虚拟网卡 bond0 配置手册

文档属性

客户名称		文件类别	技术文档
文件名	虚拟网卡bond0配置手册	是否保密	是
编制者	赵寒冰	编制者职位	售后工程师
编制者邮箱	zhaohanbing@kylinos.cn	编制日期	2021-8-2
版本修订记录			
版本号	修订时间	修订说明	
V1.0	2021-8-2	新建	

目录

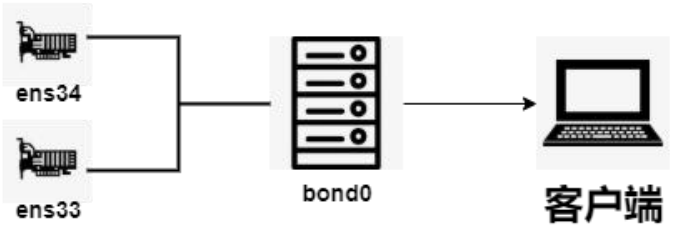
一、文档说明

Bond 就是将多块网卡虚拟成为一块网卡的技术，通过 bond 技术让多块网卡看起来是一个单独的以太网接口设备并具有相同的 IP 地址。

Bond 的原理是网卡在混杂 (promisc) 模式下运行；在这种模式下，网卡不像在通常情况下，只接收目的硬件地址是自身 Mac 的以太网帧，对于别的数据帧都滤掉，以减轻驱动程序的负担；而是接收网络上所有的数据帧，而且修改了驱动程序中的 mac 地址，将两块网卡的 Mac 地址改成相同，可以接收特定 mac 的数据帧，然后把相应的数据帧传送给 bond 驱动程序处理。

网卡 bond 后的主要工作模式有两种：主备的工作方式和负载均衡方式。

双网卡 bond 拓扑图



在主备模式下 ，只有主网卡 eth0 工作，eth1 作为备份网卡是不工作的，只有当一个网络接口失效时（ 例如主交换机掉电等 ），为了不会出现网络中断，系统会按照配置指定的网卡顺序启动工作，保证机器仍能对外服务，起到了失效保护的功能。

