ISTITUTO NAZIONALE DI FISICA NUCLEARE

Sezione di Bari

INFN/code-xx/xxx
3 giugno 2006

CCR-02/2006/P

HIGH AVAILABILITY A BASSO COSTO

Domenico Diacono, Sabino Calò

INFN-Sezione di Bari, Via E. Orabona, 4, I-70126 Bari, Italy

Abstract

Lo scopo di questo progetto è quello di portare in produzione servizi erogati in Alta Disponibilità, al fine di ridurre il tempo di interruzione del servizio allo 0,01%. La caratteristica peculiare rispetto ad altri progetti che ottengono lo stesso risultato è l'utilizzo di componenti hardware e software standard, facilmente reperibili e generalmente più economici.

Workshop CCRWS06 – Otranto (LE)

Pubblicato da ***SIS–Pubblicazioni***
Laboratori Nazionali di Frascati

1 INTRODUZIONE

La necessità di fornire con continuità i servizi informatici fondamentali può essere soddisfatta in vari modi. Ad esempio è possibile duplicare i servizi su più macchine indipendenti, o realizzare cluster di macchine capaci di erogare lo stesso servizio e che condividono uno storage esterno. Ambedue le soluzioni hanno vari svantaggi, come la necessità di moltiplicare gli sforzi amministrativi o di acquistare hardware proprietario

In queste pagine viene illustrato un metodo che, utilizzando hardware a basso costo e sistemi software openSource, permette di gestire uno o più servizi in alta disponibilità, con l'obiettivo di avere cioè tempi di indisponibilità del servizio dell'ordine dello 0.01% del tempo totale di attività.

Uno dei requisiti essenziali per la erogazione di un servizio in alta disponibilità è naturalmente la gestione opportuna della alimentazione elettrica. In assenza di un gruppo di continuità e di un impianto elettrogeno autonomo si può migliorare il tempo di disponibilità del servizio, ovviando a guasti hardware e software della macchina, ma certamente non si può garantire la continuità voluta.

Si tratta quindi di valutare caso per caso, in relazione alle necessità connesse al servizio in questione e alle disponibilità economiche, l'opportunità di realizzare un cluster in alta disponibilità: la procedura qui descritta, semplice e poco costosa, potrebbe fornire una soluzione alla portata di molte strutture.

2 STRUTTURA DEL CLUSTER

La struttura del cluster viene illustrata nella figura seguente:

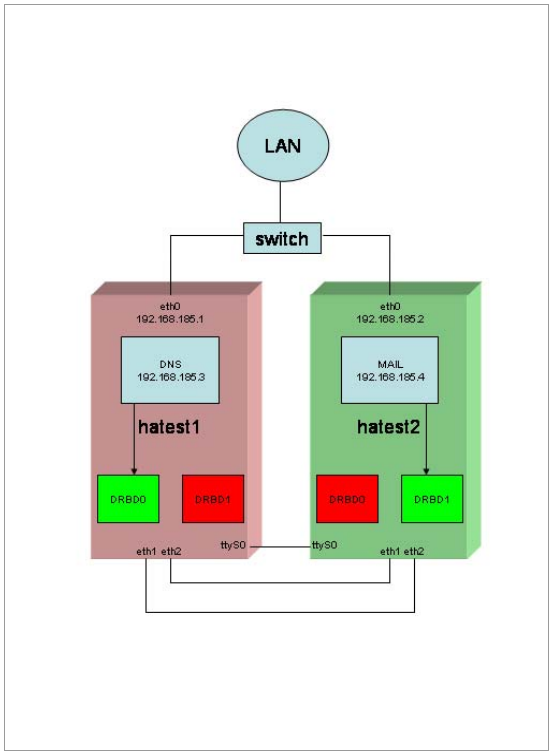

FIG. 1: Struttura del cluster

I requisiti hardware fondamentali che i PC devono possedere per poter essere utilizzati come membri del cluster sono i seguenti:

1. Sistema di protezione del disco. Nel caso in esame si è utilizzato un controller RAID 5 hardware, ma potrebbe essere sufficiente il ricorso al RAID 1 software tra due dischi EIDE montati su due canali distinti. La scelta della soluzione da adottare dipende ovviamente dal budget disponibile e dalle caratteristiche del servizio da erogare.
2. Tre schede di rete, di cui almeno due con velocità 1Gbit/sec per la comunicazione interna tra i nodi.
3. Una porta seriale per la comunicazione interna.

