Channel Bonding per il Cluster Arcosu

Carlo Podda, CRS4

CRS4
Centro di Ricerca, Sviluppo e Studi Superiori in Sardegna
Polaris, Edificio 1
Loc. Pixina Manna
09010 Pula (Cagliari), Italy

E-mail: carlo@crs4.it

Contents

1 Introduzione

L'utilizzo del channel bonding consente alle macchine che hanno più di una scheda di rete di "unire" le due schede come fossero una sola nuova scheda (master). Questa interfaccia "virtuale" (master) dovrebbe consentire prestazioni tali da raggiungere "teoricamente" la banda cumulata delle due schede originali (slaves).

2 Configurazione

2.1 Descrizione dei server con le schede in channel bonding

Per il cluster beowulf del CRS4 il channel bonding è stato configurato nei nuovi server NFS e di backup aiodda.crs4.it e aiodda2.crs4.it. Si tratta di macchine dual xeon 3.0GHz 1Mb di cache, con motherboard Supermicro X5DP8-G2. La dotazione di dischi e di 6 dischi Seagate 10K rpm SCSI ultra320 da 76 GB l'uno e di 4 dischi Maxtor MaxLine Plus II Serial ATA da 250 GB l'uno. Sia i dischi SATA che i dischi SCSI sono configurati in raid5 hardware: quelli scsi con controller raid aggiuntivo adaptec zero channel 2010S, quelli sata con controller 3ware 8506-4LP. In tutto la macchina dispone di circa 750GB su dischi SATA e 367 GB su dischi SCSI. Le aiodda hanno inoltre due schede giga ethernet di tipo Intel PRO/1000 (82546EB).

L'utilizzo delle macchina è quello di server NFS disponibile on line per gli utenti del cluster sulle macchine di calcolo. Le macchine sono collegate al resto del cluster tramite uno switch giga ethernet SMC8624T (TigerSwitch).

2.2 Le macchine utilizzate nel test

I client utilizzati nei test sono 6 macchine (scivu1, ... scivu6) con le seguenti caratteristiche:

dual cpu opteron 246 (2000 Mhz)
1 GB ram DDR 333
due NIC NetXtreme BCM5704 Gigabit Ethernet (driver tg3)
1 Maxtor 6Y080L0 ATA disk 80 GB.

Channel bonding è utilizzato al CRS4 sulle macchine server, in particolare, su aiodda1 e aiodda2. Aiodda1 e aiodda2 sono macchine gemelle, ciascuna è configurata per poter sostituire in caso di bisogno l'altra. Le macchine sono configurate per effettuare una copia notturna dei dati da aiodda1 ad aiodda2. Sia il backup dei dati utente che la copia dei dati tra i server viene effettuato con una procedura che fa uso del programma rsync. Tramite un utilizzo particolare di rsync, gestito da uno script di shell appositamente creato (testato da tempo nel server piscinas) sarà possibile fare il backup solo ed esclusivamente dei file effettivamente modificati o creati nell'intervallo dall'ultimo backup. Tramite gli hard link viene eliminato lo spreco di spazio disco da un backup all'altro. I backup verranno conservati per un periodo di tempo da definire (tipicamente un mese per i file system utente sotto backup e per i file system di sistema: software e codice vario). Il periodo di conservazione sarà variato a seconda dell'utilizzo delle macchine e della capacità dei dischi.

L'utilizzo delle macchine come server NFS per lettura e scrittura fa delle aiodda le candidate ideali per la configurazione delle schede in channel bonding. Data la potenza di elaborazione dei nodi di calcolo (perlopiù dual opteron anche se sono presenti 8 dual athlon e 4 dual pentiumIII) e la velocita delle connessioni alla rete della maggior parte di esse (giga ethernet per tutte le 18 dual opteron) il collo di bottiglia della connessione tra più nodi e il server diventerebbe l'unica interfaccia giga ethernet utilizzata. Inoltre il possibile trasferimento per il backup notturno di grossi insiemi di dati da aiodda1 a aiodda2 non fa che avvalorare la scelta di una configurazione di questo tipo.

3 Configurazione software

Per la configurazione delle due schede slave (eth0 e eth1) in una sola scheda master (bond0) abbiamo seguito le istruzioni della procedura di configurazione del channel bonding descritta in maniera dettagliata nella documentazione fornita con il kernel. [1][2].