在负载均衡工作模式下，由于两块网卡都正常工作，它能提供两倍的带宽，在这种情况下出现一块网卡失效，仅仅会是服务器出口带宽下降，也不会影响网络使用。

二、背景

基于银河麒麟 V10（SP1）完成麒麟软件-虚拟网卡 bond0 配置手册。

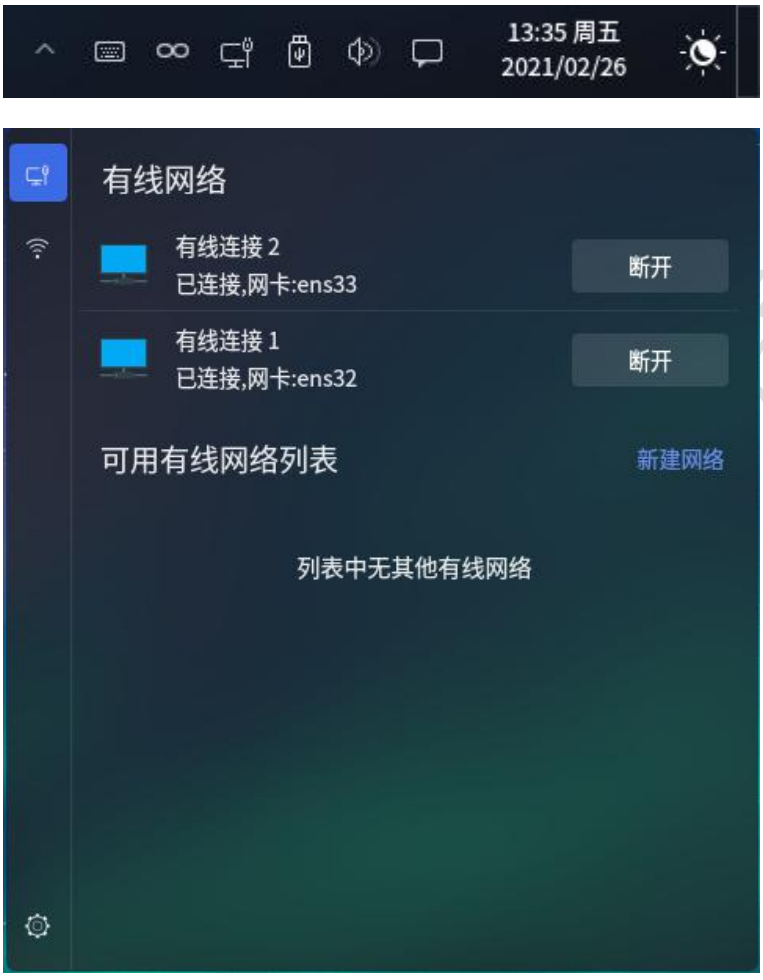
三、方案

基于银河麒麟 V10（SP1）配置 bond 服务，分为两种方式：图形化配置及命令行配置。

图形化配置方式：

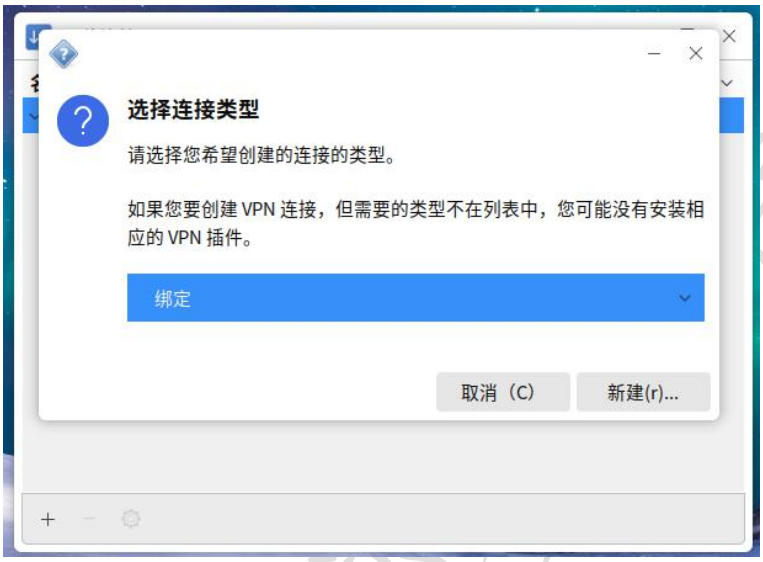
以双网卡 ens32、ens33，模式 **热备** 为例

- 点击桌面右下角 **网络工具** 图标，然后点左下角齿轮图标“**设置网络**”

