# 七种网卡绑定模式详解

2017-07-07 15:10 | 阅读 4.7k | 评论 0



**概览:**

目前网卡绑定mode共有七种(0~6)bond0、bond1、bond2、bond3、bond4、bond5、bond6

**常用的有三种:**

mode=0：平衡负载模式，有自动备援，但需要”Switch”支援及设定。

mode=1：自动备援模式，其中一条线若断线，其他线路将会自动备援。

mode=6：平衡负载模式，有自动备援，不必”Switch”支援及设定。

**说明:**

需要说明的是如果想做成mode 0的负载均衡,仅仅设置这里optionsbond0 miimon=100 mode=0是不够的,与网卡相连的交换机必须做特殊配置（这两个端口应该采取聚合方式），因为做bonding的这两块网卡是使用同一个MAC地址.从原理分析一下（bond运行在mode0下）：

mode 0下bond所绑定的网卡的IP都被修改成相同的mac地址，如果这些网卡都被接在同一个交换机，那么交换机的arp表里这个mac地址对应的端口就有多 个，那么交换机接受到发往这个mac地址的包应该往哪个端口转发呢？正常情况下mac地址是全球唯一的，一个mac地址对应多个端口肯定使交换机迷惑了。所以 mode0下的bond如果连接到交换机，交换机这几个端口应该采取聚合方式（cisco称为 ethernetchannel，foundry称为portgroup），因为交换机做了聚合后，聚合下的几个端口也被捆绑成一个mac地址.我们的解 决办法是，两个网卡接入不同的交换机即可。

mode6模式下无需配置交换机，因为做bonding的这两块网卡是使用不同的MAC地址。

**七种bond模式说明:**

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

**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（IEEE802.3ad 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IEEE802.3ad 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(receiveload 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（roundrobin）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

**Linux网口绑定:**

通过网口绑定(bond)技术,可以很容易实现网口冗余，负载均衡，从而达到高可用高可靠的目的。前提约定:

**2个物理网口分别是：eth0,eth1**

**绑定后的虚拟口是：bond0**

**服务器IP是：10.10.10.1**

**第一步，配置设定文件:**

```
[root@woo ~]# vi  /etc/sysconfig/network-scripts/ifcfg-bond0

DEVICE=bond0

BOOTPROTO=none

ONBOOT=yes

IPADDR=10.10.10.1
```

> NETMASK=255.255.255.0
>
> NETWORK=192.168.0.0
>
> [root@woo ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth0
>
> DEVICE=eth0
>
> BOOTPROTO=none
>
> MASTER=bond0
>
> SLAVE=yes
>
> [root@woo ~]# vi /etc/sysconfig/network-scripts/ifcfg-eth1
>
> DEVICE=eth1
>
> BOOTPROTO=none
>
> MASTER=bond0
>
> SLAVE=yes

### 第二步，修改modprobe相关设定文件，并加载bonding模块：

> 1.在这里，我们直接创建一个加载bonding的专属设定文件/etc/modprobe.d/bonding.conf
>
> [root@woo ~]# vi /etc/modprobe.d/bonding.conf
>
> alias bond0 bonding
>
> options bonding mode=0 miimon=200
>
> 2.加载模块(重启系统后就不用手动再加载了)
>
> [root@woo ~]# modprobe bonding
>
> 3.确认模块是否加载成功：
>
> [root@woo ~]# lsmod | grep bonding
>
> bonding 100065 0

### 第三步，重启一下网络，然后确认一下状况：

> [root@db01 ~]# service network restart
>
> Shutting down interface bond0: [ OK ]

```
Shutting down loopback interface:  [  OK  ]

Bringing up loopback interface:  [  OK  ]

Bringing up interface bond0:  [  OK  ]

[root@db01 ~]#  cat /proc/net/bonding/bond0

Ethernet Channel Bonding Driver: v3.4.0-1 (October 7, 2008)

Bonding Mode: fault-tolerance (active-backup)

Primary Slave: None

Currently Active Slave: eth0

MII Status: up

MII Polling Interval (ms): 100

Up Delay (ms): 0

Down Delay (ms): 0

Slave Interface: eth0

MII Status: up

Speed: 1000 Mbps

Duplex: full

Link Failure Count: 0

Permanent HW addr: 40:f2:e9:db:c9:c2

Slave Interface: eth1

MII Status: up

Speed: 1000 Mbps

Duplex: full

Link Failure Count: 0

Permanent HW addr: 40:f2:e9:db:c9:c3

[root@db01 ~]#  ifconfig | grep HWaddr
```

```
bond0     Link encap:Ethernet  HWaddr 40:F2:E9:DB:C9:C2

eth0      Link encap:Ethernet  HWaddr 40:F2:E9:DB:C9:C2

eth1      Link encap:Ethernet  HWaddr 40:F2:E9:DB:C9:C2
```

从上面的确认信息中，我们可以看到3个重要信息：

1.现在的bonding模式是active-backup

2.现在Active状态的网口是eth0

3.bond0,eth1的物理地址和处于active状态下的eth0的物理地址相同，这样是为了避免上位交换机发生混乱。

任意拔掉一根网线，然后再访问你的服务器，看网络是否还是通的。

**第四步，系统启动自动绑定、增加默认网关：**

```
[root@woo ~]# vi /etc/rc.d/rc.local

#追加

ifenslave bond0 eth0 eth1

route add default gw 10.10.10.1
```

#如可上网就不用增加路由，0.1地址按环境修改.

————————————————————————————————

留心：前面只是2个网口绑定成一个bond0的情况，如果我们要设置多个bond口，比如物理网口eth0和eth1组成bond0，eth2和eth3组成bond1，

**多网口绑定：**

那么网口设置文件的设置方法和上面第1步讲的方法相同，只是/etc/modprobe.d/bonding.conf的设定就不能像下面这样简单的叠加了：

```
alias bond0 bonding

options bonding mode=1 miimon=200

alias bond1 bonding

options bonding mode=1 miimon=200
```

**正确的设置方法有2种：**

**第一种,你可以看到，这种方式的话，多个bond口的模式就只能设成相同的了：**

> alias bond0 bonding
>
> alias bond1 bonding
>
> options bonding max_bonds=2 miimon=200 mode=1
>
> 

**第二种，这种方式，不同的bond口的mode可以设成不一样：**

> alias bond0 bonding
>
> options bond0 miimon=100 mode=1
>
> install bond1 /sbin/modprobe bonding -o bond1 miimon=200 mode=0
>
> 

仔细看看上面这2种设置方法，现在如果是要设置3个，4个，甚至更多的bond口，你应该也会了吧！

**后记：**

简单的介绍一下上面在加载bonding模块的时候，options里的一些参数的含义：

miimon 监视网络链接的频度，单位是毫秒，我们设置的是200毫秒。

max_bonds 配置的bond口个数

mode bond模式，主要有以下几种，在一般的实际应用中，0和1用的比较多。

文章源自微信公众号： DevOps