Linux Interface Bonding

P.M.J. de Bruijn

27th February 2005

License

Contents

1 Introduction

Do you have a really, really important server? Of course you do. Now wouldn't it be nice if we were able to protect that server from network failures. Network failures you ask? Well a lot can go wrong with your network equipment. Switches and hubs can fail. Network cards can die.

Fortunately the RedHat Enterprise Linux operating system is capable of mitigating the effects of these failures. The solution to our headaches is called bonding.

The Linux bonding driver also has load-balancing capabilities, but they are beyond the scope of this article.

2 Bonding, In Theory

The basic concept behind bonding is that two (or more) become one. You'll basically merge two physical NICs into a single logical NIC. The physical NICs will be called the slave interfaces. The logical NIC controls all the slave interfaces and thus is called the master interface.

When the logical interface is brought up, the bonding driver brings up the first physical interface in the team. The logical interface uses it's own MAC address and will respond to all incoming traffic destined for it.

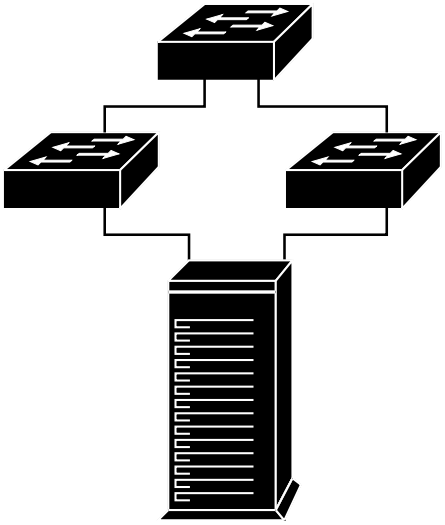
The master interface polls the active slave interface on a regular basis, to see whether network connectivity is still present. If the bonding driver determines that active slave interface went down, it quickly assigns the secondary slave interface the MAC address previously used by the primary slave interface. The interface is then brought up, and in turn becomes the active slave interface. It then begins accepting traffic.

3 Bonding, In Practice

3.1 Hardware

3.1.1 Switches

You can setup a NIC team to operate on a single switch, which will only protect you from NIC or cable failure. For maximum effectiveness you'll also need three (un)managed switches. One switches will act as a network backbone on the core layer. The other two will be the distribution/access layer switches. Ofcourse the backbone switch will still present a single point of failure into your network, this can be compensated for, by buying a modular switch, which can hotswap broken components.



3.1.2 NICs

To take proper advantage of bonding you'll need a server with at least two NICs, preferably (but not nessecarily) the same.

I used two Compaq NC3121 NICs, which are powered by Intel 825528B chips. Fortunately Intel maintains it's e100 driver well.

Keep in mind that the point of having transparent failover is avoiding user complaints. When using two different NICs, you risk not knowing how the secondary NIC will act when called upon.

3.2 Software

3.2.1 Distribution

For this article I'm using CentOS 3.4, which in turn is based on RedHat Enterprise Linux 3 Update 4. But the steps provided here should work on any RedHat like Linux distribution.

3.2.2 Driver

The first and most important thing to do is make sure you load the bonding driver at boot time. For this we need to modify the the /etc/modules.conf file, and adjust it to look like this:

modules.conf

```
alias bond0 bonding
options bond0 mode=active-backup miimon=100
alias eth0 e100
alias eth1 e100
```

The alias statements make sure the device files are associated with the proper drivers. Make sure you alias the bonding driver before you alias another NIC driver, this is important when using an SNMP agent. The options statement will tell the bonding driver what mode we will use and at what interval (in ms) to monitor our NICs link status.

For more information on the mode parameter please review the bonding.txt document which should be supplied with the kernel sources of your distribution.

3.2.3 Configuration

After the driver is properly loaded, we can start configuring our boot scripts.

Each interface is configured through a corresponding ifcfg-ifname file located in /etc/sysconfig/network-scripts/. We'll start by configuring our master interface bond0:

ifcfg-bond0

```
DEVICE=bond0
BOOTPROTO=static
ONBOOT=yes
BROADCAST=192.168.2.255
IPADDR=192.168.2.13
NETMASK=255.255.255.0
```

```
GATEWAY=192.168.2.1
USERCTL=no
TYPE=Ethernet
```

Next the configuration of our first physical interface:

ifcfg-eth0

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

Our failover physical interface:

ifcfg-eth1

```
DEVICE=eth1
BOOTPROTO=none
MASTER=bond0
SLAVE=yes
USERCTL=no
TYPE=Ethernet
```

As you might have noticed the slave interfaces have no IP addresses assigned to them, because they inherit the IP address of their master interface.

3.2.4 Activating

Everything should now be ready. You can use the normal ifup command to activate our newly created NIC team:

```
#ifup bond0
ip_tables: (C) 2000-2002 Netfilter core team
Enslaving eth0 to bond0
Enslaving eth1 to bond0
e100: eth0 NIC Link is Up 100Mbps Full duplex
e100: eth1 NIC Link is Up 100Mbps Full duplex
#
```

As you can see, both eth0 and eth1 are enslaved to bond0, as they should be.

4 Production Use

4.1 miimon

When using bonding in a production environment, you should carefully select the miimon interval. A smaller interval incurs more CPU load.

4.1.1 TCP

Most TCP based protocols should work perfectly using the 100ms miimon interval used in this article. Connection may stall for a split second when the server is changing NICs.

4.1.2 UDP

UDP based protocols are a whole different ball park. It depends heavily on the upper layer protocol whether loss of connectivity and/or data will occur. UDP is a connectionless protocol, without a garantee of safe delivery. Make sure you carefully test all your mission critical applications based on UDP.

4.2 Network

When using the multi-tier network approach presented here, you should also be aware that it may take the switches some time to notice the changes in the network that are occuring.

If your server does failover to it's second NIC, the second switch it's connected to will be briefly confused. The switch has the same MAC address available on it's upstream link.

The same thing goes for the backbone switch, the MAC address will be seen briefly on two different downstream ports. Eventually the old port will age the MAC address out of it's MAC database, and connectivy will resume. Lucky for us, this all happens within a couple of ms, and nobody will notice, if all goes well.

5 Conclusion

Linux interface bonding allows us to provide reliable network connectivity to any given Linux server without incurring the cost of an add-on software product.