:14 localhost kernel: [5865.356079] exc_page_fault+0x1d/0x50
mar-19 15:15:14 localhost kernel: [5865.356079] ? ans_exc_page_fault+0x4/0x30
mar-19 15:15:14 localhost kernel: [5865.356081] ans_exc_page_fault+0x1a/0x30
mar-19 15:15:14 localhost kernel: [5865.356086] RIP: 0013:0x7fb7b08b560
mar-19 15:15:14 localhost kernel: [5865.356086] Code: unable to access opcode bytes at RIP 0x7fb7b08b560.
mar-19 15:15:14 localhost kernel: [5865.356094] RSP: 0020:00007ffefed523a8 EFLAGS: 00010022
mar-19 15:15:14 localhost kernel: [5865.356096] RAX: 00007fb7b08b560 RBX: 0000000000000000 RCX: 0000000000000000
mar-19 15:15:14 localhost kernel: [5865.356096] RDX: 00007fb7b08b1c80 RSI: 0000000000000000 RDI: 0000000000000000
mar-19 15:15:14 localhost kernel: [5865.356097] ROP007fb7b08b7800 RBP: 00007ffefed523d0 R02: 0000000000000000
mar-19 15:15:14 localhost kernel: [5865.356097] ROP007fb7b08b7800 RBP: 00007ffefed523d0 R03: 00007ffefed523e0
mar-19 15:15:14 localhost kernel: [5865.356099] R04: 0000000000000000 R10: 0000000000000000 R11: 00007ffefed523e0
mar-19 15:15:14 localhost kernel: [5865.356099] R12: 0000000000000000 R13: 0000000000000000 R14: 0000000000000000 R15: 0000555fde1d1611
mar-19 15:15:14 localhost kernel: [5865.356099] error: RIPNPrintAllMemory is NULL
mar-19 15:15:14 localhost kernel: [5865.360270] #12 slab info
mar-19 15:15:14 localhost kernel: [5865.360280] slabinfo - version: 2.1
mar-19 15:15:14 localhost kernel: [5865.360240] # name <active_objs> <num_objs> <objsize> <objperslab> <pagesperslab> | tunables <-batchcount> <sharedfactor>
mar-19 15:15:14 localhost kernel: [5865.360243] nf_conntrack_expect 0 0 248 33 2 : tunables 0 0 0 0 : slabdata 124 11 0
mar-19 15:15:14 localhost kernel: [5865.360243] tunables 0 0 0 0 : slabdata 1 1 0
mar-19 15:15:14 localhost kernel: [5865.360243] rc_inode_cache 46 46 704 46 8 : tunables 0 0 0 0 : slabdata 1 1 0

```

### 图 13-3 触发 oom 的进程

[illegible]

## OOM 可能的原因

- cgroup内存不足

使用的内存超出了cgroup中memory.limit\_in\_bytes配置的大小，如下示例演示memory.limit\_in\_bytes配置为80M，使用memhog模拟分配100M，触发OOM，/var/log/messages部分日志如下，可以从日志中看到memhog所在进程（PID: 2021820）使用了81920kB内存，超出了限制，触发了OOM：

```
warning|kernel[-]|[2919920.414131] memhog invoked oom-killer: gfp_mask=0xcc0(GFP_KERNEL),
order=0, oom_score_adj=0
info|kernel[-]|[2919920.414220] memory: usage 81920kB, limit 81920kB, failcnt 30
err|kernel[-]|[2919920.414272] Memory cgroup out of memory: Killed process 2021820 (memhog)
total-vm:105048kB, anon-rss:81884kB, file-rss:1544kB, shm-emem-rss:0kB, UID:0 pgtables:208kB
oom score_adj:0
```

- 父cgroup内存不足

在子cgroup中内存仍然足够，但是父cgroup的内存不足，超过了内存限制，如下示例演示父cgroup memory.limit\_in\_bytes配置为80M，两个子cgroup memory.limit\_in\_bytes均配置为50M，在两个子cgroup中使用程序循环分配内存，触发OOM，/var/log/messages部分日志如下：

```
warning|kernel[-]|[2925796.529231] main invoked oom-killer: gfp_mask=0xcc0(GFP_KERNEL),
order=0, oom_score_adj=0
info|kernel[-]|[2925796.529315] memory: usage 81920kB, limit 81920kB, failcnt 199
err|kernel[-]|[2925796.529366] Memory cgroup out of memory: Killed process 3238866 (main) total-
vm:46792kB, anon-rss:44148kB, file-rss:1264kB, shmem-rss:0kB, UID:0 ptables:124kB oom_score_adj:0
```

- 系统全局内存不足

一方面由于OS的空闲内存不足，有程序一直在申请内存，另一方面也无法通过内存回收机制解决内存不足的问题，因此触发了OOM，如下示例演示OS中使用程序循环分配内存，触发OOM，/var/log/messages部分日志如下，可以从日志中看到内存节点Node 0的空闲内存（free）已经低于了内存最低水位线（low），触发了OOM：

```
kernel: [ 1475.869152] main invoked oom: gfp_mask=0x100dca(GFP_HIGHUSER_MOVABLE|
__GFP_ZERO), order=0
kernel: [ 1477.959960] Node 0 DMA32 free:22324kB min:44676kB low:55844kB high:67012kB
reserved_highatomic:0kB active_anon:174212kB inactive_anon:1539340kB active_file:0kB
inactive_file:64kB unevictable:0kB writepending:0kB present:2080636kB managed:1840628kB
mlocked:0kB pagetables:7536kB bounce:0kB free_pcp:0kB local_pcp:0kB free_cma:0kB
kernel: [ 1477.960064] oom-
kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_mem
cg=/system.slice/sshd.service,task=main,pid=1822,uid=0
kernel: [ 1477.960084] Out of memory: Killed process 1822 (main) total-vm:742748kB, anon-
rss:397884kB, file-rss:4kB, shmem-rss:0kB, UID:0 pgtables:1492kB oom_score_adj:1000
```

- 内存节点（Node）的内存不足

在NUMA存储模式下，OS会存在多个内存节点，如果程序指定使用特定节点的内存，可能在OS内存充足的情况下触发OOM，如下示例演示在两个内存节点的情况下，使用程序循环在Node 1分配内存，导致Node 1内存不足，但是OS内存足够，/var/log/messages部分日志如下：

```
kernel: [ 465.863160] main invoked oom: gfp_mask=0x100dca(GFP_HIGHUSER_MOVABLE|
__GFP_ZERO), order=0
kernel: [ 465.878286] active_anon:218 inactive_anon:202527 isolated_anon:0#012 active_file:5979
inactive_file:5231 isolated_file:0#012 unevictable:0 dirty:0 writeback:0#012 slab_reclaimable:6164
slab_unreclaimable:9671#012 mapped:4663 shmem:2556 pagetables:846 bounce:0#012 free:226231
free_pcp:36 free_cma:0
kernel: [ 465.878292] Node 1 DMA32 free:34068kB min:32016kB low:40020kB high:48024kB
reserved_highatomic:0kB active_anon:188kB inactive_anon:778076kB active_file:20kB
inactive_file:40kB unevictable:0kB writepending:0kB present:1048444kB managed:866920kB
mlocked:0kB pagetables:2752kB bounce:0kB free_pcp:144kB local_pcp:0kB free_cma:0kB
kernel: [ 933.264779] oom-
kill:constraint=CONSTRAINT_MEMORY_POLICY,nodemask=1,cpuset=/,mems_allowed=0-1,global_oom,
task_memcg=/system.slice/sshd.service,task=main,pid=1733,uid=0
kernel: [ 465.878438] Out of memory: Killed process 1734 (main) total-vm:239028kB, anon-
rss:236300kB, file-rss:200kB, shmem-rss:0kB, UID:0 pgtables:504kB oom_score_adj:1000
```

- 其他可能原因

OS在内存分配的过程中，如果伙伴系统的内存不足，则系统会通过OOM Killer释放内存，并将内存提供至伙伴系统。

## OOM 问题解决方法

- 从业务进程排查，确认是否有内存泄漏，导致OOM。
- 排查cgroup limit\_in\_bytes配置是否与业务内存规划匹配，如需要调整，可以手动执行以下命令修改配置参数：

