**Linux Bonding**

**一、什么是bonding**Linux bonding 驱动提供了一个把多个网络接口设备捆绑为单个的网络接口设置来使用，用于网络负载均衡及网络冗余  
  
  
**二、bonding应用方向**  
  
**1、网络负载均衡**  
  
对于bonding的网络负载均衡是我们在文件服务器中常用到的，比如把三块网卡，当做一块来用，解决一个IP地址，流量过大，服务器网络压力过大的问题。对于文件服务器来说，比如NFS或SAMBA文件服务器，没有任何一个管理员会把内部网的文件服务器的IP地址弄很多个来解决网络负载的问题。如果在内网中，文件服务器为了管理和应用上的方便，大多是用同一个IP地址。对于一个百M的本地网络来说，文件服务器在多 个用户同时使用的情况下，网络压力是极大的，特别是SAMABA和NFS服务器。为了解决同一个IP地址，突破流量的限制，毕竟网线和网卡对数据的吞吐量是有限制的。如果在有限的资源的情况下，实现网络负载均衡，最好的办法就是 bonding   
  
**2、网络冗余**  
  
对于服务器来说，网络设备的稳定也是比较重要的，特别是网卡。在生产型的系统中，网卡的可靠性就更为重要了。在生产型的系统中，大多通过硬件设备的冗余来提供服务器的可靠性和安全性，比如电源。bonding 也能为网卡提供冗余的支持。把多块网卡绑定到一个IP地址，当一块网卡发生物理性损坏的情况下，另一块网卡自动启用，并提供正常的服务，即：默认情况下只有一块网卡工作，其它网卡做备份  
  
  
**三、bonding实验环境及配置  
  
1、实验环境**系统为：CentOS，使用4块网卡（eth0、eth1 ==> bond0；eth2、eth3 ==> bond1）来实现bonding技术  
 **2、bonding配置  
  
第一步：先查看一下内核是否已经支持bonding**  
1）如果内核已经把bonding编译进内核，那么要做的就是加载该模块到当前内核；其次查看ifenslave该工具是否也已经编译  
modprobe -l bond\*或者 modinfo bonding  
modprobe bonding  
lsmod | grep 'bonding'  
echo 'modprobe bonding &> /dev/null' >> /etc/rc.local（开机自动加载bonding模块到内核）  
which ifenslave  
注意：默认内核安装完后就已经支持bonding模块了，无需要自己手动编译  
  
2）如果bonding还没有编译进内核，那么要做的就是编译该模块到内核  
（1）编译bonding  
tar -jxvf kernel-XXX.tar.gz  
cd kernel-XXX  
make menuconfig  
选择 " Network device support " -> " Bonding driver support "  
make bzImage  
make modules && make modules\_install  
make install  
（2）编译ifenslave工具  
gcc -Wall -O -I kernel-XXX/include ifenslave.c -o ifenslave  
  
**第二步：主要有两种可选择（**第1种：实现网络负载均衡，第2种：实现网络冗余）  
  
**例1：实现网络冗余（即：mod=1方式，使用eth0与eth1）**  
  
（1）编辑虚拟网络接口配置文件（bond0），并指定网卡IP  
vi /etc/sysconfig/network-scripts/ifcfg-bond0  
DEVICE=bond0  
ONBOOT=yes  
BOOTPROTO=static  
IPADDR=192.168.0.254  
BROADCAST=192.168.0.255  
NETMASK=255.255.255.0  
NETWORK=192.168.0.0  
GATEWAY=192.168.0.1  
USERCTL=no  
TYPE=Ethernet  
注意：建议不要指定MAC地址  
  
vi /etc/sysconfig/network-scripts/ifcfg-eth0  
DEVICE=eth0  
BOOTPROTO=none  
ONBOOT=yes  
USERCTL=no  
MASTER=bond0  
SLAVE=yes  
注意：建议不要指定MAC地址  
  
vi /etc/sysconfig/network-scripts/ifcfg-eth1  
DEVICE=eth1  
BOOTPROTO=none  
ONBOOT=yes  
USERCTL=no  
MASTER=bond0  
SLAVE=yes  
注意：建议不要指定MAC地址  
  
