Linux 以太网卡 Bonding 驱动简介

Thomas Davis (tadavis@blb.gov)
李　峰 (lifeng@telebyte.com.cn)
Willy Tarreau (Willy@meta-x.org)
Constantine Gavrilov (const-g@xpert.com)
Chad N. Tindel (ctindel@ieee.org)
Janice Girouard (girouard@us.ibm.com)

2007年5月21日

目　录

第 一 节

注 意

Bonding 的驱动最早是从 Donald Becker 的 beowulf 补丁这个项目，从内核版本 2.0 开始发展起来的。从它出现后，这个驱动已经发生了一些变动，最早的 Linux 内核和 beowulf 站点上的工具对于现在的内核版本来说，都不可用了。

对于这个驱动新的版本，对于较早版本内核的补丁和用户空间工具的更新，请按这篇文章后面的链接处理。

第 二 节

安 装

2.1 对内核编译时支持 bonding 驱动

对于最新版本的 bonding 驱动，使用内核 2.4.12 或更新的版本(否则你需要对内核打补丁)

使用命令‘`make menuconfig /xconfig/config`‘[1]，并在“Network device support”选项下，选择“Bonding driver support”。最好将驱动配置成内核模块方式，因为只有这样才可以向模块驱动传递参数并且配置多于一个的 bonding 设备。

之后编译和安装内核和内核模块。

第 三 节

取得和安装用户空间工具

这个版本的 bonding 驱动需要更新 ifenslave 程序。最早的内核版本和 beowulf 项目所支持的工具将不再有效的工作。内核版本 2.4.12 和之后的版本将包含最新版本的 ifenslave.c 于 Documentation/network 目录中。对于较早的内核版本，请查阅文章后面相关的链接。

注意!!! 如果你运行 RedHat 7.1 或其后发行的版本，你需要注意，因为/usr/include/linux 这个目录不再一定是 /usr/src/linux/include/linux 这个目录的链接。在这种情况下，如果你编译 ifenslave 将显示成功，但你的 bond 将不工作。在编译时使用 -l 选项将确保你使用:

[1]这里的menuconfig,xconfig,config 三个并列的可选的参数，这是在编译内核时选择不同的方式来选择编译方式。

/usr/src/linux/include/linux/if_bonding.h 这个文件，而不是用:
/usr/include/linux 目录下的文件。

为了安装 ifenslave.c，按下面的方式来操作：

```
#gcc -Wall -Wstrict-prototypes -O -I/usr/src/linux/include \
> ifenslave.c -o ifenslave
#cp ifenslave /sbin/ifenslave
```

第 四 节

配置你的系统

关于模块的参数，请查阅下面的章节。你将在/etc/conf.modules (或 /etc/modules.conf)文件中至少加入下面的一行：

```
alias bond0 bonding
```

使用标准的发布版本技术来定义 bond0 网络接口。如，在当前的 RedHat 发布版本，在 /etc/sysconfig/network-scripts 目录下建立 如下内容：

```
DEVICE=bond0
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
```

(注意，将上面的 192.168.1 换成适应你网络的值)

所有的属于这个 trunk 的接口将有一个 SLAVE 和 MASTER 定义。例如，在 RedHat 中，如果你希望用 eth0 和 eth1 (或其它的网络接口)作为 bonding 接口 bond0 的一部分，配置文件(ifcfg-eth0, ifcfg-eth1,等)将如下所示：

```
DEVICE=eth0
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
```

(在 eth1 配置文件中，并且用 MASTER=bond1 来配置第二个 bonding 接口)

重启你的网络系统或仅仅用管理工具重新启动 bonding 设备(如果允许这样做)，否则，重启系统。(对于 RedHat 发行版本，你可以使用 ‘ifup bond0‘
或 ‘/etc/rc.d/init.d/network restart‘)

如果你使用的发行版本中管理工具不支持配置文件中的 master/slave 提示，你需要按下面的方式来配置 bonding 设备：

```
#/sbin/ifconfig bond0 192.168.1.1 up
#/sbin/ifenslave bond0 eth0
#/sbin/ifenslave bond0 eth1
```

(在 ifconfig 命令中用你的 IP 地址来替换你所使用的网络地址，以及客户的子网掩码)

你可以创建一个脚本来执行这些命令，并将它放在合适的 rc 目录中。

如果你指定所有的网络接口在 bonding 设备装载前被安装，你可以使用有用的模块管理工具功能：在你的 modules.conf 文件中，告知相关信息，并查询 bond0，modprobe 将先安装所有的网络接口：

```
probeall band0 eth0 eth1 bonding
```