```
echo <value> > /sys/fs/cgroup/memory/<cgroup_name>/memory.limit_in_bytes
```
- 如果确认业务需要比较多的内存，建议升级弹性云服务器内存规格。

# 14 IPVS 报错问题说明

## 问题背景

IPVS ( IP Virtual Server ) 指IP虚拟服务器，用于负载均衡、网络转发等目的。用户在系统上配置了IPVS虚拟服务器，但未配置真实服务器的情况下，会在VNC上出现错误日志。

## 问题现象

配置了IPVS虚拟服务器，但未配置真实服务器时，当网络请求发往该虚拟服务器地址后，通过华为云VNC登录的控制台上可以看到类似如下的错误日志。

图 14-1 错误日志

```
[32264.645949][T268365] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
[32265.234919][T268366] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
[32265.954662][T268367] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
[32266.557032][T268368] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
[32267.166530][T268369] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
[32267.725920][T268370] IPVS: wlc: TCP 192.168.1.1:5000 - no destination available
```

## 解决方法

1. 安装ipvsadm。
2. 执行ipvsadm -Ln，查询当前虚拟服务器的配置。找到报错的虚拟服务器对应的表项。

图 14-2 未配置真实服务器

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port[:Subnet] Scheduler Established(Sec.) Flags
  -> RemoteAddress:Port[:0if] Forward Weight ActiveConn InActConn UtepAddr:utepport Unild Mac
TCP 192.168.1.1:5000 wlc
```

如上图所示没有对应的真实服务器，则说明配置不完整，会引发错误打印。需要排查对应的业务流程是否正确。

图 14-3 已配置真实服务器

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port[:Subnet] Scheduler Established(Sec.) Flags
  -> RemoteAddress:Port[:0if] Forward Weight ActiveConn InActConn UtepAddr:utepport Unild Mac
TCP 192.168.1.1:5000 wlc
  -> 192.168.1.2:5000 Masq 1 0 0
```



如上图是完整的有真实服务器的配置。

3. 如果希望排除该IPVS错误日志在VNC上对用户操作的干扰，可以按如下的方式处理（选择其中一种即可）

- 关闭业务发送的网络请求，具体操作需要用户根据自身业务情况来处理。
- 执行以下命令调整内核printk打印等级。

```
echo 3 4 1 7 > /proc/sys/kernel/printk
```

#### 说明

如果临时修改系统配置，建议用户选择适当的时机恢复系统配置。

- 用华为云的CloudShell方式登录云服务器进行操作。

# 15 中文环境执行 sulogin 命令终端显示乱码说明

## 问题背景

使用sulogin命令可以进行单用户登录。sulogin命令目前不支持中文，如果用户将系统语言环境修改为中文，执行sulogin命令时终端会显示乱码。

## 问题现象

执行export LANG="zh\_CN.UTF-8" 修改语言环境为中文后，再执行sulogin终端显示出乱码，如下图所示：

图 15-1 sulogin 终端显示出现乱码

```
[root@localhost ~]# export LANG="zh_CN.UTF-8"
[root@localhost ~]# sulogin
?? root ?????
(?? Control-D ???):
```

## 解决方法

执行sulogin命令时，可以临时设置LANG环境变量为英文，比如将LANG设置为en\_US.UTF-8：

图 15-2 设置 LANG 环境变量为英文

```
[root@localhost ~]# LANG="en_US.UTF-8" sulogin
Give root password for maintenance
(or press Control-D to continue):
[root@localhost ~]#
```

# 16 ECS 开启 IPv6 后，HCE 系统内无法获取到 IPv6 地址

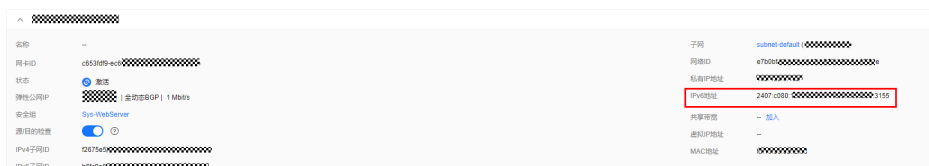
## 问题背景

在弹性云服务器ECS控制台上开启云服务器网卡的IPv6功能后，由于未在操作系统内部正确配置IPv6，导致HCE系统内无法获取到IPv6地址。

## 问题现象

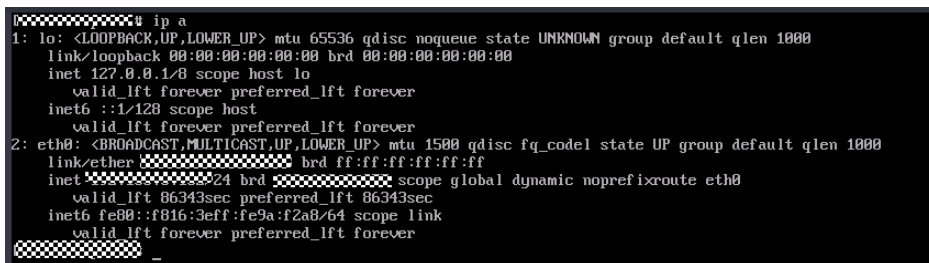
在弹性云服务器ECS控制台上已开启IPv6功能，在详情界面已显示IPv6地址。

图 16-1 详情界面 IPv6 地址



但是进入操作系统内部，无法获取到IPv6地址。

图 16-2 操作系统内部



## 解决方法

1. 手动配置dhcp自动获取ipv6地址，如下图在对应网卡配置文件（/etc/sysconfig/network-scripts/ifcfg-ethx）中添加以下参数。  
IPV6INIT="yes"  
DHCPV6C="yes"

图 16-3 添加参数

```
[root@ ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
BOOTPROTO="dhcp"
ONBOOT="yes"
TYPE="Ethernet"
PERSISTENT_DHCLIENT="yes"
IPV6INIT="yes"
DHCPV6C="yes"
[root@ ~]#
```

2. 执行以下命令重启NetworkManager服务即可获取到IPv6地址。  
systemctl restart NetworkManager

图 16-4 获取 IPv6 地址

```
[root@ ~]# systemctl restart NetworkManager
[root@ ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:00:00:00 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.1/24 brd 10.0.0.255 scope global dynamic noprefixroute eth0
        valid_lft 86394sec preferred_lft 86394sec
    inet6 2407::3155/128 scope global dynamic noprefixroute
        valid_lft 7195sec preferred_lft 7195sec
    inet6 fe80::f816:3eff:fe9a:f2a8/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
[root@ ~]#
```

# 17 如何设置自动注销时间 TMOUT?

## 操作背景

为了保证系统的安全性，以及减少用户在不使用系统时所造成的资源浪费，在用户离开系统一段时间后，必须对连接进行注销。注销有诸多方法，修改TMOUT变量为其中的解决方案之一。

TMOUT是一个环境变量，它决定了在系统自动注销前所空闲的秒数。因此，在设置了此变量后，若用户在规定时间内没有进行主动活动，则连接将自动断开。若没有设置此变量，或值为0，表示禁用自动注销，连接不会因用户长时间不活动而断开。

## 临时设置自动注销时间 TMOUT

1. 执行以下命令查看自动注销时间（即TMOUT的值）。