La prima cosa da verificare è che con il kernel sia stato effettivamente compilato il modulo bonding.o

Facciamo partire il programma di configurazione del kernel

```
make menuconfig/xconfig/config
```

quindi selezioniamo

```
"Bonding driver support" da "Network device support".
```

(È consigliato creare e caricare bonding come modulo in quanto secondo la documentazione è l'unico modo per passare i parametri desiderati al sistema). Compiliamo il kernel.

La versione di bonding richiede una versione aggiornata del programma ifenslave (il file Documentation/networking/ifenslave.c contiene il codice). Per compilare ifenslave:

```
gcc -Wall -Wstrict-prototypes -O -I/usr/src/linux-2.X.X/include \
   ifenslave.c -o ifenslave
```

(IMPORTANTE: è necessario compilare utilizzare il file

```
/usr/src/linux/include/linux/if_bonding.h
```

invece della versione in /usr/include/linux. Se si utilizzasse questa versione il channel bonding non funzionerebbe.)

Se la compilazione va a buon fine

```
cp ifenslave /sbin/ifenslave
```

Passiamo ora alla sezione di configurazione dei moduli. Nel file /etc/modules.conf dobbiamo aggiungere la seguente riga:

```
alias bond0 bonding
```

(in seguito parleremo del passaggio di parametri particolari al momento del caricamento del modulo bonding)

Nella cartella
/etc/sysconfig/network-script/
abbiamo creato il file ifup-bond0 fatto cosi:

```
(da verificare)
DEVICE=bond0
IPADDR=156.148.82.XX
NETMASK=255.255.254.0
NETWORK=156.148.82.0
BROADCAST=156.148.83.255
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
```

Comè già detto in precedenza le interfacce che fanno parte di un trunk (altro modo per definire l'accoppiamento di due o più interfacce in channel bonding) sono definite slaves, l'interfaccia ottenuta è invece la master. I file ifup-eth0 e ifuop-eth1 sono diventati quindi:

```
DEVICE=eth0  (DEVICE=eth1 nel file ifup-eth1)
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
```

Possiamo ora far partire la rete con la nuova configurazione.

```
/etc/rc.d/init.d/network restart
```

Controlliamo lo stato delle interfacce attive con

```
/sbin/ifconfig
```

Se tutto è OK possiamo passare alla configurazione dello switch per mettere in trunk le due porte utilizzate per il bonding . Nel nostro switch bisogna andare su Port/Trunk Membership e creare il trunk che associ le due porte utilizzate.

Analizzando i log di sistema /var/log/messagges controlliamo se ci sono messaggi di errore o warning. Con la configurazione adottata verranno segnalati possibili errori dovuti alla mancanza del parametro miimon. Nel nostro caso abbiamo eliminato l'errore inserendo nel file
/etc/modules.conf la riga:

```
options bond0 miimon=100
```

Con questa opzione diciamo al driver di controllare ogni 100 millisecondi se una delle interfacce va giù. Questo eviterà rallentamenti e possibili congestioni o problemi della rete. Inoltre con MII implementiamo l'High Availability utilizzando. Con questo parametro settato il driver bonding sa come comportarsi se una delle interfacce va giù disabilitandola per evitare perdite di pacchetti o congestioni della rete. Di default, come nel nostro caso dove non viene passato in
/etc/modules.conf il parametro mode =X, il bonding viene implementato utilizzando una politica di distribuzione dei pacchetti in round-robin. Con il round robin la scheda da utilizzare di volta in volta viene scelta sulla base della sua disponibilità a prescindere dalla sorgente e/o destinazione del pacchetto.

4 Test e Performance

Su aiodda2 abbiamo eseguito dei test per verificare l'efficienza della rete con e senza utilizzare channel bonding. I test sono stati eseguiti con il programma iperf [3] un tool per la misurazione delle prestazioni della banda di rete tra diverse macchine. Ogni test consisteva nella trasmissione simultanea da ciascuno dei 6 client scivu verso il server aiodda di un flusso di pacchetti TCP verso il server. La durata di ogni trasmissione era di 10 secondi. Sul server iperf è stato avviato con il comando

```
iperf -s -D (modalit{\‘a} server tcp come daemon)
```

Sul client:

```
iperf -c aiodda2
```

Alla fine di ciscuna trasmissione il client scrive su un file le performance misurate in Mbytes/sec. Con questo test condotto simultaneamente su diversi

client abbiamo testato il comportamento dei sistemi in un caso "limite" per un cluster di calcolo parallelo ad alte prestazioni. Prima per il channel bonding poi per la configurazione single ethernet sono stati eseguiti 20 trasmissioni per client. I risultati finali sintetizzati nella tabella 4.1 mostrano le prestazioni medie per client e il flusso totale medio di ciascun test. Possiamo vedere come utilizzando il channel bonding si ha un miglioramento delle prestazioni di circa il 63%.