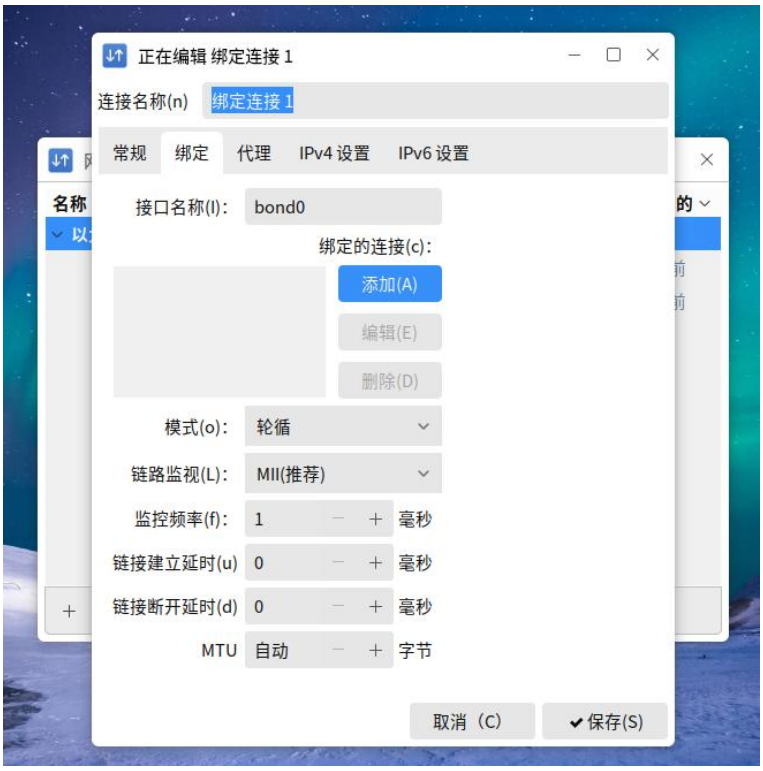


- 在弹出窗口点击右下角＋号**增加一个新的链接**

- 然后选择连接类型为 **绑定**



- 点击上图 **新建** 之后，即可进入 bond 编辑界面进行 bond 配置



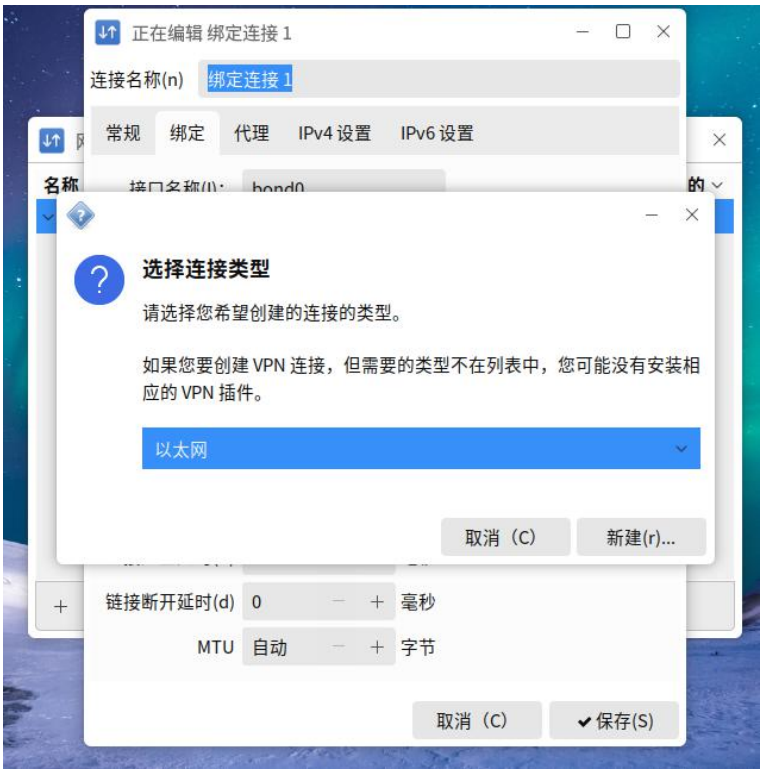
例如：