```
echo $TMOUT
```

显示空白表示没有设置此值。

2. 执行以下命令，设置当前会话下的自动注销时间。

```
export TMOUT=秒数
```

## 永久设置自动注销时间 TMOUT

### 方式一

执行以下命令修改文件/etc/profile（若修改此文件不生效，可再修改/etc/bashrc，修改流程相同），这样可以使所有应用此配置文件的用户均受此自动注销时间的影响。

```
vim /etc/profile
```

或者

```
vim /etc/bashrc
```

添加以下命令至文件末尾。以设置自动注销时间为1200秒为例，实际值可自定义，设置为0禁用注销功能。

```
export TMOUT=1200
```

图 17-1 设置自动注销时间

```
test -r /etc/bashrc
then
# Bash login shells run only /etc/profile
# Bash non-login shells run only /etc/bash
# Check for double sourcing is done in /etc
. /etc/bashrc
fi

export TMOUT=1200
```

保存文件后执行以下命令刷新。

```
source /etc/profile
```

### 方式二

依次执行以下命令直接修改自动注销时间。

```
sed -i '$a\export TMOUT=1200' /etc/profile
source /etc/profile
```

执行以下命令查看自动注销时间。

```
echo $TMOUT
```

若显示定义的数值，则说明自动注销功能设置成功。

图 17-2 自动注销功能设置成功

```
[root@localhost ~]# echo $TMOUT
1200
```

# 18 panic\_on\_oom 系统参数变更说明

## 参数说明

panic\_on\_oom参数是控制系统遇到OOM时如何反应的。当系统遇到OOM的时候，通常会有两种选择：

- 触发系统panic，可能会出现频繁宕机的情况。
- 选择一个或者几个进程，触发OOM killer，结束选中的进程，释放内存，让系统保持整体可用。

可以通过以下命令查看参数取值：

**cat /proc/sys/vm/panic\_on\_oom**或者

**sysctl -a | grep panic\_on\_oom**

- 值为0：内存不足时，触发OOM killer。
- 值为1：内存不足时，根据具体情况可能发生kernel panic，也可能触发OOM killer。
- 值为2：内存不足时，强制触发系统panic，导致系统重启。

## 变更说明

在HCE 2.0.2503版本之前，panic\_on\_oom参数默认值为1，在HCE 2.0.2503版本之后，panic\_on\_oom参数默认值修改为0。

由HCE 2.0.2503之前的版本升级到HCE 2.0.2503版本之后，再重新回退到升级之前的版本，此时，panic\_on\_oom参数默认为0。如果需要在回退后的版本中对该参数进行修改，可以参考以下方式：

- 临时配置，立即生效，但重启系统后恢复成默认值。  
比如要把panic\_on\_oom参数配置为1，则可以执行命令：  
`sysctl -w vm.panic_on_oom=1`

- 持久化配置，系统重启后仍然有效。  
比如要把panic\_on\_oom参数配置为1，则可以执行命令：  
`vim /etc/sysctl.conf`

在该文件中添加一行 `vm.panic_on_oom = 1`，然后再执行命令 `sysctl -p` 或者重启系统即可持久化该配置。

# 19 如何添加缺少的字符集

默认情况下，HCE系统中只安装了部分常用的字符集。如果您需要使用额外的字符集，请参考如下步骤手动安装：

 **注意**

安装前请确认系统的repo源配置正确，详情请参见[HCE的REPO源配置与软件安装](#)。

**步骤1** 执行dnf install glibc-all-langpacks glibc-locale-archive -y。

图 19-1 安装字符集

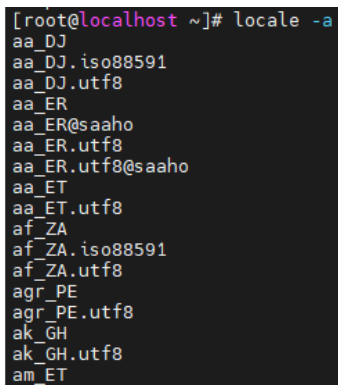
```
Last metadata expiration check: 0:46:51 ago on Sat 01 Mar 2025 10:50:12 AM CST.
Dependencies resolved.
=====
Package                                Architecture Version                                Repository                               Size
=====
Installing:
glibc-all-langpacks                   aarch64    2.34-70.r88.hce2                      hceversion                               28 M
glibc-locale-archive                  aarch64    2.34-70.r88.hce2                      hceversion                               26 M
=====
Transaction Summary
=====
Install 2 Packages

Total download size: 54 M
Installed size: 431 M
Downloading Packages:
(1/2): glibc-locale-archive-2.34-70.r88.hce2.aarch64.rpm 47 MB/s | 26 MB 00:00
(2/2): glibc-all-langpacks-2.34-70.r88.hce2.aarch64.rpm 44 MB/s | 28 MB 00:00
-----
Total                                                    85 MB/s | 54 MB 00:00
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
```

**步骤2** 执行locale -a命令查看所需的字符集是否已经包含。



图 19-2 查看已安装字符集



```
[root@localhost ~]# locale -a
aa_DJ
aa_DJ.iso88591
aa_DJ.utf8
aa_ER
aa_ER@saaho
aa_ER.utf8
aa_ER.utf8@saaho
aa_ET
aa_ET.utf8
af_ZA
af_ZA.iso88591
af_ZA.utf8
agr_PE
agr_PE.utf8
ak_GH
ak_GH.utf8
am_ET
```

**步骤3** 如需要切换语言，则使用如下命令进行切换（这里以波兰语为例）：

```
localectl set-locale LANG=pl_PL.UTF-8
```

**步骤4** 重启HCE后生效。

----结束

# 20 如何解决证书切换导致的安全启动失败问题

## 问题背景

2025年3月份之前，在HCE版本发布的RPM包hce-sign-certificate-1.0-1.hce2中，针对安全启动功能提供了两本验签证书：HCE\_Secure\_Boot\_RSA\_Code-Signing\_Authority\_1.cer 和 Huawei\_Code-Signing\_Authority\_CA\_3.der.cer，用于OS安全启动过程中shim，grub，vmlinuz组件的验签。这两本证书任意一个都可以导入BIOS，作为验签证书，其中HCE\_Secure\_Boot\_RSA\_Code-Signing\_Authority\_1.cer证书在2025.04已经到期了，无法使用。

如果服务器安装了2025年3月份之前HCE版本，开启了安全启动功能且只导入了HCE\_Secure\_Boot\_RSA\_Code-Signing\_Authority\_1.cer证书，在升级2025年5月份之后的HCE版本，重启后，会存在系统无法引导启动的问题。

## 排查方法

- 步骤1** 确认当前安装的系统版本是否为2025年3月份之前的HCE2.0版本。如果是，执行步骤2；如果否，不存在该问题。

```
[root@localhost ~]# cat /etc/hce-latest  
hceversion=HCE-2.0.2412.1_aarch64  
compiletime=2025-02-28-10-00-01
```

- 步骤2** 执行mokutil --sb查看系统是否开启安全启动功能。如果显示“SecureBoot enabled”，执行步骤3；如果否，不存在该问题。

```
[root@localhost ~]# mokutil --sb  
SecureBoot enabled
```

- 步骤3** 查看BIOS中导入的证书。

