

# **Cluster di due server Linux con DRBD&Heartbeat**

da <http://escher07.altervista.org>

## **Generalità**

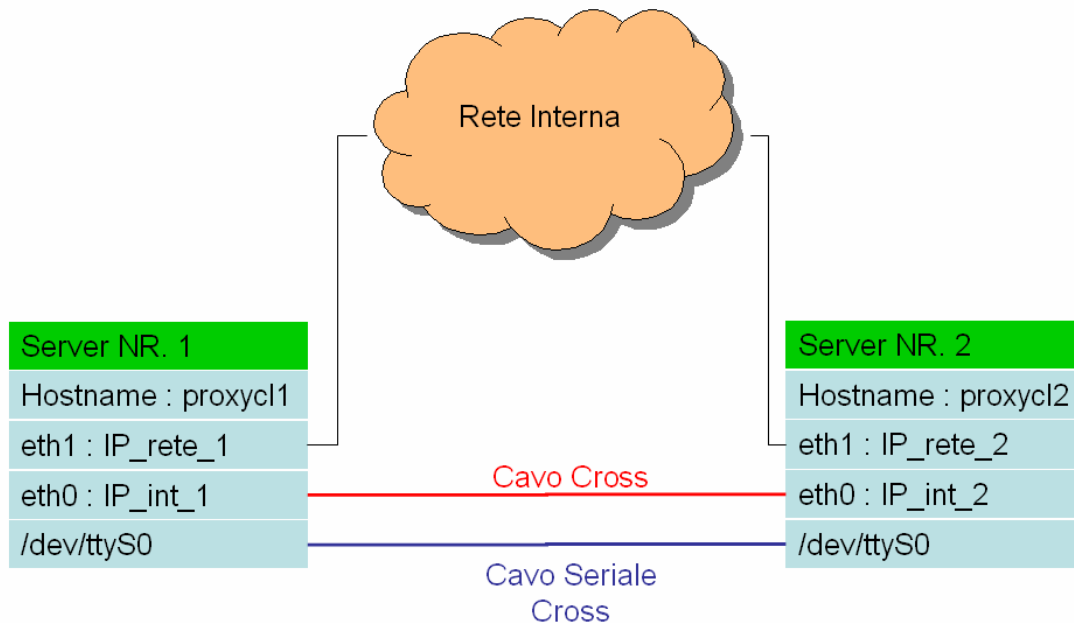
Questo documento descrive i passi che sono stati compiuti per configurare un cluster costituito da due server identici per realizzare, nello specifico, un Proxy + DNS Interno ad alta affidabilità in ambiente Linux. Nel testo, che è stato un po' il mio verbale della predisposizione del cluster ho preferito lasciare anche parti relative agli errori commessi ed i metodi che ho adottato per uscire dagli inghippi pensando che magari sono banali ma che potrebbero essere di qualche utilità a chi finisca nelle stesse trappole. Ho cambiato solo qualche indirizzo, nome host, password e simili per non compromettere la sicurezza del cliente rivelando delle informazioni riservate.

Il tutto chiaramente non ha niente di accademico e vuole piuttosto riportare un caso molto pratico per cui eventuali critiche (perchè costruttive!) commenti e suggerimenti sono sempre bene accetti. Vale sempre il discorso che non mi assumo alcuna responsabilità sulle conseguenze dirette o indirette dell'applicazione corretta o meno di quanto detto.

Un'ultima nota : il cluster serve per incrementare la tolleranza al guasto rispetto a quella che si avrebbe dei sistemi singoli. Il che vale chiaramente, per tutto quello che si è deciso di ridondare. In altri termini si dà per scontato che i cavi (cross e seriale) ed il collegamento delle due macchine alla rete sia "perfetto". Questo ci fa capire che ad esempio un primo punto di miglioramento può essere quello di ridondare anche il cavo cross (in tal caso i server devono avere 3 schede di rete) e magari anche la seriale e/o il collegamento alla LAN.

## **Piano degli Indirizzi**

La configurazione obiettivo è questa:



Dove effettuiamo le seguenti assegnazioni:

```
IP_rete_1    = 172.16.250.101
IP_int_1     = 192.168.0.1
IP_rete_2    = 172.16.250.102
IP_int_2     = 192.168.0.2
```

Le schede Ethernet sono tutte delle Gigabit : E' stata scelta per la comunicazione interna la eth0, l'unica che supporta un MTU di 9000 (settato con `ifconfig eth0 mtu 9000`) in modo da ottimizzare il transito dei pacchetti di sincronizzazione fra i due server. Per entrambi i server è stato configurato in questo modo il file `/etc/hosts`:

```
172.16.250.101    proxycl1    proxycl1
172.16.250.102    proxycl2    proxycl2
172.16.250.103    proxycluster proxycluster
192.168.0.1       proxyclint1 proxyclint1
192.168.0.2       proxyclint2 proxyclint2
```

Il collegamento seriale andrà configurato ma **sarà gestito da HeartBeat**. Attenzione a non mettere `pppd` nell'avvio automatico (come dicono ad esempio i vari serial HowTo, corretti ma per utilizzare la seriale e farla gestire da `pppd`!) altrimenti Heartbeat trova la device occupata e succede un macello!

Se tutto è stato configurato correttamente non dovremmo trovare nel "messages" alcuna riga su `pppd` ed una riga come questa:

```
Jan 9 18:38:07 proxycl1 heartbeat: [2876]: info: glib: Starting serial heartbeat on tty /dev/ttyS0
(115200 baud)
```

e all'apparizione dell'altro nodo qualcosa del genere:

```
Jan 9 18:49:02 proxycl1 heartbeat: [3653]: info: Link proxycl2:/dev/ttyS0 up.  
Jan 9 18:49:02 proxycl1 heartbeat: [3653]: info: Status update for node proxycl2: status init  
Jan 9 18:49:02 proxycl1 ipfail: [3868]: info: Link Status update: Link proxycl2//dev/ttyS0 now has  
status up
```

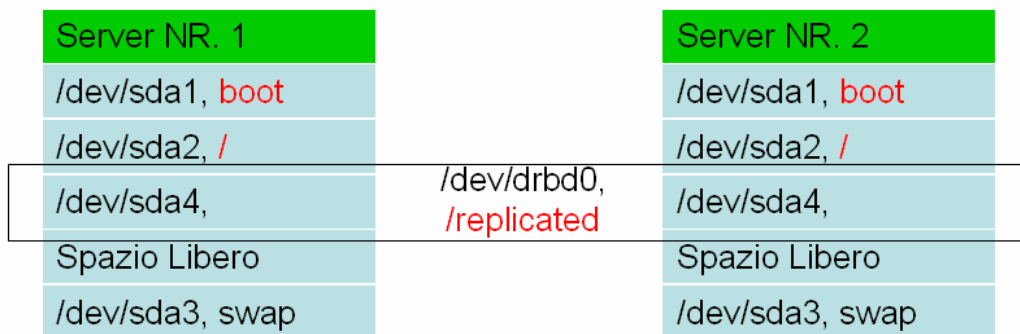
Inoltre facendo cat /proc/ttyS0 dovremmo ottenere dei messaggi simili:

```
>>>  
t=status  
st=active  
dt=9c0  
protocol=1  
src=proxycl2  
(1)srcuuid=fZV7nxBtQBOY/kNpuzXUmA==  
seq=b55e  
hg=1a  
ts=45a48caf  
ld=0.05 0.02 0.00 1/634283  
ttl=4  
auth=1 b1c57c7a88ead8d7de6317c31e4689a706cbd0d5  
<<<  
>>  
t=NS_ackmsg  
dest=proxycl1  
ackseq=b752  
(1)destuuid=E/TypolPRjq998E+atSM6w==  
src=proxycl2  
(1)srcuuid=fZV7nxBtQOY/kNpuzXUmA==  
hg=1a  
ts=45a48caf  
ttl=4  
auth=1 88a88e68aa61ea9dc9dfcd39ad9ee0592ced2258  
<<<
```