在“**绑定**”项进行添加 slave 网卡和配置模式等；

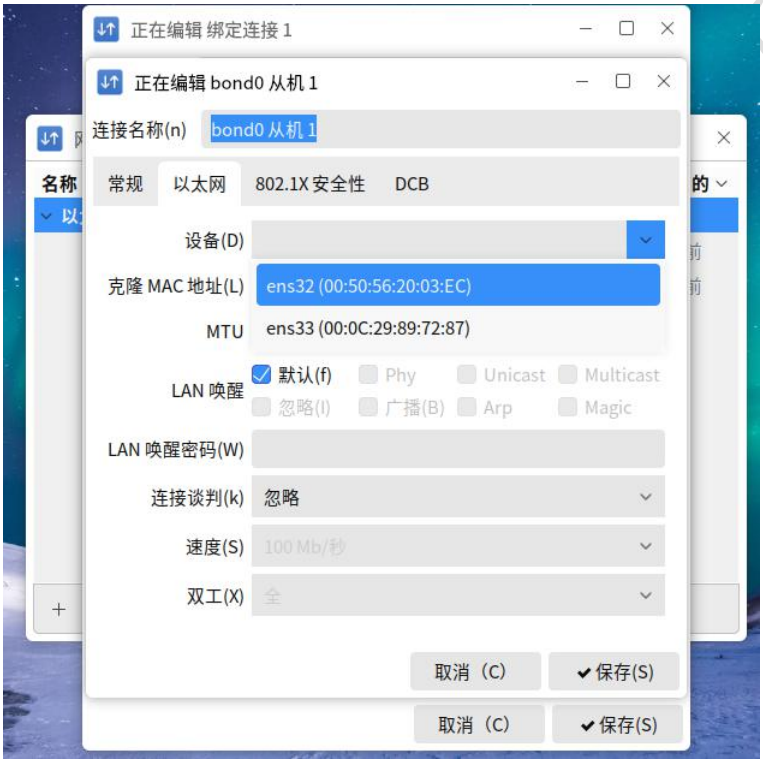
在“IPV4 设置”进行 ip 配置

添加网卡：

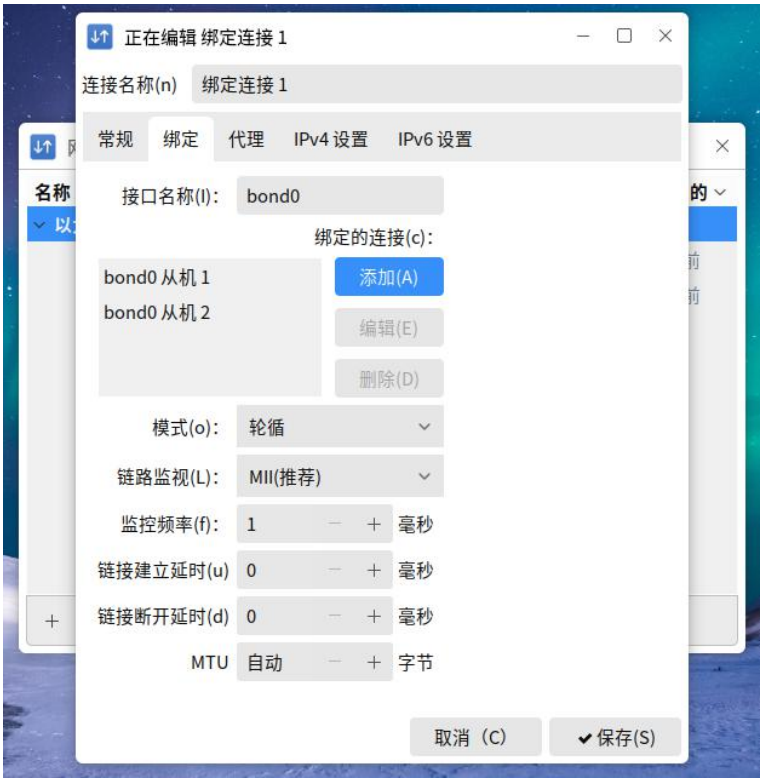
点击“**绑定的连接**”下方的“添加”，在弹出框新建类型为“**以太网**”的连接