```
[root@localhost ~]# mokutil --db | grep "Subject:"
```

图 20-1 系统导入证书查询

```
X509v3 Subject Key Identifier:  
[root@localhost ~]# mokutil --db | grep "Subject:"  
Subject: C=CN, O=Huawei, OU=Huawei Trust Service, CN=HCE Secure Boot RSA Code-Signing Authority 2  
Subject: C=CN, O=Huawei, CN=Huawei Root CA  
[root@localhost ~]#
```

- 如果上述命令查询结果存在“Huawei Root CA”，则不会有该问题，直接跳到结束；

- 如果上述命令查询结果没有“Huawei Root CA”，有“Huawei Code-Signing Authority CA 3”，OS升级到2025年5月份之后的HCE2.0版本，不会有该问题；
- 如果上述命令查询结果中即不存在“Huawei Root CA”，也不存在“Huawei Code-Signing Authority CA 3”，有“HCE Secure Boot RSA Code-Signing Authority 1”，则存在该问题。

----结束

## 解决方案

获取HCE2.0镜像源中更新的证书，[https://repo.huaweicloud.com/hce/2.0/updates/x86\\_64/Packages/](https://repo.huaweicloud.com/hce/2.0/updates/x86_64/Packages/)路径下的hce-sign-certificate-1.0-2.hce2.x86\_64.rpm，解压或安装后，将新证书Huawei\_Root\_CA.cer导入BIOS中。

- BIOS证书导入参考：

鲲鹏服务器：<https://support.huawei.com/enterprise/zh/doc/EDOC1100088653/97a0d5a0>

2288H V5：<https://support.huawei.com/enterprise/zh/doc/EDOC1000163371/afc5c7f8?idPath=23710424|251364409|21782478|21872244>

2288H V6：<https://support.huawei.com/enterprise/zh/doc/EDOC1100195299/fdb56216?idPath=23710424|251364409|21782478|23692812>

# 21 如何进行网卡中断绑核

网卡中断绑核是一种优化网络性能的技术，它可以将网卡的数据处理任务分配到特定的CPU核心上，从而提高网络数据的处理效率和吞吐量。本文介绍如何在HCE 2.0上配置网卡中断绑核。

## 前置条件

确认已经关闭irqbalance服务。如果irqbalance服务未关闭，可以执行systemctl stop irqbalance进行关闭；如果需要取消irqbalance服务开机自启动，执行systemctl disable irqbalance进行配置。

## 操作步骤

1. 执行ethtool -l eth0查看网卡硬件队列数，本例中有两个网卡队列

### 说明

eth0为网卡名，请根据实际情况进行替换，以下步骤均以eth0举例

```
[root@localhost ~]# ethtool -l eth0
Channel parameters for eth0:
Pre-set maximums:
RX:                n/a
TX:                n/a
Other:             n/a
Combined:          2
Current hardware settings:
RX:                n/a
TX:                n/a
Other:             n/a
Combined:          2
[root@localhost ~]#
```

2. 执行lscpu查看当前CPU核心数，本例有4个CPU核心

```
[root@localhost ~]# lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Address sizes:          46 bits physical, 48 bits virtual
Byte Order:             Little Endian
CPU(s):                 4
On-line CPU(s) list:    0-3
Vendor ID:              GenuineIntel
```

3. 执行以下命令查看当前网卡队列对应中断，本例中eth0的数据输入使用的中断号分别是25和27，数据输出使用的中断号分别是26和28

```
cat /proc/interrupts | grep virtio0 | awk '{print $1 $(NF)}'
```

```
combined:
[root@localhost ~]# cat /proc/interrupts | grep virtio0 | awk '{print $1 $(NF)}'
24:virtio0-config
25:virtio0-input.0
26:virtio0-output.0
27:virtio0-input.1
28:virtio0-output.1
[root@localhost ~]#
```

### 说明

网卡和virtio的对应关系可以通过以下命令进行查看：

```
ethtool -i eth0 | grep bus-info | awk -F "bus-info:" '{print $2}' | xargs -I {} ls /sys/bus/pci/drivers/
virtio-pci/{} | grep virtio
```

4. 查看当前网卡中断绑定情况

执行命令cat /proc/irq/{25,26,27,28}/smp\_affinity\_list进行查看，本例中25,26,27,28号中断均绑定在CPU3上

```
[root@localhost ~]# cat /proc/irq/{25,26,27,28}/smp_affinity_list
3
3
3
3
[root@localhost ~]#
```

5. 手动绑定网卡中断

执行以下命令将eth0的数据输入中断分别绑定到CPU0和CPU1

```
echo 0 > /proc/irq/25/smp_affinity_list
echo 1 > /proc/irq/27/smp_affinity_list
```

执行以下命令将eth0的数据输出中断分别绑定到CPU2和CPU3

```
echo 2 > /proc/irq/26/smp_affinity_list
echo 3 > /proc/irq/28/smp_affinity_list
```

6. 验证网卡中断绑定是否成功

执行cat /proc/irq/{25,26,27,28}/smp\_affinity\_list进行查看，显示已设置成功

```
[root@localhost ~]# cat /proc/irq/{25,26,27,28}/smp_affinity_list
0
2
1
3
[root@localhost ~]#
```

# 22 x86\_64 环境如何调整 memcpy 默认水位线?

## 背景信息

glibc memcpy默认水位线参数x86\_non\_temporal\_threshold对内存带宽影响较大，业务可以根据实际情况调整水位线大小以获得较优的内存拷贝性能，本节将介绍如何调整memcpy默认水位线及其具体影响。

## 调整方法

以下为glibc社区推荐配置：

```
export GLIBC_TUNABLES=glibc.cpu.x86_non_temporal_threshold=$((getconf LEVEL3_CACHE_SIZE) * 3 / 4))
```

## memcpy 算法综述

在glibc-2.34中，memcpy和memmove共享一套逻辑，其实现算法在glibc的源码中有简要介绍：

```
sysdeps/x86_64/multiarch/memmove-vec-unaligned-erms.S
```

```
/* memmove/memcpy/memcpy is implemented as:
1. Use overlapping load and store to avoid branch.
2. Load all sources into registers and store them together to avoid
   possible address overlap between source and destination.
3. If size is 8 * VEC_SIZE or less, load all sources into registers
   and store them together.
4. If address of destination > address of source, backward copy
   4 * VEC_SIZE at a time with unaligned load and aligned store.
   Load the first 4 * VEC and last VEC before the loop and store
   them after the loop to support overlapping addresses.
5. Otherwise, forward copy 4 * VEC_SIZE at a time with unaligned
   load and aligned store. Load the last 4 * VEC and first VEC
   before the loop and store them after the loop to support
   overlapping addresses.
6. If size >= __x86_shared_non_temporal_threshold and there is no
   overlap between destination and source, use non-temporal store
   instead of aligned store. */
```

其中，如第6条所述，如果超过\_\_x86\_shared\_non\_temporal\_threshold水位线，将使用non-temporal store代替aligned store。non-temporal store使用的mov指令为movntdq指令即绕过CPU L3 cache直接访问内存，在cache missing下相比aligned store省略了读cache和写cache的操作，比较适用于大块内存拷贝的场景。

# 23 网络服务重启导致 resolv.conf 配置内容发生变更

## 问题现象

在系统重新启动或者网络服务重启时，/etc/resolv.conf中的DNS服务器配置nameserver IP发生变更。

## 原因分析

/etc/resolv.conf中nameserver配置内容变更，与网口配置文件中的参数PEERDNS和RESOLV\_MODS有关。

网络服务重启过程中，脚本/etc/sysconfig/network-scripts/ifup-post和/etc/sysconfig/network-scripts/ifdown-post会对网口配置文件（比如/etc/sysconfig/network-scripts/ifcfg-\*）中的配置项“RESOLV\_MODS=no”或者“PEERDNS=no”进行检查，如果这两个参数不存在或者某一个不存在，那么脚本将修改/etc/resolv.conf中的内容。

参数说明：

1. PEERDNS：是否指定DNS。如果使用DHCP协议，默认为yes。
  - yes：如果DNS配置项存在，修改/etc/resolv.conf中的DNS。
  - no：不修改/etc/resolv.conf中的DNS
2. RESOLV\_MODS：是否写入DNS。
  - yes：文件/etc/resolv.conf中写入MS\_DNS1和MS\_DNS2的值。
  - no：不修改/etc/resolv.conf中的DNS

## 解决方法

在网口配置文件/etc/sysconfig/network-scripts/ifcfg-\*中添加以下配置内容，然后重启网络服务：