Le macchine hanno HD da 74 GB (in realtà 2 HD fisici da 74 GB in RAID1) e montano Linux Red Hat ES 3 U5 con le seguenti impostazioni:

partizione	Tipo	mount	start	end	MB	MB (input)
/dev/sda1	ext3	boot	1	13	102	auto
/dev/sda2	ext3	/	14	2563	20003	20000
/dev/sda4	ext3		2564	5113	20003	20000
Libero			5114	9455	34060	auto
/dev/sda3	ext3	swap	9456	9709	1992	auto

Il tutto per realizzare un qualcosa di questo tipo:



In sostanza nelle partizioni di sistema alcune cartelle fisiche (es. /etc/squid) sono sostituite da dei link simbolici presenti sulla drbd0 (es. /replicated) chiaramente dopo che questa è stata creata e formattata.

Nota: in fase di installazione la /dev/sda4 non è associata ad alcuna cartella del filesystem (ovvero non c'è una riga corrispondente di mount all'interno dell'fstab). Questo perché le /dev/sda4 non saranno accessibili direttamente ma dovrà esserlo il livello superiore, ovvero la /dev/drbd0 associata alla /replicated (una cartella /replicated per ogni macchina che puntano alla stessa risorsa condivisa corrispondente alla "somma" delle due /dev/sda4).

Riguardo ai pacchetti è stato scelto il default più i seguenti:

Editor (3/5)  
 DNS (2/2)  
 Server Ftp (1/1)  
 Strumenti di sviluppo (60/62)  
 Sviluppo Kernel (5/5)  
 Sviluppo Software Gnome (39/39)

-----  
 Totale : 2,367 MB

E' evidente che è opportuno che i due server siano sincronizzati nel senso che "battano" la stessa ora : per far questo utilizziamo il demone NTPD, per la cui configurazione (di base) si rimanda al file "Impostare NTP".

## Installazione dei programmi

### Prerequisito Heartbeat : Installazione di Libnet

Con la RED HAT 9 Enterprise 3 si ottengono vari warning scaricando la Libnet dal sito ufficiale perché questa è rimasta alla 0.x. Le versioni recenti si trovano come RPM : di queste quella che ci consente di terminare anche il passo 2 è la **libnet-1.0.2-2.1.el3.rf.i386.rpm** prelevata da:

<http://dag.wieers.com/packages/libnet/>

### HeartBeat

Dal sito <http://linux-ha.org/download/index.html> scarichiamo la versione 2.0.7 ovvero il seguente file:

[heartbeat-2.0.7.tar.gz](http://linux-ha.org/download/index.html)

Decomprimiamolo creando una cartella heartbeat-2.0.7 ad esempio sotto usr/src come di consueto (quindi \usr/src heartbeat-2.0.7). Qui lanciamo il comando:

```
./ConfigureMe package
```

...con cui vengono generati gli RPM, come dimostrano i log:

```
Wrote: /usr/src/redhat/SRPMS/heartbeat-2.0.7-1.src.rpm
Wrote: /usr/src/redhat/RPMS/i386/heartbeat-2.0.7-1.i386.rpm
Wrote: /usr/src/redhat/RPMS/i386/heartbeat-ldirectord-2.0.7-1.i386.rpm
Wrote: /usr/src/redhat/RPMS/i386/heartbeat-stonith-2.0.7-1.i386.rpm
Wrote: /usr/src/redhat/RPMS/i386/heartbeat-pils-2.0.7-1.i386.rpm
Wrote: /usr/src/redhat/RPMS/i386/heartbeat-debuginfo-2.0.7-1.i386.rpm
```

Si tratta a questo punto di posizionarsi nelle cartelle in questione (/usr/src/redhat/SRPMS e /usr/src/redhat/RPMS/i386) e dare i seguenti comandi:

```
rpm --install /usr/src/redhat/SRPMS/heartbeat-2.0.7-1.src.rpm
rpm --install /usr/src/redhat/RPMS/i386/heartbeat-pils-2.0.7-1.i386.rpm
rpm --install /usr/src/redhat/RPMS/i386/heartbeat-debuginfo-2.0.7-1.i386.rpm
rpm --install /usr/src/redhat/RPMS/i386/heartbeat-stonith-2.0.7-1.i386.rpm
rpm --install /usr/src/redhat/RPMS/i386/heartbeat-2.0.7-1.i386.rpm
rpm --install /usr/src/redhat/RPMS/i386/heartbeat-ldirectord-2.0.7-1.i386.rpm
```

L'installazione degli RPM va eseguita in questo ordine (meglio da linea di comando come nelle righe precedenti che da interfaccia grafica che qualche volta si blocca) ma si può verificare che prima di poterla fare bisogna procurarsi degli RPM perché questi pacchetti hanno a loro volta delle dipendenze la cui presenza è indispensabile. Riguardo a queste possiamo far riferimento a questa pagina:

<http://www.ultramonkey.org/download/3/rh.el.3/RPMS/>

dove troviamo questi RPM utili:

<a href="#">arptables-noarp-addr-0.99.2-1.rh.el.um.1.noarch.rpm</a>	27-Dec-2005 09:53
<a href="#">arptables_jf-0.0.7-0.3E.i386.rpm</a>	25-Jul-2005 11:58
<a href="#">heartbeat-1.2.3.cvs.20050927-1.rh.el.um.3.i386.rpm</a>	03-Aug-2006 05:37
<a href="#">heartbeat-ldirectord-1.2.3.cvs.20050927-1.rh.el.um.3.i386.rpm</a>	03-Aug-2006 05:37
<a href="#">heartbeat-pils-1.2.3.cvs.20050927-1.rh.el.um.3.i386.rpm</a>	03-Aug-2006 05:37
<a href="#">heartbeat-stonith-1.2.3.cvs.20050927-1.rh.el.um.3.i386.rpm</a>	03-Aug-2006 05:37
<a href="#">ipvsadm-1.21-1.rh.el.1.um.1.i386.rpm</a>	04-Apr-2005 12:07
<a href="#">libnet-1.1.2.1-1.rh.el.um.1.i386.rpm</a>	04-Apr-2005 12:07
<a href="#">perl-Authen-SASL-2.08-1.rh.el.um.1.noarch.rpm</a>	04-Apr-2005 12:07
<a href="#">perl-Convert-ASN1-0.18-1.rh.el.um.1.noarch.rpm</a>	04-Apr-2005 12:07
<a href="#">perl-IO-Socket-SSL-0.96-1.rh.el.um.1.noarch.rpm</a>	04-Apr-2005 12:07
<a href="#">perl-Mail-IMAPClient-2.2.9-1.rh.el.um.1.noarch.rpm</a>	04-Apr-2005 12:07
<a href="#">perl-Net-SSLeay-1.25-1.rh.el.um.1.i386.rpm</a>	04-Apr-2005 12:07
<a href="#">perl-Parse-RecDescent-1.94-1.rh.el.um.1.noarch.rpm</a>	04-Apr-2005 12:07
<a href="#">perl-XML-NamespaceSupport-1.08-1.rh.el.um.1.noarch.rpm</a>	04-Apr-2005 12:07
<a href="#">perl-XML-SAX-0.12-1.rh.el.um.1.noarch.rpm</a>	04-Apr-2005 12:07
<a href="#">perl-ldap-0.3202-1.rh.el.um.1.noarch.rpm</a>	04-Apr-2005 12:07