（2）编辑模块载入配置文件（/etc/modprobe.conf），开机自动加载bonding模块到内核  
vi /etc/modprobe.conf

alias bond0 bonding   
options bond0 miimon=100 mode=1

alias net-pf-10 off #关闭ipv6支持  
说明：miimon是用来进行链路监测的。 比如:miimon=100，那么系统每100ms监测一次链路连接状态，如果有一条线路不通就转入另一条线路；mode的值表示工作模式，他共有0，1，2，3，4，5，6六种模式，常用为0，6，1三种，具体后面会介绍   
mode=0，表示load balancing (round-robin)为负载均衡方式，两块网卡都工作，但是与网卡相连的交换必须做特殊配置（ 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  
mode=6，表示load balancing (round-robin)为负载均衡方式，两块网卡都工作，但是该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  
mode=1，表示fault-tolerance (active-backup)提供冗余功能，工作方式是主备的工作方式，也就是说默认情况下只有一块网卡工作,另一块做备份   
注意：bonding只能提供链路监测，即从主机到交换机的链路是否接通。如果只是交换机对外的链路down掉了，而交换机本身并没有故障，那么bonding会认为链路没有问题而继续使用  
  
（4）重启并测试  
第一：由于bonding使用的模式为mod=1（网络冗余），所以eth0、eth1与虚拟的bond0同一个MAC地址  
  
  
注意：对比上面这两个图，可知mode=1模式下，eth0与eth1这两块网卡，只有一块网卡在工作（即：eth0），因为eth1网卡的RX与TX都没有在发生变化  
  
第二：测试，用ping指令ping虚拟网卡设备bond0的IP地址（192.168.0.254），然后禁用eth0设备看一下能够继续ping的通  
  
说明：如上图可得到，断开eth0（上图的右下角），还是可以ping的通的  
  
  
**例2：实现网络负载均衡和网络冗余（即：mod=0方式，使用eth0与eth1）**  
注意：VM中只能做mode=1的实验，其它的工作模式得用真机来实践  
  
跟例1的步骤一样，只需要修改模块载入配置文件（/etc/modprobe.conf），如下：

alias bond0 bonding   
options bond0 miimon=100 **mode=0**

（1）测试如下  
##目前两块网卡都处于连接状态  
root@Web:~# ifconfig | grep 'eth' | awk '{print $1}'  
eth0  
eth1  
##禁用了网卡eth0，用ping指令测试  
  
  
反之，也是一样的！  
  
**例3：实现网络负载均衡和网络冗余（即：mod=6方式，使用eth0与eth1，其中eth0设置为primay）**  
  
跟例1的步骤一样，只需要修改模块载入配置文件（/etc/modprobe.conf），如下：  
alias bond0 bonding   
options bond0 miimon=100 **mode=6**  
  
  
上图可知：mode=6时，eth0与eth1所使用的MAC是不一样的  
  
（1）测试如下  
##目前两块网卡都处于连接状态  
root@Web:~# ifconfig | grep 'eth' | awk '{print $1}'  
eth0  
eth1  
##禁用了网卡eth0，用ping指令测试  
  
  
  
**四、bonding运用的注意事项**1、bonding的模式：0-6，即：7种模式第一种模式：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（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，接收流量将会全部刘翔当前的slave。这个问题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  
  
  
2、bonding驱动选项**Bonding驱动的选项是通过在加载时指定参数来设定的。可以通过insmod或modprobe命令的命令行参数来指定，但通常在/etc/modprobe.conf配置文件中指定，或其他的配置文件中  
  