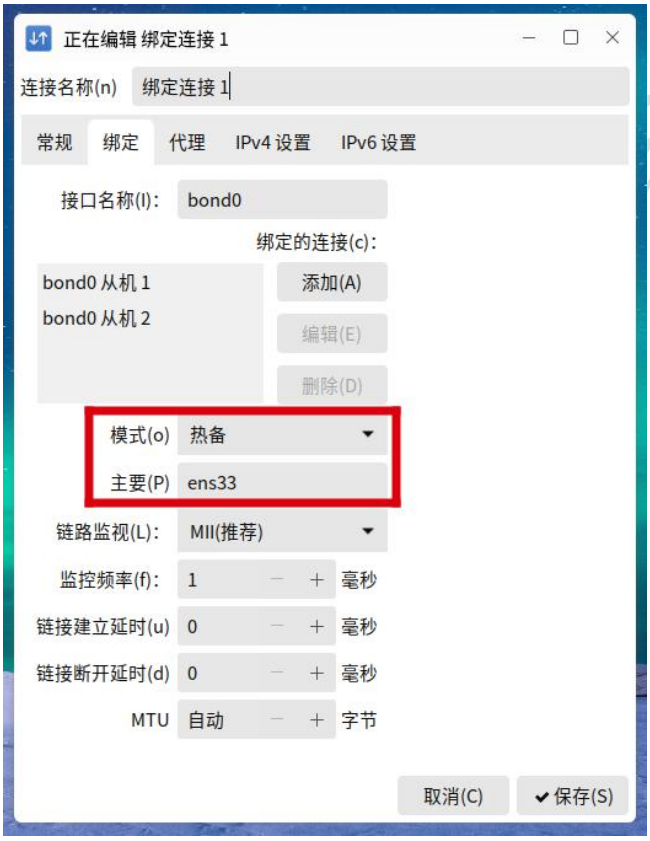
在”设备“处选择 ens32 并，按需配置后点击保存即可。



ens33 的添加与 ens32 同理，之后如下图

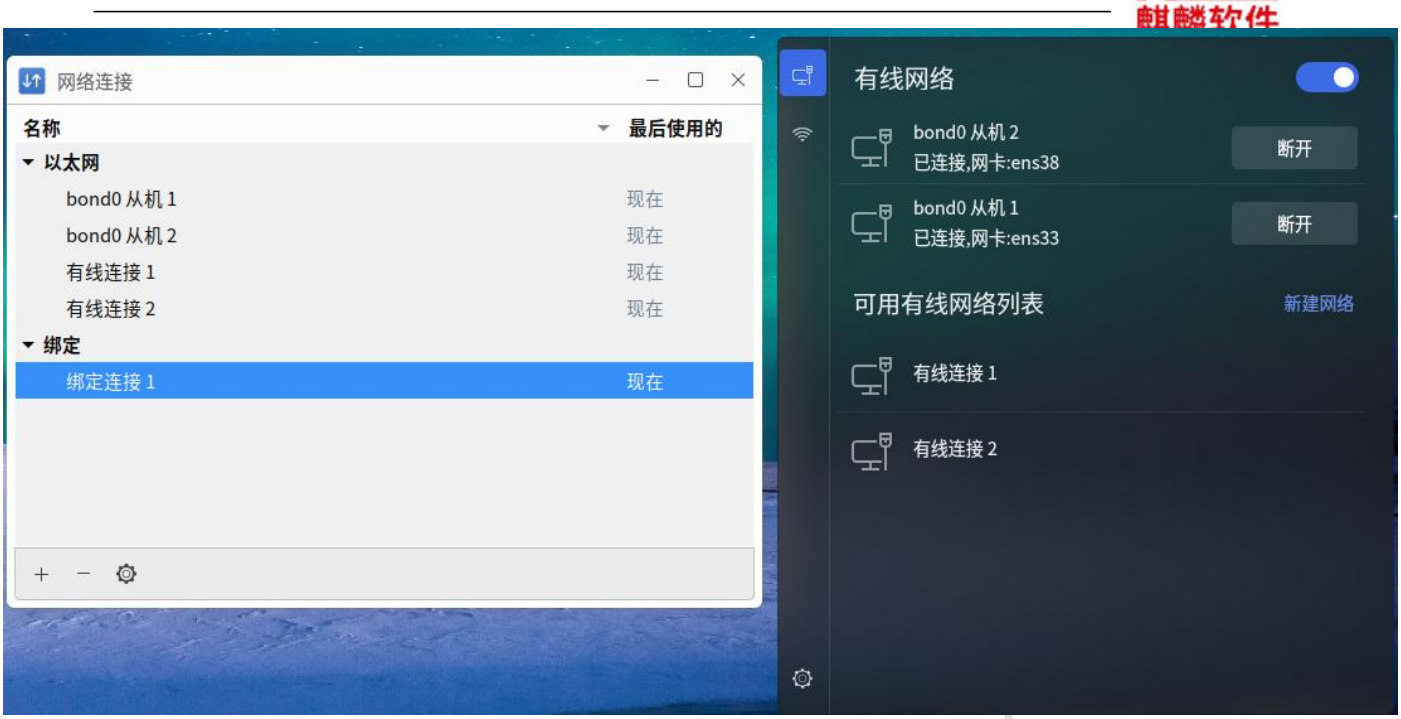


在模式中选择所需模式，如：默认为 **轮循**，可改为 **热备** （需配置主要接口名称）



其他项按需配置即可

- 配置完成后，点击保存，查看网络连接状态



命令行配置方式：

- bond 的下载部署

#ifenslave 为一种粘合和分离式的软件，可以将数据包有效的分配到 bonding 驱动。

apt-get install ifenslave -y

若未找到软件包可先更新

apt update

- 加载 bonding 的内核模块

modprobe bonding

lsmod | grep bonding

- 在开机时启动 bonding 模块

在 /etc/modules 下 增加 bonding

vi /etc/modules

bonding

或者：

echo "bonding" >> /etc/modules

```
root@kylin-VMware-Virtual-Platform:/home/kylin/桌面# lsmod | grep bonding
bonding               167936  0
root@kylin-VMware-Virtual-Platform:/home/kylin/桌面# echo "bonding" >> /etc/modules
root@kylin-VMware-Virtual-Platform:/home/kylin/桌面# cat /etc/modules
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.

bonding
```

- **配置文件新增**

vim /etc/network/interfaces

auto ens33

iface ens33 inet manual

#主网卡为 ens33

bond-master bond0

bond-primary ens33

auto ens38

iface ens38 inet manual

bond-master bond0

#绑定的网卡名称

auto bond0

#采用静态地址

iface bond0 inet static

#IP 地址

address xxx.xx.xxx.xxx

#网关

gateway xxx.xx.xxx.xxx

#掩码

netmask 255.255.255.0

#主备模式