Servono inoltre questi pacchetti:

<http://dag.wieers.com/packages/perl-Class-Date/perl-Class-Date-1.1.7-1.1.el3.rf.i386.rpm>  
<http://dag.wieers.com/packages/perl-Date-ICal/perl-Date-ICal-1.72-1.1.el3.rf.noarch.rpm>  
<http://dag.wieers.com/packages/perl-Date-Leapyear/perl-Date-Leapyear-1.71-1.1.el3.rf.noarch.rpm>  
<http://dag.wieers.com/packages/perl-TimeDate/perl-TimeDate-1.16-0.rhel3.dag.noarch.rpm>  
<http://dag.wieers.com/packages/perl-MailTools/perl-MailTools-1.60-0.rhel3.dag.noarch.rpm>  
<http://dag.wieers.com/packages/perl-Net-IMAP-Simple/perl-Net-IMAP-Simple-1.14-1.1.el3.rf.noarch.rpm>

Aggiungiamo poi da <http://www.ultrammonkey.org/download/heartbeat/extras/rhel4/>

[perl-Net-IMAP-Simple-1.16-1.rh.el.um.1.noarch.rpm](#)  
[perl-Net-IMAP-Simple-SSL-1.3-1.rh.el.um.1.noarch.rpm](#)  
[perl-POP3Client-2.17-1.rh.el.um.1.noarch.rpm](#)

...ed infine da nakedape.cc questo:

<http://ftp.nakedape.cc/pub/nakedape/rpms/main/el3/rpm/perl-libnet/>

## DBRD

Download del file “GZ”. DBRD deve comparire fra i moduli del sistema e se non lo è occorre ricompilare il kernel. Per poterlo fare assicuriamoci che siano presenti i tool di sviluppo di sistema e del kernel.

Per inciso se noi non facessimo i passaggi relativi all’aggiunta del modulo DBRB otterremmo, dopo aver fatto make e make install un messaggio tipo “*The present kernel configuration has modules disabled ...*” .

Andiamo ora nella cartella che contiene il sorgente del SO. In RH ES 3 questa è sotto /usr/src/linux-2.4 che è un collegamento alla /usr/src/linux-2.4.21-32.EL dove si trovano fisicamente i file. Questa tecnica è usata per poter mantenere la stessa struttura per compilare una versione oppure un'altra. Dunque da qui lanciamo i seguenti comandi:

```
vi Makefile (sostituire "custom" con "smp" nel paragrafo EXTRAVERSION)
make mrproper
cp /boot/config-2.4.21-32.ELsmp .config
make -s oldconfig_nonint ; make -s oldconfig_nonint
```

Osservazioni: nel caso trattato i server sono degli Xeon Dual core, quindi l'immagine lanciata di default è quella SMP ovvero col supporto multi processore. Bisogna quindi riferirsi ad essa anche se il sorgente è uno sia per la versione senza tale supporto che quella con tale supporto (/usr/src/linux-2.4.21-32.EL nel nostro caso).

La modifica al Makefile serve per far capire al compilatore che la "extraversion" a cui ci si riferisce non si chiama 2.4.21-32.ELcustom ma 2.4.21-32.ELsmp. E gli altri comandi che cosa fanno?

Con il primo eliminiamo il corrente file .config presente nella stessa cartella (e nascosto, si vede con ls -la). Con il secondo copiamo il file .config presente nella partizione di boot nel file presente nella cartella corrente. Con l'ultimo comando lanciamo la creazione della configurazione del kernel senza chiedere opzioni (nonint = non interrupting) per due volte in modo da avere anche una copia dell'ultima configurazione funzionante. A questo punto abbiamo nella cartella in questione due file, config e config.old creati in data di oggi.

Si tratta ora di compilare i moduli presenti, ovvero dare i seguenti comandi:

```
make dep
make include/linux/version.h
make modules
```

A questo punto bisogna lanciare la compilazione del modulo DRBD nella directory dei moduli del kernel. Scompattiamo ora il GZ in una cartella (/usr/src come di consuetudine) creando una sotto cartella dbrd-0.7.22 e qui diamo i seguenti comandi:

```
vi drdb_config.h ( decommentare //#define REDHAT_HLIST_BACKPORT)
cd /usr/src/drdb-0.7.22/user
make
cd /usr/src/drdb-0.7.22/drdb
make clean all
cd /usr/src/drdb-0.7.22
make install
```

Questo anche se la documentazione (file INSTALL) dice qualcosa di diverso (si può verificare che seguendo alla lettera tali istruzioni si ottiene un errore). Riguardo alla KDIR il compilatore dovrebbe far riferimento alla versione SMP che è quella con cui i due SO si avviano di default nell'esempio in questione ovvero dovremmo vedere a video il riferimento a /lib/modules/2.4.21-32.ELsmp/build.

## Configurazione dei programmi

### DBRD

Si tratta a questo punto di creare il device virtuale /dev/drbd0 corrispondente ai due device fisici che vogliamo mettere in RAID e configurarne le caratteristiche.

Il Primo passo è su ciascuna macchina togliere alla partizione da usare per il raid la device assegnata dal sistema e metterci la nostra. Dobbiamo lanciare quindi il comando che segue:

```
dd if=/dev/zero of=/dev/sda4
```

Per azzerare la corrispondente partizione, ossia riempirla fisicamente di tutti zero (tanto è vero che dopo questo “trattamento” non viene più riconosciuta come EXT3). Per inciso nel mio caso il comando effettivamente lanciato è stato questo:

```
dd if=/dev/zero of=/dev/sda4 conv=notrunc count=4096000
```

che ha dato il seguente output:

```
entrati 40960000+0 record
usciti 40960000+0 record
```

Lanciando il comando precedente ovvero, estendendo il dd a tutta la partizione (20GB) ho ottenuto qualcosa del genere :

```
Input/Output Error
entrati 40965751+0 record
usciti 40965750+0 record
```

Il che mi ha fatto pensare a problemi alla fine del file e arrotondare il tutto come nel passaggio precedente. Questo ha implicato una perdita di  $5751 * 512 = 2944512$  bytes, ovvero circa 3MB che ho ritenuto in questo contesto trascurabili senza troppi indugi.