请注意不要这个行的后面引用 bond0 自已，否则，modprobe 将在无休目的循环中死去。

第 五 节

模块参数

下面的模块参数可以被传递给 bonding 设备：

```
mode=
```

可以设置的值是 0 (循环策略[2],这是默认设置)和 1 (激活备份策略),和 2 (XOR)。如需更多信息，查看第 9 部分和 HA 小节。

```
miimon=
```

使用整数值作为 MII 链路监视频率(单位为 ms)。默认值为 0 且意味着不使用链路监视功能。如果你想用链路监视功能，100 是一个很好的值。如需更多的信息，查看 HA 相关的小节。

```
downdelay=
```

[2] round robin policy

使用一个整数值，链路将在以 ms 为单位的整数时长失效后，将被发现链路失效。这个值必需是 miimon 值的倍数。默认值是 0，如需更多信息，查看 HA 相关的小节。

```
updelay=
```

在 “link up”状态被发现后整数值的 ms 时间后，来使这个链路正常工作。这个值应是 miimon 的倍数。默认值是 0，如需更多的相关信息，查看 HA 小节的内容。

```
arp_interval=
```

使用整数值做为 arp 监视的频率(单位ms)。默认值为 0 意味着不使用 arp 监视功能。查看 HA 小节以获得更多的信息。这个域的值仅在 active_backup 模式下有效。

```
arp_ip_target=
```

当 arp_interval > 0 时一个 IP 地址被使用。这个值决定 arp 发送到目标时链路的健康状态。用ddd.ddd.ddd.ddd 格式来指定这个值。

如果你需要配置几个 bonding 设备，这个驱动需要安装多次。如，对于两个设备，你在 /etc/conf.modules 必需如下所示：

```
alias bond0 bonding
alias bond1 bonding

options bond0 miimon=100
options bond1 -o bonding1 miimon=100
```

第 六 节

测试配置

你可以使用 ifconfig 来测试你的配置文件和传输策略。如，对于循环策略，你将得到如下的信息：

```
 [root]# /sbin/ifconfig
 bond0  Link encap:Ethernet HWaddr 00:C0:F0:1F:37:B4
inet addr:XXX.XXX.XXX.YYY  Bcast:XXX.XXX.XXX.255  Mask:255.255.252.0
UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
RX packets:7224794 errors:0 dropped:0 overruns:0 frame:0
TX packets:3286647 errors:1 dropped:0 overruns:1 carrier:0
collisions:0 txqueuelen:0
eth0 Link encap:Ethernet  HWaddr 00:C0:F0:1F:37:B4
    inet addr:XXX.XXX.XXX.YYY  Bcast:XXX.XXX.XXX.255  Mask:255.255.252.0
```

```
    UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
    RX packets:3573025 errors:0 dropped:0 overruns:0 frame:0
    TX packets:1643167 errors:1 dropped:0 overruns:1 carrier:0
    collisions:0 txqueuelen:100
    Interrupt:10 Base address:0x1080

eth1 Link encap:Ethernet  HWaddr 00:C0:F0:1F:37:B4
    inet addr:XXX.XXX.XXX.YYY  Bcast:XXX.XXX.XXX.255  Mask:255.255.252.0
    UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
    RX packets:3651769 errors:0 dropped:0 overruns:0 frame:0
    TX packets:1643480 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:100
    Interrupt:9 Base address:0x1400
```

第 七 节

问题解答

1. 这个模块在SMP 下安全吗?

   是的，在旧的2.0.xx 内核版本中，bonding 补丁对于SMP 平台是不安全的。新的内核从开始设计时，对于SMP 来说就是安全的。

2. 什么样的卡能和这个模块一起使用?

   任何的以太网卡都可以(你甚至可以混合不同的网卡来实现这个模块的功能,—例如，一个Intel EtherExpress PRO/100 和一个 3com 3c905b)你甚至可以将不同的Gigabit 以太网卡绑定在一齐。

3. 我可以有多少bonding 设备?

   一个因素是你安装的每个模块。请查看模块参数小节来得到这个答案。

4. 我可以用多少个slaves 设备?

   这取决于Linux 支持的网络接口的数目，以及你的机器最多可以放置的网卡的数量。

5. 当一个slave 链路死掉后，发生什么事情?