```
PEERDNS=no  
RESOLV_MODS=no
```

# 24 如何设置 nohup 后台进程在 VNC 会话退出后不被杀死

## 问题现象

在VNC会话中启动的nohup后台进程，在VNC会话结束（如使用exit命令退出登录，或者登录超时等）后，会被系统主动杀死。

## 原因分析

出于安全等原因考虑，上游社区对getty相关的服务单元进行了修改，删除KillMode=process的配置，导致VNC登录会话退出后，所有的进程均被清理。参见社区提交：<https://github.com/systemd/systemd/commit/021acbc188a53fa528161578305406c5c9c808b2>

## 解决方法

您可通过如下方法修改getty配置来保留您的nohup后台进程不被杀死：

**步骤1** 通过vim /usr/lib/systemd/system/getty@.service命令编辑getty服务单元。

**步骤2** 在[Service]一节中增加KillMode=process配置，并退出保存。

```
[Service]
# the VT is cleared by TTYVDisallocate
# The '-o' option value tells agetty to replace 'login' arguments with an
# option to preserve environment (-p), followed by '--' for safety, and then
# the entered username.
ExecStart=/sbin/agetty -o '-p -- \u' --noclear %I $TERM
Type=idle
Restart=always
RestartSec=0
UtmpIdentifier=%I
TTYPath=/dev/%I
TTYReset=yes
TTYVHangup=yes
TTYVDisallocate=yes
IgnoreSIGPIPE=no
SendSIGHUP=yes
KillMode=process
```

**步骤3** 执行systemctl daemon-reload重新加载服务。

**步骤4** 执行systemctl restart getty@`basename \$(tty)`重启当前getty服务。





重启getty服务会导致当前会话退出。

---

----结束

# 25 系统配置 XPS 的方法及影响说明

## XPS 概述

XPS ( Transmit Packet Steering ) 是一种在多队列网卡上发送数据报文时，自动选择发送队列的机制。它通过建立发送队列和CPU集合之间的映射关系，使得内核能够根据这一映射关系，在某个CPU发送数据报文时，自动选择与该CPU相关联的发送队列来完成报文传输。内核会记录数据流的首个报文选择的发送队列，并将该队列用于后续报文的传输，从而减少为每个报文计算发送队列的开销。

XPS的优势体现如下：

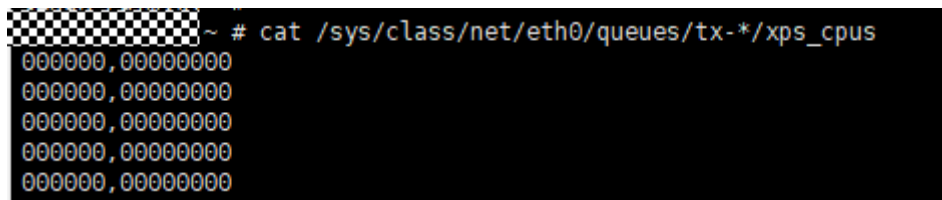
- 缓解不同CPU对同一个发送队列的竞争，进而降低网卡队列发送数据时的锁冲突，提升数据报文发送效率。
- XPS配置的发送队列与CPU之间的映射关系与virtio网卡绑定的发送队列的亲中性一致，这降低了报文发送过程中缓存失效 ( cache miss ) 的可能性和锁竞争导致的缓存失效的可能性，从而提高了网络发送性能。

## 配置 XPS

1. 运行以下命令，检查目标实例是否配置了XPS ( 确保内核使能了CONFIG\_XPS ) 。

以网卡eth0为例：

```
cat /sys/class/net/eth0/queues/tx-*/xps_cpus
```



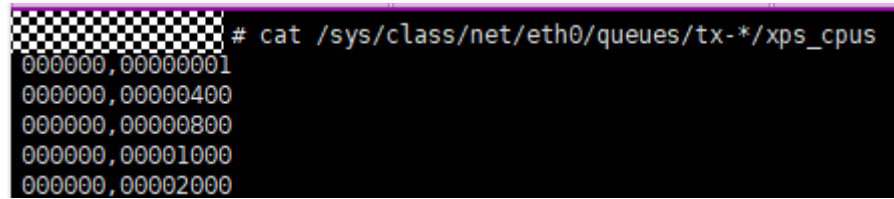
```
~ # cat /sys/class/net/eth0/queues/tx-*/xps_cpus
000000,00000000
000000,00000000
000000,00000000
000000,00000000
000000,00000000
```

2. XPS配置需要结合不同CPU和发送队列数目来进行配置，具体操作详见[开启网卡多队列功能](#)。
3. ( 可选 ) 运行以下命令，检查XPS是否配置完成。

以网卡eth0为例：

```
cat /sys/class/net/eth0/queues/tx-*/xps_cpus
```

如下图所示 ( CPU数目不同或者队列数目不同，显示结果不同 )，说明XPS已经配置成功。

A terminal window with a black background and a checkered cursor. The command being executed is `# cat /sys/class/net/eth0/queues/tx-*/xps_cpus`. The output shows five lines of hexadecimal values: `000000,00000001`, `000000,00000400`, `000000,00000800`, `000000,00001000`, and `000000,00002000`.

```
# cat /sys/class/net/eth0/queues/tx-*/xps_cpus
000000,00000001
000000,00000400
000000,00000800
000000,00001000
000000,00002000
```

## 清除 XPS

尽管配置XPS的目的是为了提升网络性能，但可能会出现配置XPS后网络性能受到影响。如果您遇到这种情况，可以运行以下命令来清除XPS配置。以网卡eth0为例：

```
sudo sh -c 'for txq in /sys/class/net/eth0/queues/tx-*; do echo 0 > $txq/xps_cpus; done'
```

# 26 系统异常重启的原因排查

## 可能性 1: 内存耗尽 (OOM - Out Of Memory)

- 现象描述:  
一个或多个进程消耗了所有物理内存和交换分区 (Swap)。为防止系统完全僵死, 内核的 OOM Killer 会强制终止占用内存最多的进程, 有时这个过程会引发系统连锁反应导致重启。
- 排查方法:  
日志确认: /var/log/messages 中存在 "Out of memory: Killed process" 或 "invoked oom-killer" 等字样。
- 解决方案:
  - a. 从业务进程排查, 确认是否有内存泄漏, 导致 OOM。
  - b. 如果确认业务需要比较多的内存, 建议升级弹性云服务器内存规格。

## 可能性 2: Kernel Panic

- 现象描述:  
内核遇到了无法恢复的致命错误 (如驱动 Bug、内核 Bug、硬件通信故障)。系统会冻结, 并根据配置自动重启。
- 排查方法:

**步骤1** 日志确认: /var/log/messages 中存在 "Kernel panic - not syncing", "Oops" 等关键字。

**步骤2** 检查自动重启配置:

```
cat /proc/sys/kernel/panic
# 输出为 0: 发生 Panic 后挂起, 等待人工处理。
# 输出为 N: 发生 Panic 后 N 秒自动重启。
```

----结束

- 解决方案:
  - a. 驱动/内核回滚: 如果问题出现在近期更新后, 将内核或相关驱动 (如显卡、网卡驱动) 回滚到之前的稳定版本。
  - b. 捕获现场: 为下次故障能保留现场, 可临时禁用自动重启:  
echo 0 > /proc/sys/kernel/panic # 临时修改

### 可能性 3: 计划任务触发重启

- 现象描述:  
配置错误触发了重启。
- 排查方法:  
检查计划任务:  