Il secondo passo è definire in modo corretto il file drbd.conf che dovrebbe essere sulla /etc. Per i nostri scopi avere una sola sezione resources chiamata ad esempio drbd0 con dei blocchi siffatti:

```
on proxycl1 {
device /dev/drbd0;
disk /dev/sda4;
address 192.168.0.1:7788;
meta-disk internal;
}
```

```
on proxycl2 {
device /dev/drbd0;
disk /dev/sda2;
address 192.168.0.2:7788;
meta-disk internal;
```

```
}
```

...in cui diciamo che le due machine 192.168.0.1 e 2 dialogano attraverso la porta 7788 e condividono il device /dev/drbd0 mettendo in RAID 1 i volumi /dev/sda4 e /dev/sda4. Inoltre modifichiamo per avere una maggiore velocità le specifiche di default con le seguenti:

```
net
max-buffers 4096;
max-epoch-size 2048;
```

```
syncer
rate 125M;
nr_requests=128;
```

Il device simbolico corrispondente al RAID1 dei due HD fisici ovvero /dev/drbd0 ancora non esiste e parimenti bisogna che esista un file drdb0 nella directory /dev. Il tutto si fa con questo comando:

```
mknod -m 0660 /dev/drdb0 b 147 0
```

lanciato su entrambe le macchine. Che vuol dire ?

-m 0660 = permessi (sintassi tipo chmod 6 = r+w)  
B = tipo file, a blocchi  
147 0 = numero primario e secondario, sinceramente non so che significa ...

Si tratta ora di sincronizzare le due risorse. Per crearlo facciamo partire drbd su entrambi i nodi con:

```
/etc/init.d/drdb start
```

...e sul server forziamo la sincronia e la costruzione dell'array con:

```
drbdadm -- --do-what-I-say primary all
```

Nota: si deve proprio scrivere "--do-what-I-say" (con la I maiuscola!) perché significa dire all'array "sincronizzati, fai come ti dico io anche se trovi i tuoi dati (inizialmente) incoerenti". Nota inoltre che se l'installazione di DRBD è stata fatta correttamente cercando una lista dei moduli presenti lo si deve trovare presente:

```
# lsmod | grep drbd
drbd          142668  1
```

Su ogni macchina può essere monitorato l'avanzamento delle operazioni con il comando `cat /proc/drbd` ottenendo messaggi di questo tipo (in questo caso riferiti a quanto fatto sul server):

```
[root@proxycl1 root]# cat /proc/drbd
version: 0.7.22 (api:79/proto:74)
SVN Revision: 2554 build by root@proxycl1, 2006-12-15 11:42:53
0: cs:SyncSource st:Primary/Secondary ld:Consistent
   ns:234120 nr:0 dw:0 dr:246280 al:0 bm:13 lo:1473 pe:416 ua:3040 ap:0
   [>.....] sync'ed: 1.4% (16639/16866)M
   finish: 0:59:09 speed: 4,476 (2,868) K/sec
```

**Attenzione** : se i tempi di presunta fine sono altissimi (es. 130 ore, a me era successo ☺) vuol dire che il file hosts su entrambe le macchine era configurato male ed in particolare non c'erano due record per macchina (es. un solo record con 192.168.0.x proxyclx x=1,2) il che costringeva ad utilizzare la stessa scheda di rete per i messaggi di dati e di sincro, il che dilata enormemente i tempi.

Completata la sincronizzazione e' necessario procedere alla formattazione della partizione condivisa, sul nodo primario (da cui avevamo mappato la /dev/drbd0 con l'mknod) eseguire questo comando:

```
# mkfs.ext3 /dev/drbd0
```

Se le cose sono andate bene otteniamo un output del genere:

```
mke2fs 1.32 (09-Nov-2002)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
2545920 inodes, 5087950 blocks
254397 blocks (5.00%) reserved for the super user
First data block=0
156 block groups
32768 blocks per group, 32768 fragments per group
16320 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
    4096000
```

```
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done
```

This filesystem will be automatically checked every 37 mounts or 180 days, whichever comes first. Use tune2fs -c or -i to override.

Verifichiamo che il sistema è in grado di montare il volume che abbiamo creato:

```
# mount /dev/drbd0 /replicated
# mount
/dev/sda2 on / type ext3 (rw)
none on /proc type proc (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/sda1 on /boot type ext3 (rw)
none on /dev/shm type tmpfs (rw)
/dev/drbd0 on /replicated type ext3 (rw)
```

Se il servizio drbd è sempre attivo dovremmo ottenere a primario:

```
# drbdadm state all
Primary/Secondary
```

...e viceversa a secondario.

I due nodi dovrebbero ora essere sincronizzati, ossia avere lo stesso contenuto nella /replicated, montata dal primario. Se i contenuti delle due /replicated sono diversi, per forzare la risincronizzazione dal primario (ossia dalla macchina che risponde Primary/Secondary dal comando drbdadm state all) diamo i seguenti comandi:

```
# mount /dev/drbd0 /replicated
# drbdadm invalidate_remote all
```

Se per caso la device è già montata chiaramente il primo comando dà un messaggio di errore. Con il solito cat /proc/drbd dovremmo vedere un avanzamento, ossia un qualcosa del genere:

```
[root@proxycl1 root]# cat /proc/drbd
version: 0.7.22 (api:79/proto:74)
SVN Revision: 2554 build by root@proxycl1, 2006-12-15 11:42:53
0: cs:SyncSource st:Primary/Secondary ld:Consistent
   ns:36224 nr:0 dw:352570 dr:753455 al:0 bm:1579 lo:0 pe:4129 ua:2288 ap:0
   [>.....] sync'ed: 0.2% (19855/19874)M
   finish: 1:41:39 speed: 3,284 (1,156) K/sec
```

```
[root@proxycl1 root]# cat /proc/drbd
version: 0.7.22 (api:79/proto:74)
SVN Revision: 2554 build by root@proxycl1, 2006-12-15 11:42:53
0: cs:SyncSource st:Primary/Secondary ld:Consistent
   ns:69504 nr:0 dw:352570 dr:786163 al:0 bm:1581 lo:0 pe:4128 ua:2145 ap:0
   [>.....] sync'ed: 0.3% (19823/19874)M
   finish: 2:00:49 speed: 2,756 (1,824) K/sec
```

NB: il cs è lo stato che viene mostrato col comando drbdadm cstate all, mentre lo st è quello mostrato con drbdadm state all. Se lanciamo lo stesso comando da secondario otteniamo cs: SyncTarget e st:Secondary/Primary.

Bisogna ora fare in modo che drbd sia avviato allo startup nei due nodi. Nella nostra distribuzione di Linux questo si può fare abilitandolo per i runlevel 3,4,5 con:

```
chkconfig --level 345 drbd on
```

Infine occorre inserire in crontab il comando che permette di rinegoziare la connessione tra i nodi in caso di disconnessione (reboot, sostituzione disco fisico etc). Il tutto si fa digitando crontab -e e poi nel file che si apre in modo VI aggiungere questa riga:

```
*/10 * * * * drbdadm connect all
```

Il comando in questione viene lanciato ogni 10 minuti (\* / 10 = tutti i minuti saltando di 10 in 10).

Con la configurazione in questione abbiamo creato il RAID-1 via rete per cui possiamo fare un po' di prove. Ad esempio accendendo e spegnendo server. Consideriamo che per funzionare DRBD ha bisogno che uno dei due server (quello che in quel momento è primario) monti in RW la /dev/drbd0

sulla /replicated (l'altro vedrà la propria modificarsi in sincronia con questa). Per cui per fare queste prove è necessario inserire un comando di mount allo startup.

### DBRD : testing

Finita la sincronizzazione, ho a primario:

```
[root@proxycl1 init.d]# df
Filesystem      blocchi di 1K  Usati Disponib.  Uso%  Montato su
/dev/sda2        20161204  2708164  16428900  15% /
/dev/sda1         101089    15562    80308  17% /boot
none             512144      0  512144  0% /dev/shm
/dev/drbd0       20032180   32828  18981764  1% /replicated
```

Nessun device /dev/drbd0 è invece montato dal secondario.