Una delle interfacce ethernet, la eth0, è raggiungibile dall'esterno, nel caso in figura con indirizzo di rete 192.168.185.1 e 192.168.185.2 rispettivamente per il primo (hatest1) e il secondo (hatest2) nodo, le altre due interfacce servono per la comunicazione interna. I due nodi sono collegati tra loro anche da un cavo null modem sulla interfaccia seriale ttyS0.

Su ogni nodo nel cluster illustrato nella figura viene eseguito un diverso servizio: hatest1 funziona da DNS e hatest2 da Mail server. Il DNS usa lo spazio disco contrassegnato come DRBD0, e risponde all'indirizzo 192.168.185.3, il mail server usa lo spazio DRBD1 e risponde a 192.168.185.4. Nel seguito ci si riferirà al solo servizio mail e la configurazione in cluster del secondo servizio è simmetrica.

I due device DRBD sono dei block device contenenti i dati sui quali lavora il servizio, che vengono replicati via rete tra i due nodi. Uno solo può montarli, quello che in quel momento eroga il servizio.

Si noti come l'indirizzo dei servizi è diverso da quello delle interfacce di rete pubblica dei nodi, perchè i servizi stessi devono essere raggiungibili su entrambe le macchine. La realizzazione della alta disponibilità si basa infatti sulla possibilità che ogni servizio ha di migrare da un nodo all'altro nel caso di guasto hardware o blocco software. Così facendo si sposta non solo il servizio vero e proprio, ad esempio il demone DNS, ma anche l'indirizzo IP sul quale risponde, lo spazio disco e i file che utilizza. In realtà lo spazio disco è fisicamente già presente sul nodo in standby e sincronizzato con quello del nodo attivo tramite il software DRBD (vedi seguito), la migrazione fa si che il nodo acquisisca il diritto di montarlo, utilizzarlo e diventare attivo. Per sapere se è il momento di migrare un servizio i nodi si controllano a vicenda attraverso tutti i possibili canali di comunicazione a disposizione, accertandosi del loro "stato di salute", e controllando inoltre la effettiva operatività dei servizi a loro assegnati.

Ovviamente ogni nodo deve tranquillamente poter sopportare il carico di tutti i servizi erogati dal cluster: in assenza di questa condizione questo tipo di cluster non ha ragione di esistere, dato che una migrazione dei servizi ne provocherebbe comunque il blocco.

Le versioni del software utilizzato nel seguito sono le seguenti:

- Kernel 2.6.11.7
- DRBD 0.7.10

- Heartbeat 1.2.3

3 LO SPAZIO DISCO E LA RETE INTERNA

Il file system condiviso costituisce il punto nevralgico del sistema. Ogni nodo deve poter avere accesso ai dati, e gli stessi devono al tempo stesso essere protetti dall'accesso contemporaneo dei due nodi, che ne comprometterebbe l'integrità. Nel cluster in esame manca per scelta un sistema separato di storage condiviso: lo spazio su disco è ricavato dai dischi interni ad ogni singolo nodo.

Per realizzare questo spazio condiviso e protetto si è usato il software "Distributed Replicated Block Device" (DRBD). Il funzionamento di DRBD ricorda semanticamente un RAID 1 (mirror) tra due dispositivi di rete. Su ogni nodo si riserva una partizione del disco per la creazione del device condiviso: la copia di DRBD presente localmente legge il file di configurazione /etc/drbd.conf, con le informazioni contenute costruisce il block device /dev/drbd# e decide se il nodo può scrivere o meno su di esso, cioè se è primario o secondario rispetto a tale device.

Se un nodo è primario può montare il device nel file system come una normale partizione e modificarlo. Il lavoro di DRBD consiste nel propagare tutte le modifiche attraverso il collegamento interno anche al secondo nodo. Questo fa si che di tutti i dati siano presenti due copie sincronizzate, sul primo e sul secondo nodo. Quando avviene la migrazione dal primo al secondo nodo del servizio che usa tali dati questo li ritroverà nello stesso stato in cui li ha lasciati prima di migrare.

3.1 Installazione

La rete interna del cluster in un primo momento è configurata in modo che due interfacce di rete, collegate con un cavo incrociato, possano comunicare tra loro. Si è assegnato quindi alla porta eth1 del nodo hatest1 l'indirizzo 192.168.0.1 e alla porta eth2 di hatest2 l'indirizzo 192.168.0.2. Tramite questo collegamento interno avviene la sincronizzazione del device DRBD.