```
sudo grep -r "reboot\|shutdown" /etc/cron.d/ /etc/cron.hourly/ /etc/cron.daily/ /etc/cron.weekly/ /etc/cron.monthly/ /var/spool/cron/
```
- 解决方案:  
修正配置: 找到并删除或注释错误的重启计划任务。

# 27 操作系统控制台常见问题

本文记录了操作系统控制台中常见的问题

## 27.1 使用操作系统控制台收费吗？

操作系统控制台旨在帮助用户快速定位故障根因、解决问题，目前并无收费计划。

## 27.2 为什么有些 ECS 虚拟机不支持添加？

只有运行中的ECS虚拟机才可以添加。

## 27.3 系统常驻组件的资源消耗高吗？

操作系统控制台添加节点时会自动在节点安装hce-agent插件作为系统常驻组件，负责采集系统/进程热点数据，是控制台所有功能的依赖组件。

常态化运行过程中的资源消耗如下：

- CPU：单核CPU的平均使用率为1%。
- 内存：平均占用内存120MiB，最大不超过200MiB。

## 27.4 如何检查节点是否支持 eBPF 功能？

1. 节点命令行执行如下命令：

```
ls /sys/kernel/btf/vmlinux
```

若执行结果包含/sys/kernel/btf/vmlinux，则继续进行步骤2的检查，否则节点不支持eBPF功能。

2. 节点命令行执行如下命令：

```
cat /boot/config-$(uname -r) | grep CONFIG_PERF_EVENTS
```

若执行结果包含CONFIG\_PERF\_EVENTS=y，则节点支持eBPF功能，否则节点不支持eBPF功能。

## 27.5 节点插件状态离线问题排查

### 问题现象

操作系统控制台中使用“添加节点”功能后，节点插件状态显示“离线”。

### 可能原因

#### 1. 节点未开机

在弹性云服务器控制台的云服务列表中，搜索离线节点名称，查看节点状态，如果节点处于关机状态，需要对节点进行开机操作。



#### 2. 插件未安装或未运行

使用root用户，登录服务器节点，执行如下命令查询插件运行状态。

```
systemctl status hce-agent
```

如果如下图显示服务不存在，可以在节点管理中删除节点后重新添加。

```
[root@allinone-centos ~]# systemctl status hce-agent
Unit hce-agent.service could not be found.
[root@allinone-centos ~]#
```

如果如下图显示服务状态为inactive，可以在节点上执行“systemctl start hce-agent”命令手动启动插件。

```
[root@xg-test-node ~]# systemctl status hce-agent
o hce-agent.service - Huawei Cloud HCE Agent
   Loaded: loaded (/etc/systemd/system/hce-agent.service; enabled; vendor preset: disabled)
   Active: inactive (dead) since Fri 2025-12-12 12:34:39 MSK; 7s ago
   Process: 2149 ExecStart=/usr/local/uniagentd/extension/install/hce-agent/hce-agent -c hce-agent.yml (code=exited, status=0/SUCCESS)
   Main PID: 2149 (code=exited, status=0/SUCCESS)

Dec 12 12:33:46 xg-test-node systemd[1]: Started Huawei Cloud HCE Agent.
Dec 12 12:34:39 xg-test-node systemd[1]: Stopping Huawei Cloud HCE Agent...
Dec 12 12:34:39 xg-test-node hce-agent[2163]:
Dec 12 12:34:39 xg-test-node hce-agent[2163]:
Dec 12 12:34:39 xg-test-node hce-agent[2163]:
Dec 12 12:34:39 xg-test-node hce-agent[2163]:
Dec 12 12:34:39 xg-test-node hce-agent[2163]: hce-agent 1.0.0
Dec 12 12:34:39 xg-test-node systemd[1]: hce-agent.service: Deactivated successfully.
Dec 12 12:34:39 xg-test-node systemd[1]: Stopped Huawei Cloud HCE Agent.
```

#### 3. 网络未打通

登录服务器节点，执行如下命令（使用当前区域的region\_code替换\${region\_code}）：

```
curl https://hceos-agent.${region_code}.myhuaweicloud.com
```

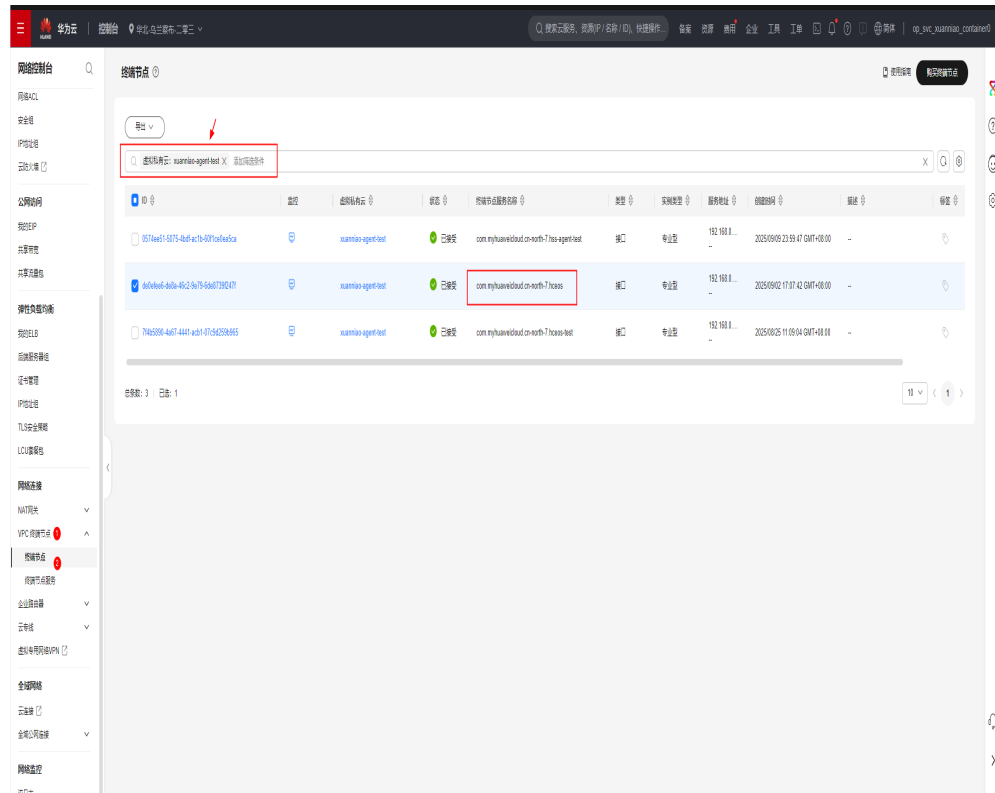
如果如下图显示无法解析域名，则需要购买VPC终端节点打通网络。