下面列出可用的bonding驱动参数。如果参数没有指定，驱动会使用缺省参数。刚开始配置bond的时候，建议在一个终端窗口中运行"tail -f /var/log/messages"来观察bonding驱动的错误信息【译注：/var/log/messages一般会打印内核中的调试信息】  
有些参数必须要正确的设定，比如miimon、arp\_interval和arp\_ip\_target，否则在链接故障时会导致严重的网络性能退化。很少的设备不支持miimon，因此没有任何理由不使用它们。  
有些选项不仅支持文本值的设定，出于兼容性的考虑，也支持数值的设定，比如，"mode=802.3ad"和"mode=4"效果是一样的  
  
具体的参数列表：  
1）primay  
指定哪个slave成为主设备（primary device），取值为字符串，如eth0，eth1等。只要指定的设备可用，它将一直是激活的slave。只有在主设备（primary device）断线时才会切换设备。这在希望某个slave设备优先使用的情形下很有用，比如，某个slave设备有更高的吞吐率  
注意： primary选项只对active-backup模式有效  
  
2）updelay  
指定当发现一个链路恢复时，在激活该链路之前的等待时间，以毫秒计算。该选项只对miimon链路侦听有效。updelay应该是miimon值的整数倍，如果不是，它将会被向下取整到最近的整数。缺省值为0  
  
3）arp\_interval  
指定ARP链路监控频率，单位是毫秒(ms)。如果APR监控工作于以太兼容模式（模式0和模式2）下，需要把switch(交换机)配置为在所有链路上均匀的分发网络包。如果switch(交换机)被配置为以XOR方式分发网络包，所有来自ARP目标的应答将会被同一个链路上的其他设备收到，这将会导致其他设备的失败。ARP监控不应该和miimon同时使用。设定为0将禁止ARP监控。缺省值为0  
  
4）arp\_ip\_target  
指定一组IP地址用于ARP监控的目标，它只在arp\_interval > 0时有效。这些IP地址是ARP请求发送的目标，用于判定到目标地址的链路是否工作正常。该设定值为ddd.ddd.ddd.ddd格式。多个IP地址通过逗号分隔。至少指定一个IP地址。最多可以指定16个IP地址。缺省值是没有IP地址  
  
5）downdelay  
指定一个时间，用于在发现链路故障后，等待一段时间然后禁止一个slave，单位是毫秒(ms)。该选项只对miimon监控有效。downdelay值应该是miimon值的整数倍，否则它将会被取整到最接近的整数倍。缺省值为0  
  
6）lacp\_rate  
指定在802.3ad模式下，我们希望的链接对端传输LACPDU包的速率。可能的选项：  
（1）slow 或者 0  
请求对端每30s传输LACPDU  
（2）fast 或者 1  
请求对端每1s传输LACPDU  
（3）缺省值是slow  
  
7）max\_bonds  
为bonding驱动指定创建bonding设备的数量。比如：如果max\_bonds为3，而且bonding驱动还没有加载，那么bond0，bond1，bond2将会被创建。缺省值为1  
  
6）miimon  
指定MII链路监控频率，单位是毫秒(ms)。这将决定驱动检查每个slave链路状态频率  
0表示禁止MII链路监控。100可以作为一个很好的初始参考值。下面的use\_carrier选项将会影响如果检测链路状态。更多的信息可以参考“高可靠性”章节。缺省值为08）mode  
指定bonding的策略。缺省是balance-rr （round robin，循环赛）。可选的mode包括：0，1，2，3，4，5，6 **3、bonding链路监测方法**  
  
官方文档里说有两种针对链路的监测方法（注意：这两种监测不能同时使用）  
  
第一种：miimon（这种方法是最常见的，此方法使用系统的mii-tool命令进行监测）  
  
模块加载设置（/etc/modprobe.conf）：  
# Start of bonding configure  
alias bond0 bonding  
options bond0 miimon=100 mode=1  
注意：使用cat /proc/net/bonding/bond0，可查看Bonding Mode: load balancing (round-robin)状态  
options bond0 miimon=100 mode=0  
注意：使用cat /proc/net/bonding/bond0，可查看Bonding Mode: load balancing ((active-backup))状态   
  
root@Web:~# mii-tool  
eth0: negotiated 100baseTx-HD, link ok  
eth1: negotiated 100baseTx-HD, link ok  
  