Su ogni nodo è necessario riservare due partizioni del device fisico per la creazione dei block-device condivisi. Tali partizioni non devono comparire all'interno di /etc/fstab, perché non devono essere montate automaticamente dal sistema.

Il file di configurazione /etc/drbd.conf, che va riportato assolutamente identico sui due nodi, è formato da tante sezioni quanti sono i block device da configurare: ogni riga che inizia con "resource" aggiunge un nuovo device al cluster. In ogni sezione la parte contrassegnata da "net" riguarda le impostazioni della rete interna di sincronia. Nelle successive sotto-sezioni, che iniziano con la parola "on" seguita dal nome del nodo che esporta le risorse, si deve indicare quali sono le partizioni ("disk") che andranno a formare il block device ("device"), e su quale indirizzo ("address") e quale porta avverrà la sincronia dei dati. L'indirizzo IP per la sincronia si riferisce alla rete interna.

I nomi dei nodi, nel caso in esame "hatest1" e "hatest2" devono essere quelli restituiti dal comando hostname.

Una volta installato DRBD su ambedue i nodi, e modificato il file di configurazione

secondo le proprie esigenze, è necessario creare i file device con:

#mknod -m 0660 /dev/drbd0 b 147 0

E' quindi possibile avviare il servizio con

#/etc/init.d/drbd start

Se tutto funziona si deve ottenere un risultato di questo tipo:

#cat /proc/drbd
version: 0.7.10 (api:77/proto:74)
SVN Revision: 1743 build by portage@hatest1, 2005-04-06 13:54:30
0: cs:Connected st:Secondary/Secondary ld:Inconsistent
ns:0 nr:0 dw:0 dr:0 al:0 bm:23833 lo:0 pe:0 ua:0 ap:0

Per montare su hatest1 il device come primario e quindi poter creare il file system, è necessario forzare la sincronizzazione della risorsa mail con:

#drbdadm -- --do-what-I-say primary mail

Dopo aver impartito il comando lo stato del sistema sarà il seguente:

#cat /proc/drbd
version: 0.7.10 (api:77/proto:74)
SVN Revision: 1743 build by portage@hatest1, 2005-04-06 13:54:30
0: cs:SyncSource st:Primary/Secondary ld:Consistent
ns:696248 nr:0 dw:0 dr:704440 al:0 bm:42 lo:0 pe:33 ua:2048 ap:0
[>.....] sync'ed: 0.2% (380633/381313)M
finish: 4:55:16 speed: 21,764 (23,204) K/sec

In questo stato il device è già utilizzabile, cioè è possibile creare da hatest1 su/dev/drbd0 il filesystem preferito e montarlo sulla directory /mail. Non è possibile montare contemporaneamente lo stesso device su hatest2: ogni device deve essere montato solo sul nodo primario che lo utilizza. Quando il secondario ne avrà bisogno dovrà essere reso primario, e di questo se ne occuperà il software di gestione del cluster.

3.2 Channel Bonding

I due nodi sono collegati su una rete interna dedicata per due motivi: per permettere la sincronizzazione dei due device condivisi DRBD e per permettere un controllo reciproco affidabile dello stato di salute dei nodi e quindi il failover dei servizi. Tale rete interna, che

nella prima fase della installazione è stata realizzata con un semplice cavo incrociato, riveste dunque una particolare importanza, in quanto la sua continua disponibilità è essenziale per assicurare l'integrità dei dati. Per garantirne la disponibilità anche nel caso di guasto della scheda di rete si è utilizzato il metodo del "channel bonding": due schede di rete per ogni nodo forniscono una sola interfaccia di comunicazione per il kernel, che le usa insieme per l'invio dei pacchetti TCP. Se una delle due linee di comunicazione smette di funzionare, per un guasto ad una scheda di rete o ad un cavo, l'interfaccia rimane inalterata.