Se entrambi i sistemi vengono riavviati, si può verificare che sono tutti e due secondari e nessuno può fare il mount. Se facciamo diventare ora proxycl2 primario con drbdadm primary all si può verificare che ora può montare il device in questione, ottenendo la situazione precedente (ma a macchine invertite).

### HeartBeat : prima parte, il demone

Per capire bene come stanno le cose teniamo presente questo (preso da <http://wiki.linux-ha.org/DataRedundancyByDrbd>)

Up to now we only replicate the data. If one node fails, we need manual intervention. To automate this, you want to have a cluster manager disk and executable monitoring (daemon) process running: heartbeat ...

Cosa aggiunge quindi heartbeat?

- Automatizzare le transizioni primario-secondario.
- Il fatto di gestire dei demoni

Per configurare heartbeat occorre agire sui seguenti files:

- **/etc/ha.d/ha.cf**
- **/etc/ha.d/haresources**
- **/etc/ha.d/authkeys**

Questi files non vengono posizionati dall'installazione nelle directory di cui sopra. Per trovarne i modelli (dei primi due) andare nella sottocartella "doc" (ovvero /usr/src/heartbeat-2.0.7/doc in questo caso) e lanciare questo comando:

```
cp ha.cf /etc/ha.d/ha.cf
cp haresources /etc/ha.d/haresources
```

Riguardo al primo file lo scriviamo così:

```
debugfile /var/log/hadebug
logfile /var/log/halog
logfacility daemon
keepalive 1
deadtime 40
warntime 7
initdead 120
udpport 694
baud 115200
serial /dev/ttyS0
bcast eth0
auto_failback off
#stonith wti_nps /etc/ha.d/conf/wti.conf
node proxycl1
node proxycl2
ping 172.16.250.254
respawn hacluster /usr/lib/heartbeat/ipfail
```

Commenti :

- Le prime righe dicono di utilizzare il meccanismo di archiviazione dei log daemon (predefinito per l'uso dei demoni, altrimenti si ottiene uno warning) e dove mettere i log.
- Il battito del cuore viene sentito dalla seriale /dev/ttyS0.
- Baud impostati sono quelli indicati in options.ttyS0 (vedi configurazione collegamento seriale)
- La sincronizzazione si fa dalla eth0 cioè dalla rete "privata" fra le due macchine.
- Autofailback off significa che la risorsa preferenziale non si riappropria automaticamente del suo ruolo una volta andata giù (e quindi una volta ceduto il suo ruolo all'altra) e ritornata su.
- respawn serve per ipfail. Ovvero un nodo è considerato UP se riesce a pingare l'indirizzo in questione (172.16.250.254 nel nostro caso).
- Notare che hacluster deve essere un utente presente in /etc/passwd (di norma è messo automaticamente dagli RPM).
- Per ora lasciamo la configurazione di Stonith commentata, la rivediamo meglio più avanti.

Sul secondo file invece scriviamo questo:

```
proxycl1 drbddisk::r0
proxycl1 IPaddr::172.16.250.103
proxycl1 Filesystem::/dev/drbd0::/replicated
proxycl1 command.sh
```

Teniamo presente che, di fatto, nel file in questione si fa riferimento a script presenti in /etc/ha.d/resource.d. Lo script command.sh sarà posto nella stessa cartella e sarà siffatto:

```
#!/bin/sh
service named stop
service named start
service squid stop
service squid start
```

Commenti: sono attribuite al server preferenziale un serie di "risorse". In questo caso:

- La gestione della risorsa r0 (quindi deve esserci un demone DRBD che sta girando).

- L'indirizzo IP condiviso.
- I demoni che deve gestire.
- Su quale punto del filesystem andare ad innestare la /dev/drbd0

Alcune osservazioni:

Con le operazioni precedenti ci siamo posti nella situazione di primario con DRBD in esecuzione e stato Primary/Secondary. Per cui lo start di Heartbeat (dal primario) con quella configurazione può avere successo perché la partizione è montata, mentre a secondario avrà lo stesso successo anche se questa macchina non potrà correttamente avviare le risorse che coinvolgono la replicated.

Se in primario va giù Heartbeat gestirà il passaggio della /replicated da un sistema all'altro e tutto andrà pure bene.

**Se però vanno giù entrambi i server** per il ripristino del cluster occorre, a causa di quanto fatto, procedere così:

- Togliere Heartbeat dall'avvio automatico del primario
- Accendere il primario
- Verificare che questo sia in stato Primary
- Fare il mount della /replicated
- Avviare Heartbeat
- Rimettere Heartbeat dall'avvio automatico del primario
- Riaccendere il secondario

Il file haresources file deve essere **identico** per entrambi i server anche se si nomina solo il primario. Sarà compito di Heartbeat utilizzare il secondario se questo cade. Nella documentazione infatti si dice che deve essere presente una riga per ogni nodo preferenziale e non per ogni nodo!

Infine in /etc/ha.d/authkeys scriviamo:

```
auth 1
1 sha1 ccf5d63d9957b4fab4206732bf3671a39b81290f
```

La chiave privata ccf5d63d9957b4fab4206732bf3671a39b81290f può essere generata con questo comando:

```
dd if=/dev/urandom count=4 2>/dev/null | openssl dgst -sha1
```

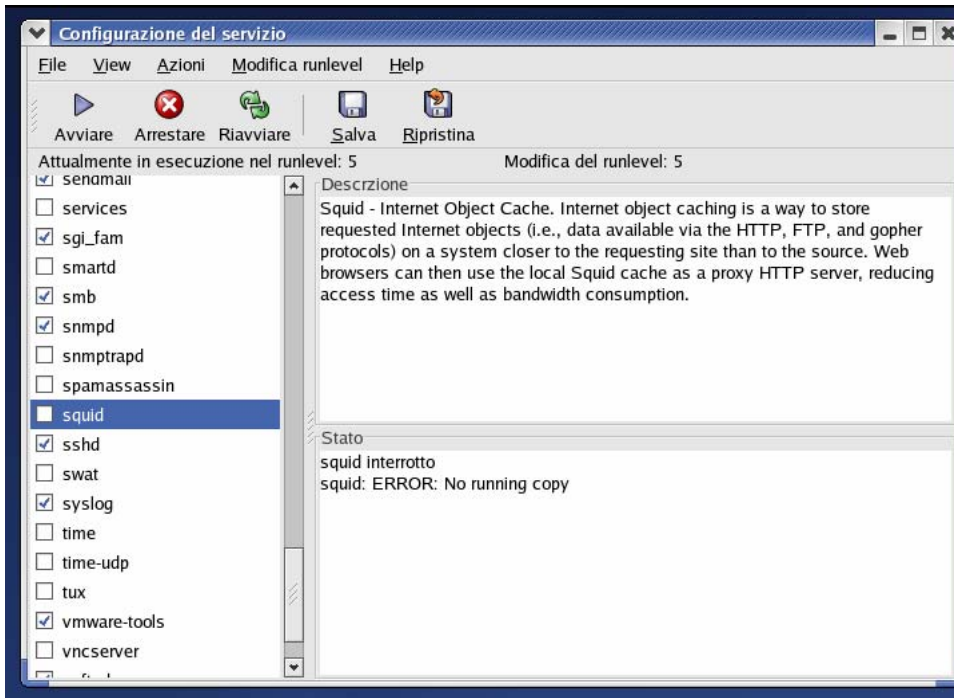
Diamo poi al nostro file i diritti necessari con il comando:

```
chmod 600 /etc/ha.d/authkeys
```

## HeartBeat : parte seconda, i servizi (clusterizzati e gestiti da H.)

Come abbiamo detto dobbiamo di configurare i servizi che devono essere ridondati in modo che le cartelle dei servizio che contengono i dati siano presenti nella cartella condivisa (/replicated) del nostro primario (proxyc11).