bond-mode active-backup

#miimon 是 100 毫秒监测一次网卡状态

bond-miimon 100

bond-slaves ens33 ens38

- 自适应负载均衡(仅 mode6 需要)

vi /etc/modprobe.d/bond.conf

alias bond0 bonding options bond0 mode=balance-alb miimon=100

- 重启网络

  sudo service network-manager restart

  sudo systemctl restart networking

- #查看状态

1. systemctl status networking.service / network-manager

```
kylin@kylin-VMware-Virtual-Platform:~/桌面$ systemctl status networking.service
● networking.service - Raise network interfaces
     Loaded: loaded (/lib/systemd/system/networking.service; enabled; vendor preset: enabled)
     Active: active (exited) since Tue 2021-08-03 09:33:15 CST; 8min ago
       Docs: man:interfaces(5)
    Process: 6872 ExecStart=/sbin/ifup -a --read-environment (code=exited, status=0/SUCCESS)
   Main PID: 6872 (code=exited, status=0/SUCCESS)

8月 03 09:33:15 kylin-VMware-Virtual-Platform systemd[1]: networking.service: Succeeded.
8月 03 09:33:15 kylin-VMware-Virtual-Platform systemd[1]: Stopped Raise network interfaces.
8月 03 09:33:15 kylin-VMware-Virtual-Platform systemd[1]: Starting Raise network interfaces...
8月 03 09:33:15 kylin-VMware-Virtual-Platform systemd[1]: Finished Raise network interfaces.
```