Innanzitutto è necessario verificare che nella configurazione del kernel, nella sezione "Network device", sia abilitato il "Bonding device support". Una volta riavviata la macchina con tale opzione abilitata si eseguono i comandi:

#cd /usr/src/linux/Documentation/networking
#gcc -Wall -Wstrict-prototypes -O -I/usr/src/linux/include ifenslave.c -o/sbin/ifenslave

A questo punto è necessario fare in modo che il modulo venga caricato con gli opportuni parametri, modificando il file /etc/modules.conf; ogni distribuzione ha la sua procedura per modificare il file, in Gentoo si procede con:

#nano -w /etc/modules.d/bond
alias bond0 bonding
options bond0 mode=3

#modules-update

dove "mode" indica la modalità di bonding, in questo caso l'uso contemporaneo delle due linee ("fault tolerance"). E' possibile provare la funzionalità da linea di comando:

#ifconfig bond0 192.168.0.1 up
#ifenslave bond0 eth0 eth1

Dopo l'esecuzione di questi comandi deve essere disponibile l'interfaccia bond0, il cui MAC address deve coincidere con quello della prima interfaccia di rete usata nella sua costruzione. Si può controllare lo stato delle interfacce di rete e del bonding con:

#ifconfig

#cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v2.6.1 (October 29, 2004)
Bonding Mode: fault-tolerance (broadcast)
MII Status: up
MII Polling Interval (ms): 0

Up Delay (ms): 0
Down Delay (ms): 0
Slave Interface: eth0
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:30:48:74:xx:xx
Slave Interface: eth1
MII Status: up
Link Failure Count: 0
Permanent HW addr: 00:30:48:74:xx:xy

Il valore "Permanent HW addr" riporta il reale indirizzo MAC delle schede di rete. L'interfaccia bond0 deve essere correttamente configurata al riavvio della macchina; in Gentoo si crea un link in /etc/init.d per l'avvio della interfaccia:

#ln -s /etc/init.d/net.eth0 /etc/init.d/net.bond0

che deve essere configurata nel file /etc/conf.d/net aggiungendo la riga seguente:

iface_bond0="192.168.0.1 broadcast 192.168.0.255 netmask 255.255.255.0"

L'avvio della interfaccia deve essere aggiunto nel runlevel mediante:

#rc-status add net.bond0 default

Si deve creare poi un file di avvio che si occupi della costruzione del channel bonding /etc/init.d/ifenslave, i cui parametri si trovano in /etc/conf.d/ifenslave:

ifenslave="eth0 eth1"
bond="bond0"

e lo si inserisce nel runlevel di default.

Tutte le operazioni vanno ripetute identiche sul secondo nodo, naturalmente usando l'indirizzo 192.168.0.2.

Il firewall deve autorizzare il dialogo sulla interfaccia bond0, e quindi deve essere presente una regola del tipo:

/sbin/iptables -A OUTPUT -o bond0 -j ACCEPT
/sbin/iptables -A INPUT -i bond0 -j ACCEPT

E' possibile ora inserire il servizio drbd nel runlevel, e avviarlo quindi alla partenza

della macchina, con l'avvertenza di farlo partire dopo ifenslave e configurarlo per l'uso della interfaccia bond0.

3.3 LVM

Sul disco del server che si sta realizzando sarà presente la posta elettronica della sezione INFN, un database in continuo aggiornamento e che non si può pensare di spegnere o semplicemente clonare per poterne realizzare il backup. Una possibile soluzione al problema viene dall'uso di LVM, ossia Logical Volume Manager. LVM permette di gestire i volumi logici in Linux mediante un modulo del kernel e una serie di programmi in user-space. La caratteristica di LVM che in questa sede risulta utile è il supporto "snapshot" del filesystem. E' possibile cioè creare una "istantanea" del file system: LVM blocca le operazioni di scrittura e lettura, attraverso il Virtual Filesystem Layer del kernel, in modo da avere uno stato coerente e crea una copia che inizialmente non occupa spazio su disco, perchè punta ai file originari. Dopo aver creato questa copia le operazioni sul filesystem originario riprendono, e l'immagine inizia ad occupare spazio progressivamente con la variazione dei file originari. La copia può quindi essere montata ed utilizzata per il backup, senza dover spegnere il servizio. Ultimato il backup può essere rimossa.

Nella caso in esame il device sottostante LVM non è una partizione fisica del disco ma il device DRBD. Dopo aver caricato il modulo del kernel dm-mod (Device mapper support) è necessario modificare il file di configurazione /etc/lvm/lvm.conf in modo che gli unici device sottoposti a scan alla partenza siano i device DRBD:

```
filter = [ "a/drbd.*/" , "r/.*/" ]
types = [ "drbd", 16 ]
```

Le stesse operazioni vanno ripetute su ambedue i nodi. Sul nodo primario si può creare il device LVM come segue:

```
#drdbadm primary mail
#vgscan
 Reading all physical volumes. This may take a while...
 No volume groups found
(Make any previously set up volume groups available)
#vgchange -a y
#pvcreate /dev/drbd0
#vgcreate mail /dev/drbd0
#lvcreate -L190G -nmail0 mail
#mkfs.ext3 /dev/mail/mail0
#mount /dev/mail/mail0 /mail0
```

La dimensione del Logical Volume è del tutto indicativa, perchè può essere cambiata in

seguito.