```
[root@xg-test-node hce-agent]# curl https://hceos-agent.cn-north-4.myhuaweicloud.com
curl: (6) Could not resolve host: hceos-agent.cn-north-4.myhuaweicloud.com
[root@xg-test-node hce-agent]# ^C
```

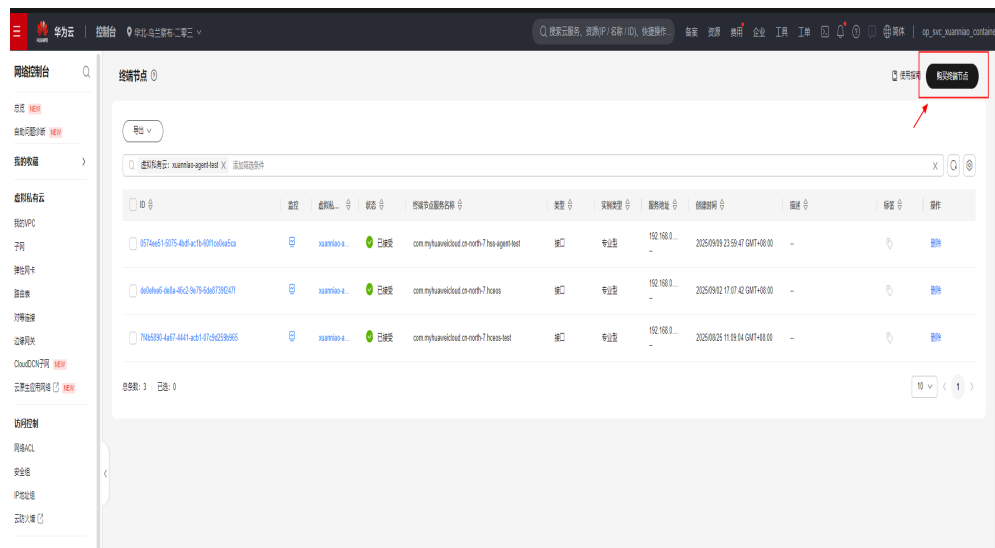
### 网络打通流程

**步骤1** 使用拥有VPC Endpoint Administrator角色或者VPC Endpoint FullAccess权限的账号登录**VPC控制台**。

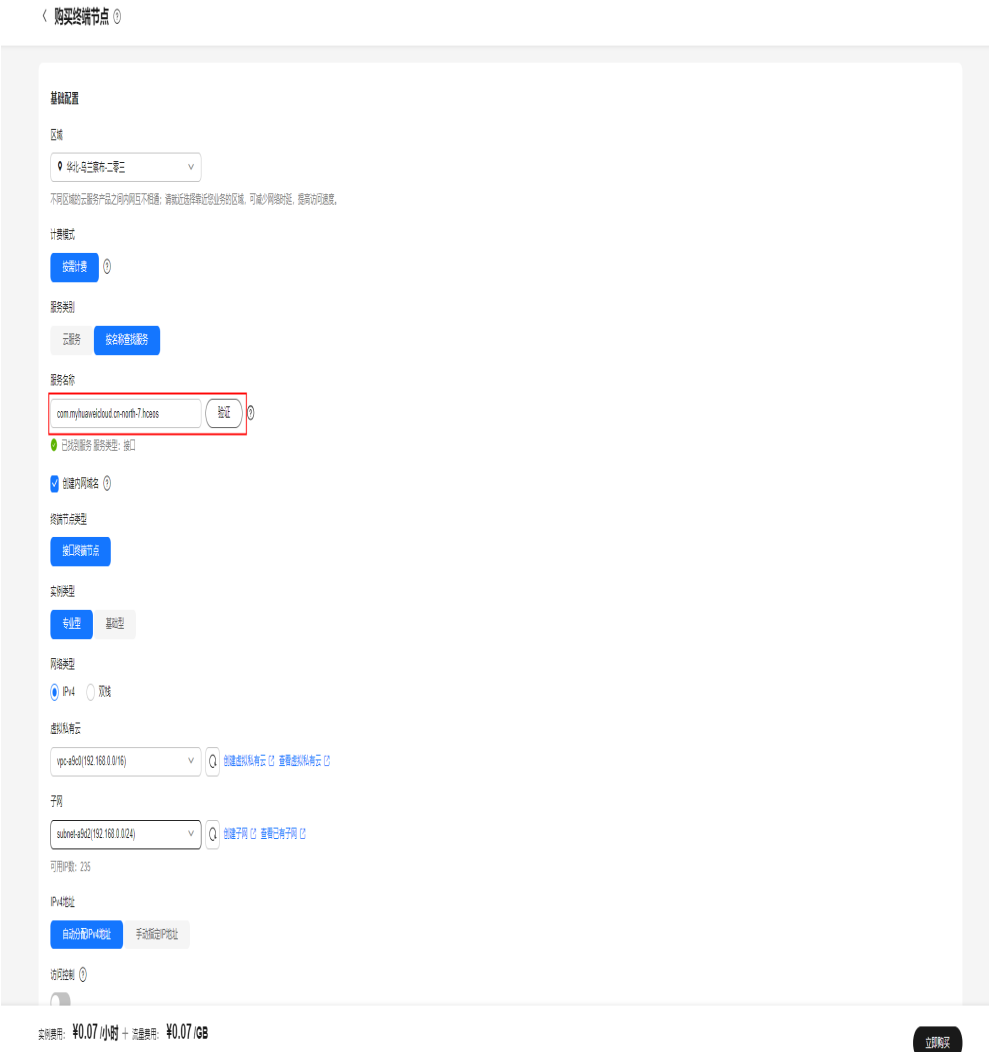
**步骤2** 选择“终端节点”，在上方搜索栏按照虚拟私有云名称筛选节点对应的虚拟私有云，检查是否存在名称为“com.myhuaweicloud.\${region\_code}.hceos”的终端节点服务，若不存在，则执行**步骤3**，否则跳过至**步骤4**。



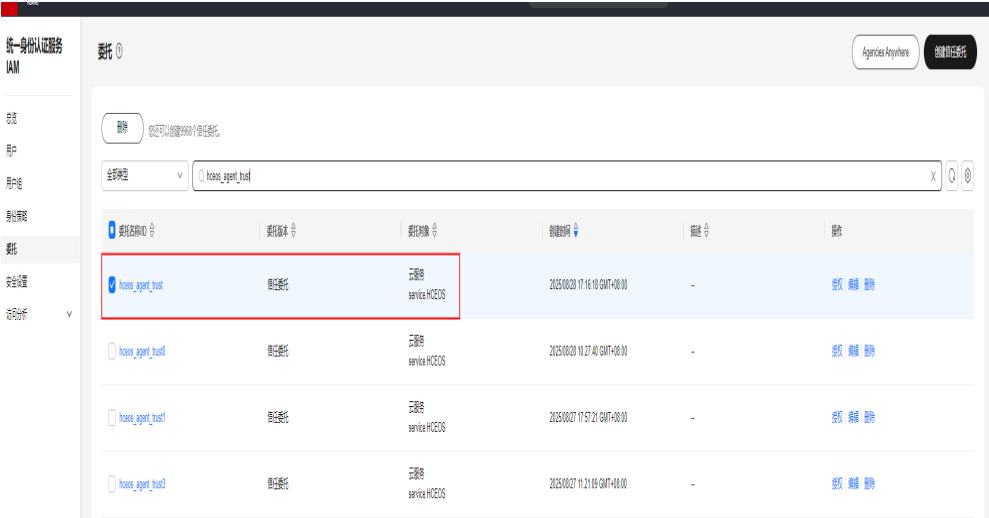
**步骤3** 单击页面右上方的“购买终端节点”按钮，选择按名称查找服务，输入 `com.myhuaweicloud.${region_code}.hceos`，并使用当前区域的 `region_code` 替换 `${region_code}`，单击“验证”，并按需选择子网、ipv4地址等选项，最后单击“立即购买”。购买完成后等待几分钟，若节点插件状态已在线，则问题已修复，结束步骤，否则继续执行**步骤4**。

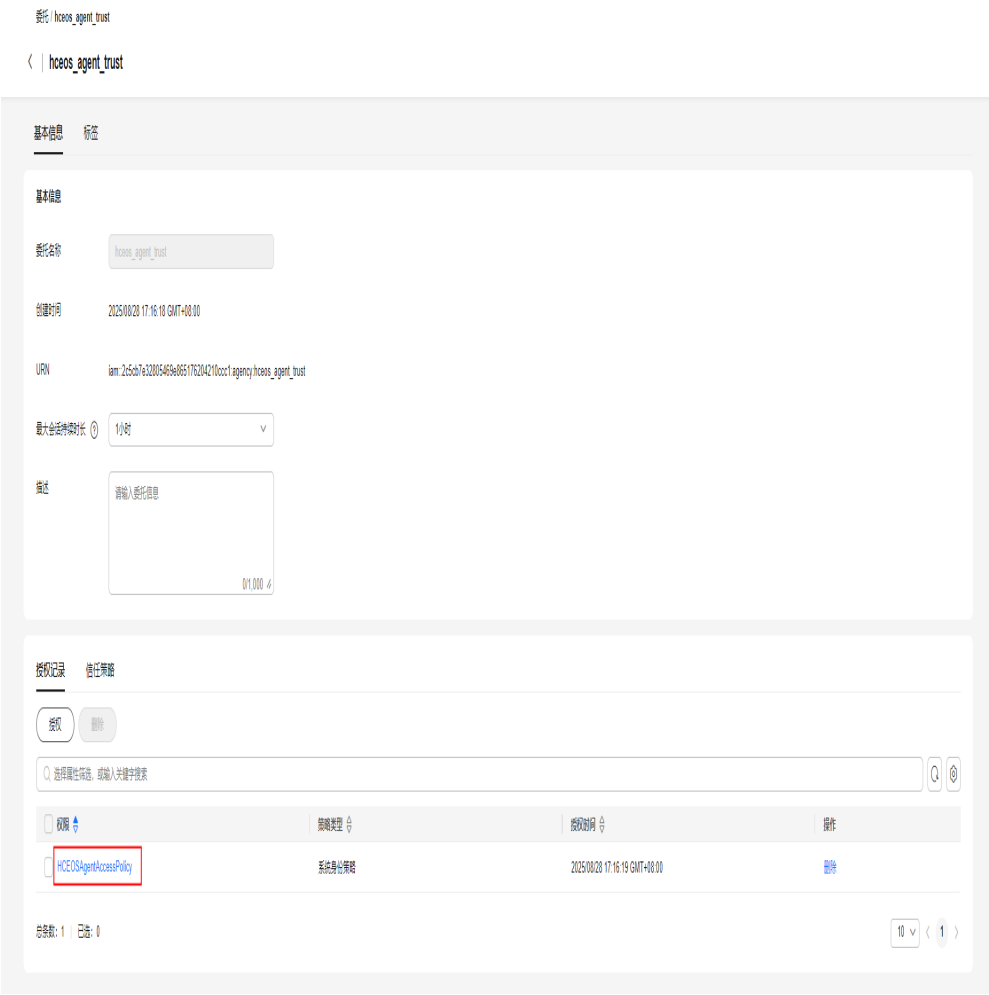




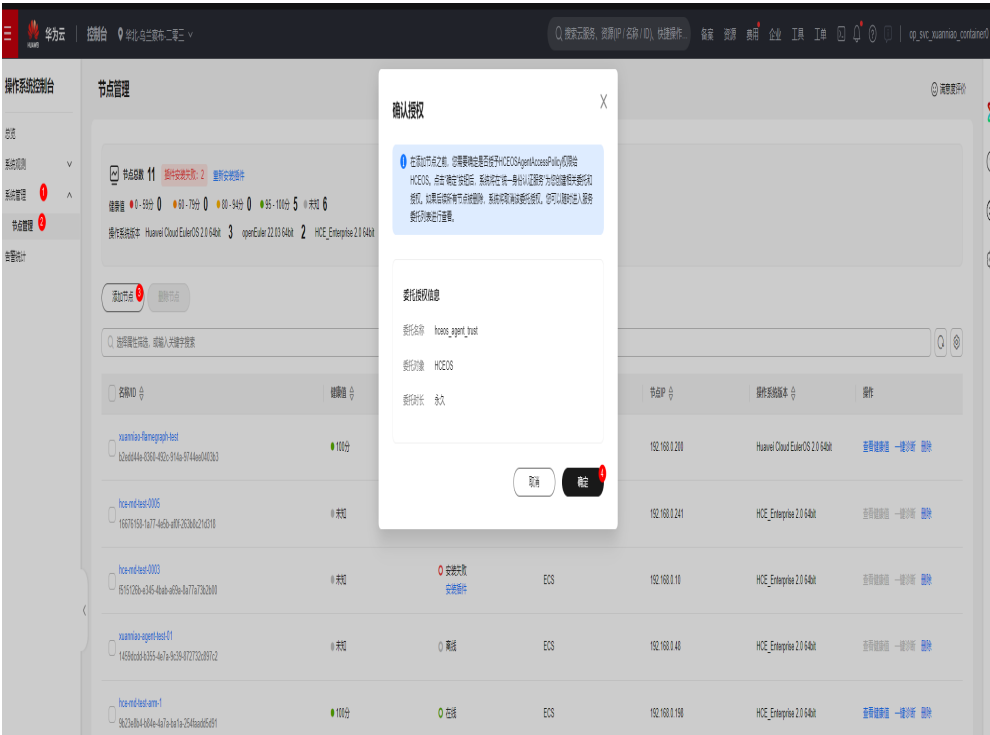


**步骤4** 登录IAM控制台，进入统一身份认证服务，在左侧导航栏选择委托，查看是否存在名称为“hceos\_agent\_trust”，委托对象为“service.HCEOS”的信任委托，若不存在则跳过至**步骤5**，若存在，单击“hceos\_agent\_trust”，单击“授权记录”，查看是否存在“HCEOSAgentAccessPolicy”系统身份策略，若不存在，则执行**步骤5**。





**步骤5** 进入[HCE控制台](#)，在左侧导航栏选择系统管理，进一步选择节点管理，单击“添加节点”按钮，弹窗中单击确定，并在节点执行“systemctl restart hce-agent”。



---结束

# 28 IMA 度量机制占用内存过大问题说明

---