   如果你的以太网卡支持MII 状态监视并且MII 监视在这个设备上被建立(查看模块参数说明),这里没有相反的结果。这里发行的bonding 驱动知道如何得到MII 信息和建立按照模块的slaves 来使能或取消其链路状态。查看HA以得到更多的信息。

对于不支持MII状态的以太网卡，或如果你希望验证包已被发送和接收，你可以配置 arp_interval 和 arp_in_target，如果包在相应的时间段内没有发送和接收，一个 arp 请求被发送到目标地址来产生发送和接收传输。如果在这个时间段后，既没有成功发送包的情况，也没有成功接收的报告，下一个 slave 将按顺序变成活跃的 slave 。

如果 mii_monitor 和 arp_interval 两个都没有被配置，bonding 设备将不会很好的绑定这些状态。这些设备将继续发送包，并且有些包将会丢失。重传输将引起一系列的问题，并造成一系列的性能下降(在这种情况下，两个 slave 中的一个失效时，将有 50% 的包丢失，对于 TCP 和 UDP 协议来说将会出现一系列的问题)

6. bonding 可以用于高可靠性项目吗？

   是的，如果你用MII监视并且你所有的卡都支持MII链路状态报告，查看HA小节以得到更多的信息。

7. 什么样的交换机/系统可以使用bonding ？

   在循环模式下，模块可同支持 trunking 的交换机工作。

   - Cisco 5500 系列交换机(查看 EtherChannel 得到相关的技术支持)
   - SunTrunking 软件
   - Alteon AceDirector 交换机WebOS (使用Trunks)
   - BayStack 交换机(trunks必须被明确的配置)。Stackable 模式 (450)可以用来对不同的物理单元的不同端口进行配置。
   - 当然还有Linux的bonding功能

   在 Active-backup 模式下，它将工作于任何的二层交换模式[3]。

8. bonding设备从哪里得到MAC地址?

   如果没有通过 ifconfig 命令明确的配置，bonding设备的 MAC 地址从第一个 slave 设备处得到。之后 MAC 地址将传送给所有的 slave 没备使用，并一直使用这个 MAC 地址直至(甚至当第一个 slave 设备被移除时，这个 MAC 地址也被使用) bonding 设备重新配置或停止使用。

   如果你想改变 bonding 设备的 MAC 地址，你可以使用 ifconfig 命令来配置:

   ```
   # ifconfig bond0 hw ether 00:11:22:33:44:55
   ```

   MAC 地址也可以通过使 bonding 设备激活停用后，在改变 bonding 设备的 slave 设备(或他们的顺序)的方式来更改。

[3] Layer-II switches

```
# ifconfig bod0 down ; modprobe -r bonding
# ifconfig bond0 (这个位置写网络参数) up
# ifconfig bond0 eth ...
```

对过这个方法，将自动将第一个 slave 设备的 MAC 地址加到 bonding 设备中。

如想恢复 slave 设备的 MAC 地址，你需要将 slave 设备从 bonding 设备分离(‘ifenslave -d bond0 eth0‘),设置设备为停用状态(‘ifconfig eth0 down‘),卸载网卡驱动模块(‘rmmod 3c59x‘,如)并且重新加载驱动模块以从设备的 eeprom 得到设备的 MAC 地址。如果 bonding 设备由几个从设备共同构成，你需要关闭所的从设备。另外的一个解决方式是通过在系统启动时查找 MAC 地址(dmesg or tail /var/log/messages)和使用手动方式用 ifconfig 来更改:

```
# ifconfig eth0 down
# ifconfig eth0 hw ether 00:20:40:60:80:A0
```

9. 可以使用什么样的传输策略?

   Round robin,是基于加载网卡设备的顺序，数据输出设备是基于下一个可用的从设备。这个源或目的地址的包无关。

   XOR,基于(源硬件地址 XOR 目的硬件地址) % slave cnt .这样可以对相同的目的硬件地址使用同一个从网卡设备。

   Active-backup 策略确保一个网卡设备且仅有一个网卡设备在给定的时刻传送数据。Active-backup 策略主要用于高用可解决方案中使用两个hubs(查看 HA 小节)。

第 八 节

高可靠

使用 bonding 设备来实现高可靠方案，你需要将驱动以模块的方式编译到内核中，因为目前只有这种方式可以将参数传递到内核中，通过参数可以改变设备的功能。

高可靠性通过使用 MII 状态报告的方式来实现。你需要验证你所使用的所有网络接口都支持 MII 链路状态报告。在 Linux 内核 2.2.17 版本中，所有的 100Mbps 兼容的设备和 yellowfin gigabit 网卡设备也支持 MII 链路状态报告。如果你的系统不支持 MII 状态报告，一个链路失败将被发现!

bonding设备可以有规律地通过检查 MII 状态寄存器来检查所有的从网络设备的状态。检查过程的时间间隔可以通过加载模块时指定参数“miimon” (MII 监视)。这个参