Single ethernet		Channel bonding	
Average single client (Mbits/sec)	Total bandwidth (single client x 6) (Mbits/sec)	Average single client (Mbits/sec)	Total bandwidth (single client x 6) (Mbits/sec)
169,25	1015,50	275,92	1655,57

Table 4.1: Prestazioni medie ottenute nei test di throughput della rete, eseguito con il programma Iperf, tra il server aiodda2 e i 6 client scivu. I risultati sono la media di 20 trasmissioni di dati, di dieci secondi l'una, e riportano le prestazioni di aiodda2 collegata alle scivu con una sola interfaccia ethernet e con due interfacce ethernet configurate in channel bonding.

Abbiamo quindi fatto una seconda coppia di test facendo variare il numero di client interessati di volta in volta nella trasmissione. Sempre con il comando *iperf -c aiodda2* eseguito tra un client e il server, poi con due client, e così via sino a 6 client. Questa volta per ogni client in ogni test sono state eseguite 10 trasmissioni in TCP di 10 secondi l'una. I risultati che mostrano la banda media in Mbits/sec per singolo client e totale al variare dei client sono mostrati nella tabella 4.2.

Nel server, rispetto alla configurazione di default sul server aiodda2, nel file /etc/sysctl.conf, sono state aggiunte le righe seguenti per incrementarne le prestazioni. Vedi [4].

```
# modifico windows size per aumentare la velocit{\'a} della rete
# increase Linux TCP buffer limits
net.core.rmem_max = 8388608
net.core.wmem_max = 8388608
net.core.rmem_default = 65536
net.core.wmem_default = 65536

# increase Linux autotuning TCP buffer limits
# min, default, and max number of bytes to use
net.ipv4.tcp_rmem = 4096 87380 8388608
```

	Single ethernet		Channel bonding	
Num Client	Average single client (Mbits/sec)	Total bandwidth (Mbits/sec)	Average single client (Mbits/sec)	Total bandwidth (Mbits/sec)
1	830,0	830,0	827,2	827,2
2	484,7	969,4	488,2	976,5
3	322,4	967,7	333,5	1000,7
4	253,2	1013,1	352,1	1408,4
5	201,9	1009,6	320,1	1600,9
6	175,1	1050,8	273,1	1638,3

Table 4.2: Prestazioni medie ottenute nei test di throughput della rete, eseguito con il programma Iperf, tra il server aiodda2 e un numero variabile di client scivu (da uno a sei). I risultati sono la media di 10 trasmissioni di dati, di dieci secondi l'una, e riportano le prestazioni di aiodda2 collegata alle scivu con una sola interfaccia ethernet e con due interfacce ethernet configurate in channel bonding.

```
net.ipv4.tcp_wmem = 4096 65536 8388608
# number of pages, not bytes
net.ipv4.tcp_mem = 4096 4096 4096
```

Acknowledgments

Questo lavoro è finanziato parzialmente dal MIUR D.D. 9/10/2002, prot. 1105/2002, prog. n. 212 e parzialmente della Regione Autonoma della Sardegna.

Riferimenti

Il mini-HOWTO di Thomas Davis cita altri riferimenti, in particolare:

- **Donald Becker's Ethernet Drivers and diag programs [5]. You will also find a lot of information regarding Ethernet, NWay, MII, etc. at www.scyld.com.**
- **For new versions of the driver, patches for older kernels and the updated userspace tools, take a look at Willy Tarreau's site [6],[7].**

References

[1] **Thomas Davis (tadavis@lbl.gov), Willy Tarreau (willy@meta-x.org), Constantine Gavrilov (const-g@xpert.com), Chad N. Tindel (ctindel@ieee.org), Janice Girouard (girouard@us.ibm.com) "Linux Ethernet Bonding Driver HOWTO" http://prdownloads.sourceforge.net/bonding/bonding.txt**

[2] **Current development on the bonding driver is posted to http://www.sourceforge.net/projects/bonding/**

[3] **http://dast.nlanr.net/Projects/iperf**

[4] **http://www-didc.lbl.gov/TCP-tuning/, http://www-didc.lbl.gov/TCP-tuning/buffers.html**

[5] **http://www.scyld.com/network/**

[6] **http://wtarreau.free.fr/pub/bonding/**

[7] **http://www-miaif.lip6.fr/willy/pub/bonding/**