Il servizio LVM non va avviato con il runlevel di default, perchè del suo avvio se ne occuperà il software di gestione del cluster.

3.4 Impatto di DRBD sulle prestazioni

La scelta del filesystem da utilizzare sul device DRBD è legata al futuro uso del server. Nel caso in esame si tratta di un sistema per la posta elettronica che ospita un server IMAP e un server Postfix, quindi sicuramente è necessario ottimizzare al meglio le performance di input/output su disco.

Nel seguito si dimostrerà che la scelta di un filesystem moderno e performante può rendere il calo relativo delle prestazioni dovuto all'uso di DRBD+LVM del tutto trascurabile. Si tratterà in seguito di decidere in ogni situazione particolare se le prestazioni "assolute" soddisfano le proprie aspettative.

DRBD può essere calibrato in modo che usi al meglio la connessione di rete a disposizione. Le variazioni dei parametri di configurazione sono state realizzate forzando un full-resync del device e accertandosi che non ci fossero errori nei log di DRBD, dovuti ad un dimensionamento errato dei buffer di comunicazione rispetto alla velocità dei dischi e della rete, intesa come intero canale di comunicazione tra i kernel dei due nodi.

Per forzare la sincronizzazione totale dal nodo remoto al nodo sul quale si esegue il comando si usa:

#drbdadm invalidate <nome risorsa>

Questo tipo di test ha portato ad impostare i seguenti parametri nel file di configurazione di DRBD:

max-buffers 4096;
max-epoch-size 2048;
rate 100M;
nr_requests=128;

La scelta è specifica per l'hardware a disposizione nel test, e deve essere rivista caso per caso.

Un'altra modifica nella configurazione di base riguarda la MTU della rete interna. E' facile vedere, ad esempio con il software Iperf:

#USE=threads emerge iperf
hatest1#iperf -s --bind 192.168.0.1
hatest2#iperf --bind 192.168.0.2 -c 192.168.0.1

che modificare la MTU della interfaccia bond0 portandola da 1500 a 9000 ("Jumbo frames")

riduce drasticamente il tempo di CPU impiegato dai processi che usano la rete. Per inciso il test in modalità broadcast misura circa 430Mbit/sec, ossia 53Mbyte/sec: il collo di bottiglia è costituito dai dischi e non dalla rete. Lo stesso effetto di diminuzione della CPU utilizzata appare evidente, una volta installato DRBD, guardando la percentuale di CPU dei processi drbd0_worker, drbd0_asender e drbd0_receiver durante un intenso lavoro di I/O sul disco.

Per modificare la MTU di bond0 è necessario modificare il file /etc/conf.d/net aggiungendo alla riga relativa alla interfaccia bond0 l'istruzione mtu 9000.

3.4.1 Test delle prestazioni

Per avere innanzitutto una idea di quanto peggiorino le operazioni di I/O a causa di DRBD e LVM, e poi per confrontare il comportamento dei diversi filesystem disponibili in Linux, si sono utilizzati tre programmi di test:iozone, tiobench e bonnie++.

L'hardware sul quale sono stati eseguiti i test: controller RAID 3ware 9500, doppio Xeon a 2,5Ghz, 2GB RAM.

La stima sulla incidenza dei vari strati software è stata fatta usando il filesystem ext3 creato prima sulla partizione fisica, poi sul device DRBD, infine sul device LVM. I risultati del test bonnie++, usato per testare la creazione e cancellazione di un grande numero di file, eseguito con:

#bonnie++ -d /mail -s 0 -r 2048 -u 0 -n 4:5M:1k:200:4

sono stati i seguenti:

TAB. 1: Test bonnie++

	Ext3			***Ext3+DRBD***			***Ext3+DRBD+LVM***		
Sequential									
Create	12	16	3147	7	26	3603	7	26	2913
Read	30	9	149	28	13	208	27	15	150
Delete	144	5	202	138	5	145	145	5	204
Random									
Create	11	16	1725	8	25	3853	7	16	3523
Read	19	5	345	18	8	265	19	16	233
Delete	110	4	145	110	4	161	113	4	112

I tre numeri per ogni colonna indicano nell'ordine il lavoro svolto, quindi numeri maggiori sono indice di prestazioni migliori, la percentuale di cpu impiegata e la latenza delle operazioni in millisecondi. Si nota il calo delle prestazioni in scrittura sequenziale, come del resto ci si aspettava, e un aumento della latenza delle operazioni per la scrittura casuale. La lettura e la cancellazione dei file sembrano non subire variazioni di rilievo.