数的值是一个整数，它代表每隔多少毫秒[4]来检查一下 MII 的状态。这个值最好不要接近于(1000/HZ)(10 ms 在i386)因为这个值将减少系统的交互性。100 ms应是一个合理的值。它代表着链路状态将在设备失效后100 ms内被发现。例如：

```
# modprobe bonding miimon=100
```

或，在你系统的 /etc/modules.conf 文件中设置如下：

```
alias bond0 bonding
options bond0 miimon=100
```

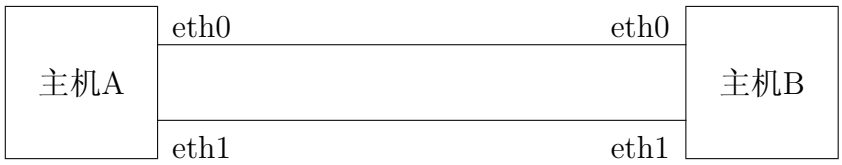
当前这里有两种策略可以用于高可靠方案中，使用的方式取决于是否：

1. 主机被连到一个单主机还是连到一个支持 trunk 功能的交换机上
2. 主机被连到多个交换机上还是连到一台不支持 trunking 功能的交换机上

8.1 HA应用于单交换机或主机 —— 负载平衡

这是最简单和最容易架设的方式。简单的配置远程设备(主机或交换机)以实现聚集几个端口的传输(Trunk,EtherChannel,等)并配置 bonding 接口。如果 bonding 模块加载时使用合适的 MII 选项，设备将自动工作。之后你可以尝试断开和恢复不同的链路，并在系统日志中查看驱动探知的事件。当测试时,你可能在很有问题的交换机上遇到当所有的 trunk 端口都停止工作时，交换机需很长时间才能停止 trunk 的情况。这不是 Linux 的问题，而确实是交换机的问题。(你可以通过重启交换机来确认问题出在哪里)

示例1: 双速传输主机到主机

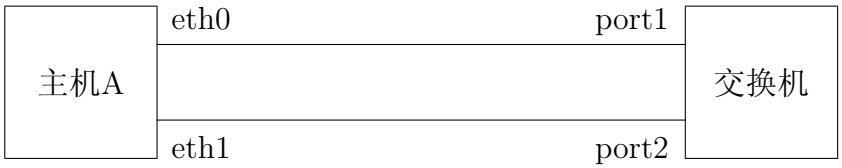


在每台主机上：

```
# modprobe bonding miimon=100
# ifconfig bond0 addr
# ifenslave bond0 eth0 eth1
```

示例2：双速的主机到交换机

[4]millisecond



主机A:

```
# modprobe bonding miimon=100
# set up a trunk on port1
# ifconfig bond0 addr
# ifenslave bond0 eth0 eth1
```

交换机:

```
# set up a trunk on port1
  and port2
```

8.2 HA在两个或多个交换机上(或不支持 trunking 的单交换机)

这个模式可能会遇到更多的问题，这是因为基于以下的事实，这里有多少端口并且主机的 MAC 地址只在一个端口可见，这可能会使交换机产生错误。

如果你需要知道哪个端口是处于活跃状态，并且哪个端口是处于备用状态，使用 ifconfig 命令。所有的备用端口都有 NOARP 标记被设定。

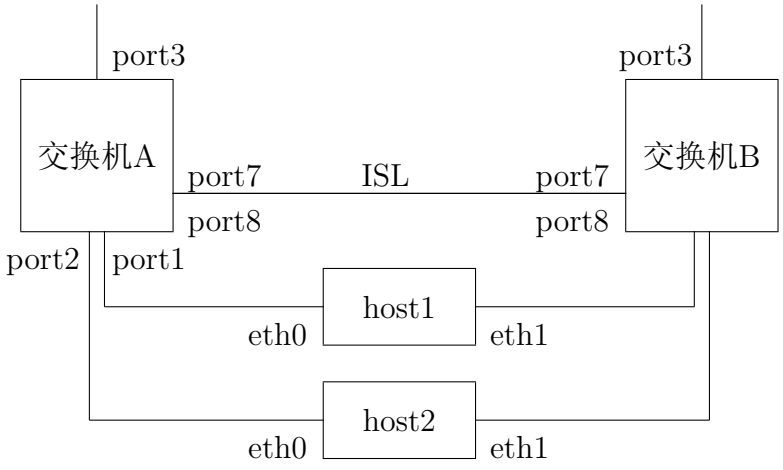
要使用这个模式，当加载模块时传递“mode=1”给模块。

```
# modprobe bonding miimon=100 mode=1
```