Nel caso in esame si tratta innanzitutto di **ridondare i demoni** squid e named. Dunque: innanzitutto eliminiamo questi ultimi dall'avvio automatico di sistema (entrambe le macchine) : ad esempio con l'interfaccia grafica operiamo così:



**Abbiamo affidato questi demoni alla gestione di Heartbeat attraverso lo script command.sh.**

A questo punto si tratta di avere i dati clusterizzati. Si preparano sulla /replicated (del primario) tutte le cartelle necessarie, ad esempio:

```
/replicated
  /etc/                named.conf
  /etc/squid           file ACL e pagine html per errori
  /var/log/squid       cartella per i log di squid (inizialmente vuota)
  /var/named           zone named
```

Si cancellano poi i corrispondenti file/cartelle sostituendoli con collegamenti a questi. Così, diamo questi comandi:

```
# cd /etc
# rm named.conf
# ln -s /replicated/etc/named.conf
# rm -r squid
# ln -s /replicated/etc/squid
```

```
# cd /var/log
# rm -r squid
# ln -s /replicated/var/log/squid
# cd /var
# rm -r named
# ln -s /replicated/var/named
```

Notare che nella /etc/squid originaria era presente un link di nome errors alla /usr/share/squid/errors/English. I file qui presenti sono stati copiati nella cartella /replicated/etc/squid che contiene oltre agli \*.acl anche una cartella fisica errors contenente quanto originariamente era in /usr/share/squid/errors/English. Pertanto dire che /etc/squid è un collegamento simbolico alla /replicated/etc/squid, per come l'abbiamo costruita dovrebbe mettere tutto a posto.

```
[root@proxycl1 etc]# ls -lo sq*
lrwxrwxrwx  1 root          21 28 dic 16:16 squid -> /replicated/etc/squid
```

Anche nel secondario bisogna predisporre i collegamenti senza curarsi che al momento saranno floating (visto che il volume /dev/drbd0 può essere montato solo dal primario).

Si tratta ora di ridondare le operazioni batch. Nella versione non clusterizzata il server in questione provvedeva anche alle seguenti operazioni batch:

- inserimento di route statiche all'avvio (script ipforward)
- archiviazione in una cartella apposita dei log di squid in modo da salvare queste informazioni prima che la rotazione dei log le sovrascrivesse.

Per **clusterizzarle adeguiamo innanzitutto la** cartella /replicated in questo modo:

```
/replicated
/etc/          named.conf
/etc/squid    file ACL
/root/bin     log_backup_cl.sh
/var/log/squid  cartella per i log di squid (inizialmente vuota)
/var/named    zone named
/home/backup  cartella per l'archiviazione dei log/backup (inizialmente vuota)
```

L'acquisizione delle route è di fatto simile al file hosts locale che è stato duplicato. Per questo il problema è stato risolto utilizzando due file ipforward presenti nella cartella non replicata /etc/init.d di entrambi i sistemi. Ossia si lavora come nella versione non clusterizzata.

Lo script rc.local (/etc/rc.d/rc.local) messo coi permessi corretti sarà quindi siffatto:

```
touch /var/lock/subsys/local
/replicated/etc/init.d/ipforward
```

Il backup è un'operazione periodica e non da effettuare all'avvio, quindi è da inserire nella crontab, che non è clusterizzata. Nella crontab -e del primario si opera così:

```
*/10 * * * * /sbin/drbdadm connect all
00 05 * * 0 /replicated/root/bin/log_backup_cl.sh
```

Si fa riferimento ad uno script posto nel volume condiviso. Allora, lo script in questione fa, in breve questo:

```
#!/bin/sh
LOGBKDIR=/replicated/home/backup
DATE=`date +%Y-%m-%d`
if [ -f /replicated/var/log/squid/access.log.1.gz ]
then
  [ ! -d $LOGBKDIR/squid ] && mkdir $LOGBKDIR/squid
  if [ ! -e /replicated/var/log/squid/ok.$DATE.txt ]
  then
    cp -p /replicated/var/log/squid/access.log.1.gz $LOGBKDIR/squid/access.log.$DATE.gz
    cp -p /replicated/root/bin/ok.txt $LOGBKDIR/squid/ok.$DATE.txt
  fi
fi
```

Il nodo secondario lancerà lo stesso script ma nella crontab –e avrà questo:

```
*/10 * * * * /sbin/drbdadm connect all
00 06 * * 0 /replicated/root/bin/log_backup_cl.sh
```

Il tutto vuol dire che alle 5 del mattino di tutte le domeniche il primario lancia lo script. Se va bene crea un file gz nella /replicated/home/backup/squid e ci mette un file il cui contenuto è irrilevante ed il cui nome è del tipo ok.YYYY.MM.DD.txt.

Un'ora dopo il secondario lancia lo stesso script. Se il precedente è andato bene (presenza di un file ok alla stessa data) non fa niente. Altrimenti provvede lui ad archiviare i log.

Si presuppone che nessuno navighi dalle 5 alle 6 di domenica mattina : nel caso è bene configurare le access list per lo scopo.

## Heartbeat : parte terza, il primo avvio e l'automatizzazione

Fatto questo avviamo Heartbeat utilizzando per questa prima volta la procedura manuale. Lanciamo in sostanza su proxycl1 il seguente comando:

```
#!/etc/init.d/heartbeat start
```

Lanciamo poi su questa macchina i seguenti controlli:

- ifconfig : possesso dell'IP 172.16.250.103
- mount : presenza della partizione condivisa (/dev/drbd su /replicated)
- ps o top per vedere se è in esecuzione uno squid

Se non danno esito positivo la "colpa" è, molto probabilmente, del file haresources. Vediamo i risultati di queste verifiche:

### ifconfig

```
eth1  Link encap:Ethernet  HWaddr 00:14:5E:82:30:06
inet addr:172.16.250.101  Bcast:172.16.255.255  Mask:255.255.0.0
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:6537 errors:0 dropped:0 overruns:0 frame:0
TX packets:629 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:803033 (784.2 Kb)  TX bytes:82728 (80.7 Kb)
Interrupt:16
```

```
eth1:0 Link encap:Ethernet  HWaddr 00:14:5E:82:30:06
inet addr:172.16.250.103  Bcast:172.16.255.255  Mask:255.255.0.0
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
Interrupt:16
```

### mount

```
/dev/sda2 on / type ext3 (rw)
none on /proc type proc (rw)
none on /dev/pts type devpts (rw,gid=5,mode=620)
usbdevfs on /proc/bus/usb type usbdevfs (rw)
/dev/sda1 on /boot type ext3 (rw)
none on /dev/shm type tmpfs (rw)
/dev/drbd0 on /replicated type ext3 (rw)
```