2. ip addr / ifconfig

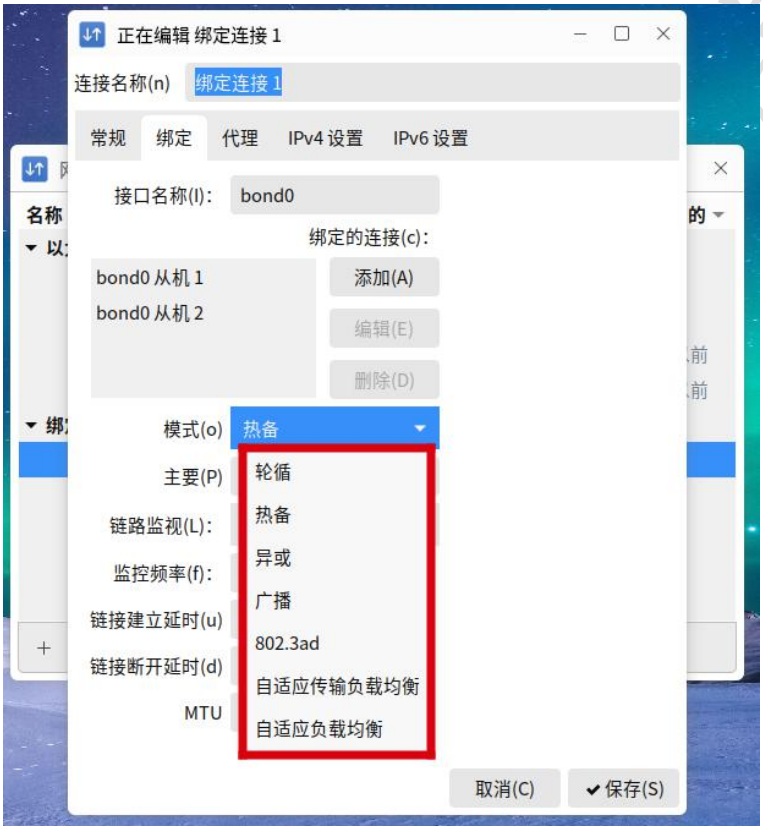
```
kylin@kylin-VMware-Virtual-Platform:~/桌面$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master bond0 state UP group default qlen 1000
    link/ether 00:0c:29:90:af:4e brd ff:ff:ff:ff:ff:ff
3: ens38: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc fq_codel master bond0 state UP group default qlen 1000
    link/ether 00:0c:29:90:af:4e brd ff:ff:ff:ff:ff:ff
4: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:0c:29:90:af:4e brd ff:ff:ff:ff:ff:ff
    inet 192.168.218.133/24 brd 192.168.218.255 scope global dynamic noprefixroute bond0
       valid_lft 1121sec preferred_lft 1121sec
    inet6 fe80::2b78:7bc:4b5f:87af/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

3. cat /proc/net/bonding/bond0

四、bond 模式说明

bond 生效后，bond0，eth33，eth38 的 mac 地址均一致。

另外，网卡绑定有多种模式，可根据需求修改，说明如下：



- 第一种模式：mod=0，即：(balance-rr) Round-robin policy（平衡轮循环策略）

特点：传输数据包顺序是依次传输（即：第 1 个包走 eth0，下一个包就走 eth1….一直循环下去，直到最后一个传输完毕），此模式提供负载平衡和容错能力；但是我们知道如果一个连接或者会话的

数据包从不同的接口发出的话，中途再经过不同的链路，在客户端很有可能会出现数据包无序到达的问题，而无序到达的数据包需要重新要求被发送，这样网络的吞吐量就会下降。

- **第二种模式：mod=1，即： (active-backup) Active-backup policy（主-备份策略）**

特点：只有一个设备处于活动状态，当一个宕掉另一个马上由备份转换为主设备。mac 地址是外部可见得，从外面看来，bond 的 MAC 地址是唯一的，以避免 switch(交换机)发生混乱。此模式只提供了容错能力；由此可见此算法的优点是可以提供高网络连接的可用性，但是它的资源利用率较低，只有一个接口处于工作状态，在有 N 个网络接口的情况下，资源利用率为 1/N。

- **第三种模式：mod=2，即：(balance-xor) XOR policy（平衡策略）**

特点：基于指定的传输 HASH 策略传输数据包。缺省的策略是：(源 MAC 地址 XOR 目标 MAC 地址) % slave 数量。其他的传输策略可以通过 xmit_hash_policy 选项指定，此模式提供负载平衡和容错能力。

- **第四种模式：mod=3，即：broadcast（广播策略）**

特点：在每个 slave 接口上传输每个数据包，此模式提供了容错能力。

- **第五种模式：mod=4，即：(802.3ad) IEEE 802.3ad Dynamic link aggregation（IEEE 802.3ad 动态链接聚合）**

特点：创建一个聚合组，它们共享同样的速率和双工设定。根据 802.3ad 规范将多个 slave 工作在同一个激活的聚合体下。