或，将下面的行放到你的/etc/modules.conf文件中。

```
alias bond0 bonding
options bond0 miimon=100 mode=1
```

示例 1: 使用多主机和多交换机来架设一个“无单点失效”解决方案。

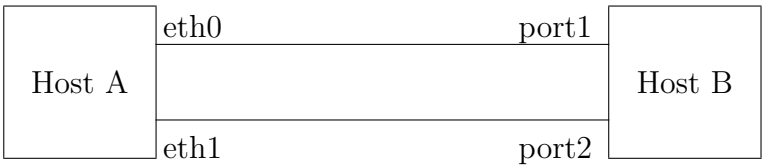


在这个配置中，这里有一个ISL— 内部交换链路[5](可以是一个trunk)，几台服务器(host1,host2 等等)同时连到每台交换机上，并且有一个或多少端口连到外部世界(port3等)。在每一个时刻在每台主机上只能有一个从网络接口处于活跃状态，然而所有的链路还都被监视着(系统可以察觉到活跃状态链路和备份链路失效)。

每次主机更改自已的活跃的网络接口状态，它利用新的网络接口直到原来的接口停止工作。在这个示例中，主机对于交换机的转发表的过期时间的影响不是十分敏感。

如果host1和host2有同样的功能，并且由其它的外部主机用来做负载平衡，最好将host1的活跃的主机接口连接到一个交换机，同时host2的主机的活跃的网络接口连接到另外一个交换机上。这种方案将在单一主机失效、单一电缆失效、单一交换机失效的情况下还能正常工作。最坏的情况是一个交换机失效时，将有一半的机器临时不可访问至其它的交换机刷新其列表时。

示例2：使用多以太网卡连接一个交换机来配置防止网卡失效(在这里交换机不需要支持trunking)




```
On host A:
  #modprobe bonding miimon=100 mode=1
  #ifconfig bond0 addr
  #ifenslave bond0 eth0 eth1
```

```
On the switch
   #(optional) minimize the time
   # for table expiration
```

每次当主机改变它的活跃网络接口状态，直至活跃状态停止后才起用一个新的网络接口。在这个示例中，主机受交换机内转发表过期时间的严重影响。

8.3 适应你交换机的时间

如果你的交换机花费了很长的时间启用备用模式，它可能不希望在当前活跃接口失效后立即激活备用接口。这时可能会延迟一会时间，当由模块参数“downdelay”(在毫秒级，必须是miimon值的倍数)决定的时间过后，这个链路才完全停止。

在相似的情况可能发生，当一个主机与交换机重协商一个丢失的链路时(如重插电缆的情况)

[5]Inter Switch Link

一个特殊情况是当一个 bonding 接口失去了所有的从链路时，然后 bonding 驱动将立即重用第一个重新激活的网络接口。甚至当 updelay 参数被指定时也如此。(如果当前的从网络接口在“updelay”状态，第一个达到激活状态的网络接口将被立即启用。)当 updelay 值设置过高时，可以减少中断时间。

示例：

```
# modprobe bonding miimon=100 mode=1 downdelay=2000 updelay=5000
# modprobe bonding miimon=100 mode=0 downdelay=0 updelay=5000
```

8.4 限 制

主要的限制是：

仅链路状态被监视。如果交换机的另外一端部分失效 (如，不能转发数据包，但链路状态是 OK),这时链路不被停用。另外的方法来检查失效的链路是在一个负载重的主机上对流入的数据祯计数。这对于小的服务来说不可用，但对于前端交换机对链路多播祯(如 VRRP)时比较有用。或者使用健康检查技服务。使用 arp_interval/arp_ip_target 参数来对流入/流出祯计数。

第 九 节

资源和链接

对于当前驱动的开发站点是:

```
http://www.sourceforge.net/projects/bonding/
```

Donald Becker 的以太网驱动程序和 diag 程序在下面的网址得到：

```
http://www.scyld.com/network/
```

你也可以从 www.scyld.com 这个站点上得到许多关于以太网，NWay,MII,等的信息。
对于这个驱动新的版本，老版本内核的补丁和更新及用户空间工具，可以访问 Willy Tarreau 的站点:

```
http://wtarreau.free.fr/pub/bonding/
http://www-miaif.lip6.fr/willy/pub/bonding/
```

如需得到最新的 Linux 内核开发信息，请咨询 Linux 内核邮件列表档案：

```
http://boudicca.tux.org/hypermail/linux-kernel/latest/
```