### ps -edaf

```
named    3250    1  0 12:35 ?        00:00:00 /usr/sbin/named -u named
squid    3509  3507  0 12:52 ?        00:00:00 (squid) -D
squid    3511  3509  0 12:52 ?        00:00:00 (unlinkd)
```

Nota:

ho ottenuto il seguente errore su squid, probabilmente dovuto al fatto di aver creato la /var/log su /replicated e non aver messo correttamente i permessi:

```
Cannot open '/var/log/squid/access.log' for writing. The parent directory must be writeable by the user 'squid', which is the cache_effective_user set in squid.conf.
```

Risolto coi seguenti comandi:

```
# cd /replicated/var/log  
# chmod 777 squid
```

Configuriamo ora HeartBeat per l'avvio in automatico su entrambe le macchine ed il processo è concluso. Nella nostra distribuzione di Linux questo si può fare con il solito comando dato a DRBD (di cui Heartbeat ha bisogno!) che lo abilita ai runlevel 3,4,5:

```
chkconfig --level 345 heartbeat on
```

Il sistema è già in grado di funzionare. Fra i miglioramenti da apportare alle configurazioni di base c'è ad esempio la configurazione dell'utility Stonith. Punto debole di un cluster come quello in esame è una situazione come questa:

- I due nodi perdono la comunicazione tra loro
- Il nodo primario crede di essere solo e rimane primario
- Il nodo secondario crede di essere solo e ... diventa primario
- Due primari sulla rete offrono lo stesso servizio e scrivono sul device DRBD locale
- A parte i problemi derivanti dall'aver due IP uguali sulla rete, se i due nodi ad un certo punto si vedono nuovamente i dati vengono corrotti.

Per scongiurare una cosa del genere si ricorre all'utility stonith di Heartbeat che in sostanza che permette di spegnere il nodo antagonista quando si presentano due primari: il primo che riesce a spegnere l'altro nodo vince il duello e rimane l'unico primario. L'altro nodo quando ripartirà troverà il primo nodo funzionante e si unirà al cluster.

Dunque, vediamo i device supportati:

```
# stonith -L  
apcmaster  
apcmastersnmp  
apcsmart  
baytech  
cyclades  
drac3  
external/riloe  
external/ssh  
ibmhmc
```

meatware  
null  
nw\_rpc100s  
rcd\_serial  
rps10  
ssh  
suicide  
wti\_nps

Visto che non ho un device hardware da poter utilizzare la scelta cade sulle opzioni evidenziate.

STONITH Device: external/ssh - ssh-based Linux host reset  
Fine for testing, but not suitable for production!  
For more information see <http://openssh.org>  
List of valid parameter names for external/ssh STONITH device:  
    hostlist

STONITH Device: meatware - Human (meatware) intervention STONITH device.  
This STONITH agent prompts a human to reset a machine.  
The human tells it when the reset was completed.  
List of valid parameter names for meatware STONITH device:  
    hostlist

STONITH Device: ssh - SSH-based Linux host reset  
Fine for testing, but not suitable for production!  
For more information see <http://openssh.org>  
List of valid parameter names for ssh STONITH device:  
    hostlist

STONITH Device: suicide - Virtual device to reboot/powerdown itself.  
List of valid parameter names for suicide STONITH device:

Allora, aggiungiamo al nostro ha.cf il seguente codice:

```
stonith meatware /etc/ha.d/splugin.cfg
```

e nella /etc/ha.d deve esserci presente un file “splugin.cfg” con sintassi compatibile col plugin chiamato.

Nota : in precedenza questa parte era stata commentata, così non c’era bisogno di aggiungere il file .cfg. Ovviamente se aggiungiamo una riga che chiama Stonith che fa riferimento ad un file di configurazione bisognerà che questo ci sia...

In questo caso, in attesa di testare meglio il sistema utilizziamo la strategia meatware ossia:

I don't want Heartbeat to fail over the cluster automatically. How can I require human confirmation before failing over?

- You configure a "meatware" [STONITH](#) device into the [ha.cf](#) file. The meatware STONITH device asks the operator to go power reset the machine which has gone down. When the operator has reset the machine he or she then issues a command to tell the meatware STONITH plugin that the reset has taken place. Heartbeat will wait indefinitely until the operator acknowledges the reset has occurred. During this time, the [resources](#) will not be taken over, and nothing will happen.

Nel file di configurazione deve essere presente solo l'host da riavviare, ossia il duale. Nel caso del "primario" dunque abbiamo:

```
proxycl2
```

Nel caso del secondario il nome duale (proxycl1) sempre riferito alla stessa interfaccia (eth1 in questo caso, quella riferita alla stessa classe 172.16.250.x a cui si riferisce l'IP condiviso). In questo caso il cluster lavorerà con prestazioni "degradate" fino a che manualmente un operatore non riattiverà il secondo nodo. Direi che sia è il modo più sicuro (e anche meno automatico) per non avere casini e va completato con un eventuale programmino che fa il ping dei due server in questione ...

Il plugin che viene chiamato nella direttiva precedente è in realtà il nome di uno script o programma che si trova nella cartella `/usr/lib/stonith/plugins`. Sotto `external` ci sono alcuni esempi di script per potersi eventualmente scrivere i propri plugin (a questo si riferiscono le necessità di avere le operazioni di `reset` etc.. di cui nella documentazione).

### Note

Nella documentazione di Stonith si parla di cluster R1 ed R2. Per sapere in quale caso ci troviamo teniamo presente quanto segue:

Configuring release 2 software to get these new release 2 capabilities is similar to that of release 1, except that the [haresources](#) file has been replaced with a common [XML](#)-based [cluster](#)-wide configuration file, which is managed by the [ClusterInformationBase](#) ([CIB](#)) component.

In questo caso il file `haresources` non ha formato XML quindi anche se utilizziamo Heartbeat 2 utilizziamo la modalità di configurazione R1, ossia quella di Heartbeat 1.0.

Per testare se la configurazione data è corretta si può fare come segue:

```
# stonith -l -t meatware -F /etc/ha.d/conf/splugin.cfg
** INFO: parse config info info=proxycl2
proxycl2
```

(prova riferita all'ipotesi di macchina corrente = proxycl1).

## Post Installazione : Test, Tuning e Varie

In questa sezione riporto tutto quanto relativo a test, ottimizzazioni sulla rete e così via.  
Riavviamo ora le due macchine (prima il primario, poi il secondario) e verifichiamo innanzitutto la correttezza della situazione iniziale:

### Proxyc1

```
[root@proxyc1 conf]# cat /proc/drbd
version: 0.7.22 (api:79/proto:74)
SVN Revision: 2554 build by root@proxyc1, 2006-12-15 11:42:53
0: cs:Connected st:Primary/Secondary Id:Consistent
   ns:0 nr:0 dw:1928 dr:690773 al:0 bm:324 lo:0 pe:0 ua:0 ap:0
[root@proxyc1 conf]# /etc/init.d/heartbeat status
heartbeat OK [pid 2472 et al] is running on proxyc1 [proxyc1]...
```

### Proxyc2

```
[root@proxyc2 root]# cat /proc/drbd
version: 0.7.22 (api:79/proto:74)
SVN Revision: 2554 build by root@proxyc2, 2006-12-19 10:45:07
0: cs:Connected st:Secondary/Primary Id:Consistent
   ns:0 nr:0 dw:679264 dr:0 al:0 bm:184 lo:0 pe:0 ua:0 ap:0
[root@proxyc2 root]# /etc/init.d/heartbeat status
heartbeat OK [pid 2471 et al] is running on proxyc2 [proxyc2]...
```