外出流量的 slave 选举是基于传输 hash 策略，该策略可以通过 xmit_hash_policy 选项从缺省的 XOR 策略改变到其他策略。需要注意的 是，并不是所有的传输策略都是 802.3ad 适应的，尤其考虑到在 802.3ad 标准 43.2.4 章节提及的包乱序问题。不同的实现可能会有不同的适应性。

必要条件：

条件 1：ethtool 支持获取每个 slave 的速率和双工设定

条件 2：switch(交换机)支持 IEEE 802.3ad Dynamic link aggregation

条件 3：大多数 switch(交换机)需要经过特定配置才能支持 802.3ad 模式

- **第六种模式：mod=5，即：(balance-tlb) Adaptive transmit load balancing（适配器传输负载均衡）**

特点：不需要任何特别的 switch(交换机)支持的通道 bonding。在每个 slave 上根据当前的负载（根据速度计算）分配外出流量。如果正在接受数据的 slave 出故障了，另一个 slave 接管失败的 slave 的 MAC 地址。

该模式的必要条件：ethtool 支持获取每个 slave 的速率；

- **第七种模式：mod=6，即：(balance-alb) Adaptive load balancing（适配器适应性负载均衡）**

特点：该模式包含了 balance-tlb 模式，同时加上针对 IPV4 流量的接收负载均衡(receive load balance, rlb)，而且不需要任何 switch(交换机)的支持。接收负载均衡是通过 ARP 协商实现的。bonding 驱动截获本机发送的 ARP 应答，并把源硬件地址改写为 bond 中某个 slave 的唯一硬件地址，从而使得不同的对端使用不同的硬件地址进行通信。

来自服务器端的接收流量也会被均衡。当本机发送 ARP 请求时，bonding 驱动把对端的 IP 信息从 ARP 包中复制并保存下来。当 ARP 应答从对端到达 时，bonding 驱动把它的硬件地址提取出来，并发起一个 ARP 应答给 bond 中的某个 slave。使用 ARP 协商进行负载均衡的一个问题是：每次广播 ARP 请求时都会使用 bond 的硬件地址，因此对端学习到这个硬件地址后，接收流量将会全部流向当前的 slave。这个问题可以通过给所有的对端发送更新 （ARP 应答）来解决，应答中包含他们独一无二的硬件地址，从而导致流量重新分布。当新的 slave 加入到 bond 中时，或者某个未激活的 slave 重新 激活时，接收流量也要重新分布。接收的负载被顺序地分布（round robin）在 bond 中最高速的 slave 上。

当某个链路被重新接上，或者一个新的 slave 加入到 bond 中，接收流量在所有当前激活的 slave 中全部重新分配，通过使用指定的 MAC 地址给每个 client 发起 ARP 应答。下面介绍的 updelay 参数必须被设置为某个大于等于 switch(交换机)转发延时的值，从而保证发往对端的 ARP 应答 不会被 switch(交换机)阻截。

必要条件：

条件 1：ethtool 支持获取每个 slave 的速率；

条件 2：底层驱动支持设置某个设备的硬件地址，从而使得总是有个 slave(curr_active_slave)使用 bond 的硬件地址，同时保证每个 bond 中的 slave 都有一个唯一的硬件地址。如果 curr_active_slave 出故障，它的硬件地址将会被新选出来的 curr_active_slave 接管。

其实 mod=6 与 mod=0 的区别：mod=6，先把 eth0 流量占满，再占 eth1，….ethX；而 mod=0 的话，会发现 2 个口的流量都很稳定，基本一样的带宽。而 mod=6，会发现第一个口流量很高，第 2 个口只占了小部分流量。