缺点：这种方法，只能监测交换机与该网卡之间的链路；如果它们之外的链路的地方断了,而交换机本身没有问题，也就是说你的网卡和交换机之间还是UP状态,它是不会认为网络中断，除非你的网卡是DOWN状态，它才会把链路转到另一块网卡上，就像是拔掉网线一样，或者把交换机端口shutdown一样  
  
第二种：arp（这种方法比较实用，你可以把它看作是arp的ping（二层ping），但是可能会给网关造成一定的压力）  
  
模块加载：  
alias bond0 bonding  
options bond0 arp\_interval=100 arp\_ip\_target=192.168.1.1 mode=active-backup primary=eth0  
解析如下：  
arp\_interval=100，表示arp的检测时间，等同于miimon=100的作用  
arp\_ip\_target=192.168.1.1，表示arp检测的目标IP，必须是同网段的，最好就是网关  
注意：如果使用arp来ping网关不通，那么在/proc/net/bonding/bond0里会一会down，一会up的  
  
优点：使用arp这种方法，如果交换机的上出现问题，网络不通，它就会把链转到另一块网卡上，但是不管是哪种方法，在第一块网卡出现问题，链路转到第二块后，如果第一块恢复正常，链路自己不会恢复的

**Debian下通过Bonding实现双网卡单IP**

Linux双网卡绑定一个IP地址，实质工作就是使用两块网卡虚拟为一块，使用同一个IP地址，是我们能够得到更好的更快的服务。本文介绍了bonding的原理，以及如何在Debian下通过bonding实现双网卡单IP的操作。

Linux双网卡绑定一个IP地址，实质工作就是使用两块网卡虚拟为一块，使用同一个IP地址，是我们能够得到更好的更快的服务。其实这项技术在Sun和Cisco中早已存在，被称为Trunking和Etherchannel技术，在Linux的2.4.x的内核中也采用这这种技术，被称为bonding。本文以[Debian](http://os.51cto.com/art/201102/244746.htm)为例介绍这个技术的实现。**bonding的原理**

在正常情况下，网卡只接收目的硬件地址(MAC Address)是自身Mac的以太网帧，对于别的数据帧都过滤掉，以减轻驱动程序的负担。但是网卡也支持另外一种被称为混杂的模式，可以接收网络上所有的帧，比如说很多用物抓包工具就需要让网卡运行在这个模式下。bonding也运行在这个模式下，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。

**bonding模块工作方式**

bonding有0-6七种模式，常用的工作方式为0、1、6三种：

mode=0，表示load balancing (round-robin)为负载均衡方式，两块网卡都工作，但是与网卡相连的交换必须做特殊配置（ 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  
mode=6，表示load balancing (round-robin)为负载均衡方式，两块网卡都工作，但是该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  
mode=1，表示fault-tolerance (active-backup)提供冗余功能，工作方式是主备的工作方式，也就是说默认情况下只有一块网卡工作,另一块做备份

**bonding配置**

使用如下命令安装bonding。

srv10:~# apt-get install ifenslave

使用如下命令让系统开机自动加载模块bonding。其中miimon参数是用于进行链路监测的。比如：miimon=100，那么系统每100ms监测一次链路连接状态，如果有一条线路不通就转入另一条线路；mode的值表示bonding工作模式。

srv10:~# sh -c "echo bonding mode=6 miimon=100 >> /etc/modules"

修改/etc/network/interfaces文件为如下内容。

auto lo

iface lo inet loopback

auto bond0

iface bond0 inet static#虚拟网卡的TCP/IP配置

address 192.168.159.100

netmask 255.255.255.0

gateway 192.168.159.1

dns-nameservers 192.168.159.1

post-up ifenslave bond0 eth0 eth1

pre-down ifenslave -d bond0 eth0 eth1

如果在安装ifenslave后没有重新启动计算机，必须手动加载bonding模块。

srv10:~# modprobe bonding mode=6 miimon=100

使用如下命令重新启动网卡。

srv10:~# /etc/init.d/networking restart