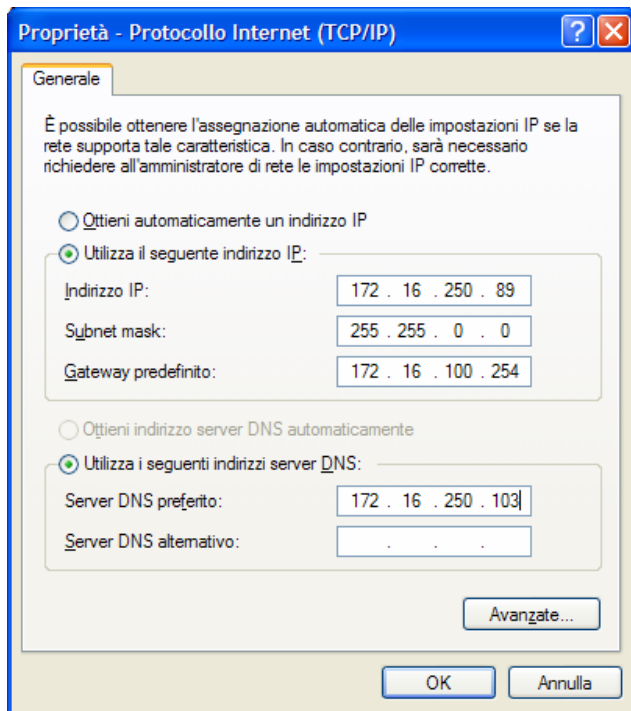
Sul resolv.conf dei due proxy avremo messo:

```
nameserver 172.16.250.103
```

Nota che se mettiamo al posto dell'indirizzo IP condiviso in forma numerica il corrispondente nome (proxyc1cluster) specificato nei due files \etc\host il riavvio di squid non ha successo.

Verifichiamo poi che i client possono utilizzare i servizi desiderati, ossia in sintesi che possano andare in internet appoggiandosi al proxy + dns in questione.

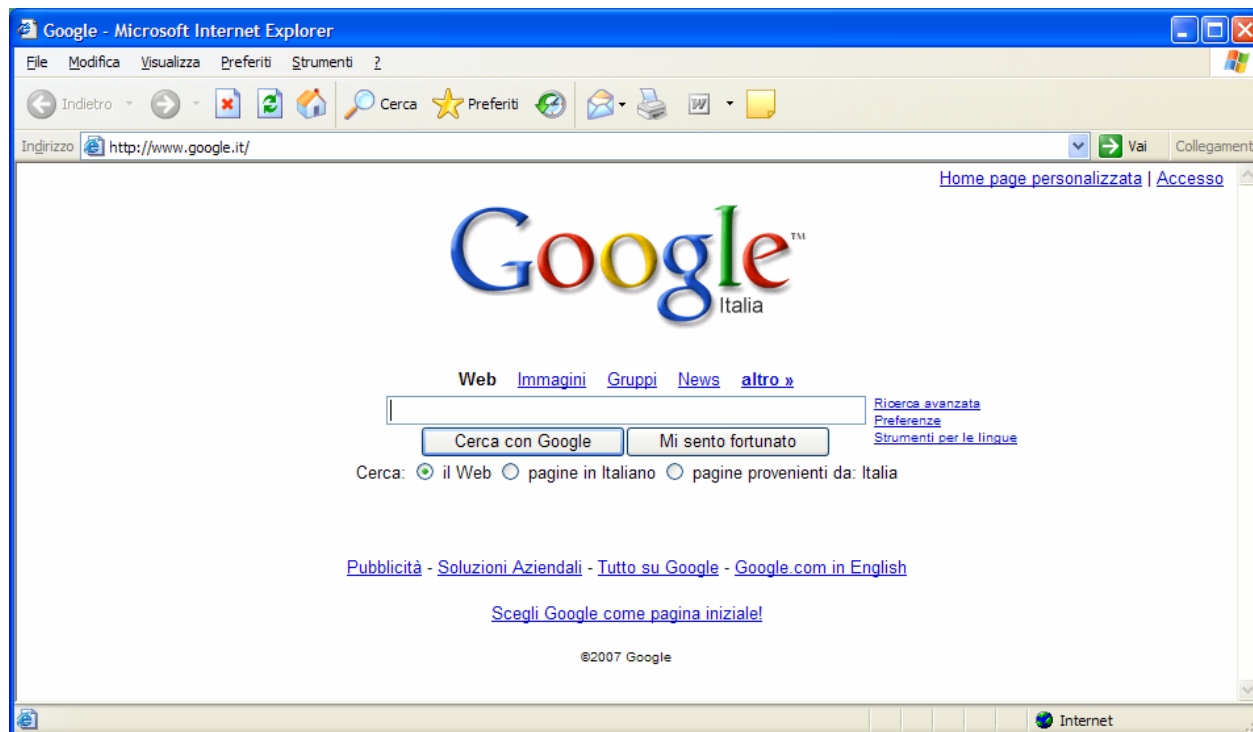
Chiaramente ci riferiremo al suo indirizzo condiviso, quindi sarà:



Affinché i client potessero navigare è stato necessario abilitare non solo l'indirizzo IP condiviso (come sarebbe sembrato logico) ma anche i singoli IP. Quindi sarà presente nella configurazione del firewall un paragrafo siffatto:

- Permettere l'uscita in UDP/53 a 172.16.250.101, 172.16.250.102, 172.16.250.103 (Query a DNS esterno);
- Permettere l'uscita in TCP/ANY a 172.16.250.101, 172.16.250.102, 172.16.250.103 (Query a DNS esterno)

Lanciando un URL da un client verifichiamo che tutto è ok:



Adesso facciamo lo stop del primario e verifichiamo che tutto funziona ancora. Diamo il seguente comando:

```
[root@proxycl1 ha.d]# /etc/init.d/heartbeat stop
Stopping High-Availability services:
Killed
```

Dato questo comando il primario fa reboot in modo automatico. A secondario abbiamo:

```
[root@proxycl2 ha.d]# cat /proc/drbd
version: 0.7.22 (api:79/proto:74)
SVN Revision: 2554 build by root@proxycl2, 2006-12-19 10:45:07
0: cs:WFCConnection st:Primary/Unknown Id:Consistent
   ns:0 nr:0 dw:675071 dr:825 al:0 bm:159 lo:0 pe:0 ua:0 ap:0
[root@proxycl2 ha.d]# /etc/init.d/heartbeat status
heartbeat OK [pid 2762 et al] is running on proxycl2 [proxycl2]...
```

Quindi è diventato primario e ha heartbeat che gira. Verifichiamo infine che navighiamo ancora: OK!

Rotazione dei log: impostamo settimanale

```
# see "man logrotate" for details
# rotate log files weekly
```

```
weekly
# keep 4 weeks worth of backlogs
rotate 4

# create new (empty) log files after rotating old ones
create
# uncomment this if you want your log files compressed
#compress
# RPM packages drop log rotation information into this directory
include /etc/logrotate.d
# system-specific logs may be also be configured here.
/var/log/halog {
    rotate 1
    postrotate
        /sbin/killall -HUP syslogd
    endscrip
}
/var/log/hadebug {
    rotate 1
    postrotate
        /sbin/killall -HUP syslogd
    endscrip
}
```