## 现象描述

系统slab内存占用持续增加，导致产品部分服务无法正常启动。通过执行slabtop命令，可见kmalloc占用了大量内存，并且这部分内存仍然在持续增加。

## 问题原因

IMA度量特性可能在特定的策略配置下带来大量内核内存的占用，从机制层面来说，IMA特性始终将度量记录存储在内存中，这种开源机制设计目的是为了保持度量结果的可信。

## 解决方案

修改cmdline，移除integrity=1 ima\_policy=tcb参数。

# 29 如何解决 curl 命令报 out of memory?

## 现象描述

curl命令在使用--data-binary参数传输大文件 / 大数据时，由于该参数会在发起请求前将指定的全部数据完整加载到内存中，当数据超过一定大小时触发 out of memory。

示例: curl -sS -L -w '%{http\_code}' -X PUT --data-binary '@/root/example.tar.gz' -H 'Content-Type: application/octet-stream' 'https://example.com/example'

显示:

curl: option --data-binary: out of memory

curl: try 'curl --help' or 'curl --manual' for more information

## 问题原因

在curl 7.73版本中新增bugfix “[curl: make file2memory use dynbuf](#)” 用于将更多的自定义内存重分配功能切换到使用动态缓冲区（dynbuf），以实现统一处理并提高可靠性，在该版本中引入MAX\_FILE2MEMORY并定义为1G，当采用--data-binary传输时，如果读取的数据大小超过1G，则报out of memory错误，在curl 8.10.0版本中设置为16G。

引入此次修改的主要原因在于原有逻辑未对读取到内存的文件大小设置明确上限，当用户通过 --data-binary 提交超大文件（如数 GB 的视频、压缩包）时，curl 会无限制地尝试将文件全部加载到内存，存在系统内存耗尽（OOM）风险。此类场景下，不仅 curl 进程会崩溃，还可能抢占系统核心资源，影响其他服务的运行，尤其在服务器端批量调用 curl 的场景中，风险被进一步放大。

## 解决方案

传输大文件优先使用-T参数（流式上传，逐块读取文件并发送，无需全量加载到内存）

示例: curl -sS -L -w '%{http\_code}' -T '/root/example.tar.gz' -H 'Content-Type: application/octet-stream' 'https://example.com/example'