Il secondo test condotto è stato iozone:

#iozone -Rab /root/ext3/live/iozone.wks -g 4G -S 512 -i 0 -i 2

Eseguito con questa linea di comando il programma effettua i test di write/rewrite e random read/write, variando automaticamente la dimensione dei blocchi di scrittura da 4Byte a 16KByte, e la dimensione del file di test da 64K fino a 4 GByte. I risultati del test sono qui presentati mediati su tali dimensioni, e divisi per operazioni effettuate nella cache del filesystem o fuori cache, quindi con file di 2 GByte o maggiori. Sull'asse delle y sono riportati i valori in Kbyte/sec:

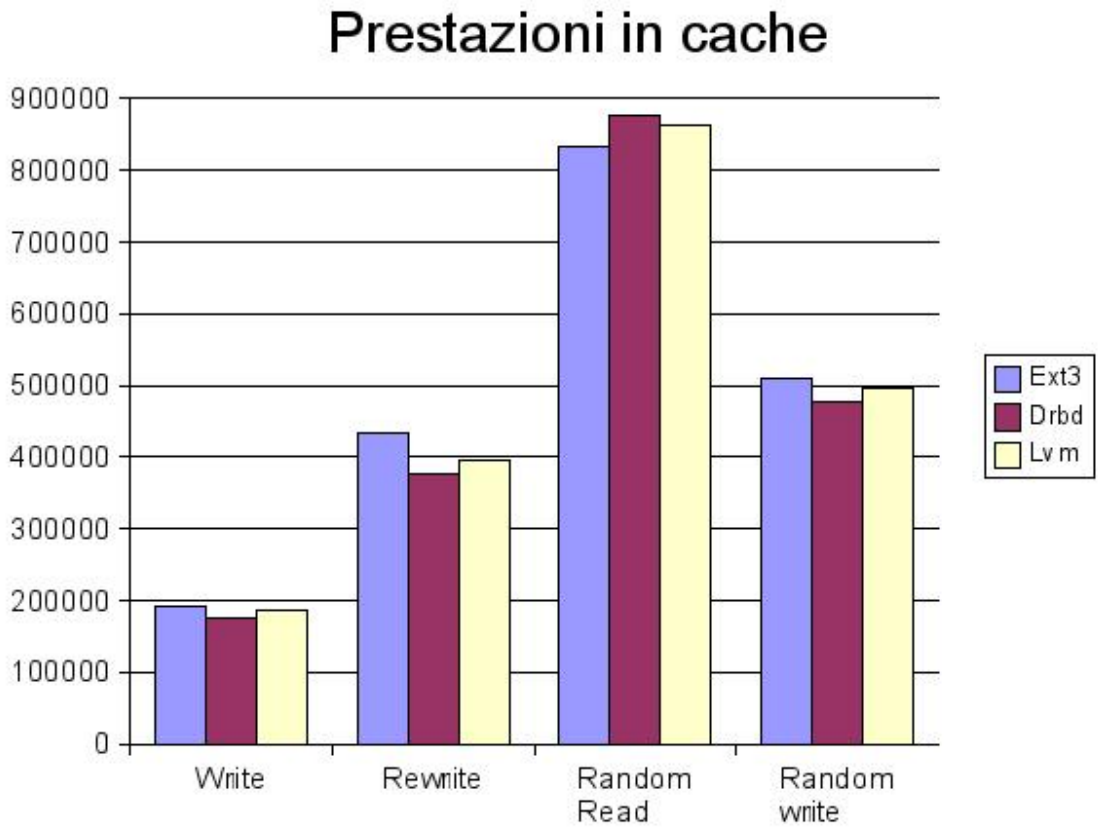

FIG. 2: Raffronto con l'intervento del caching del sistema operativo.

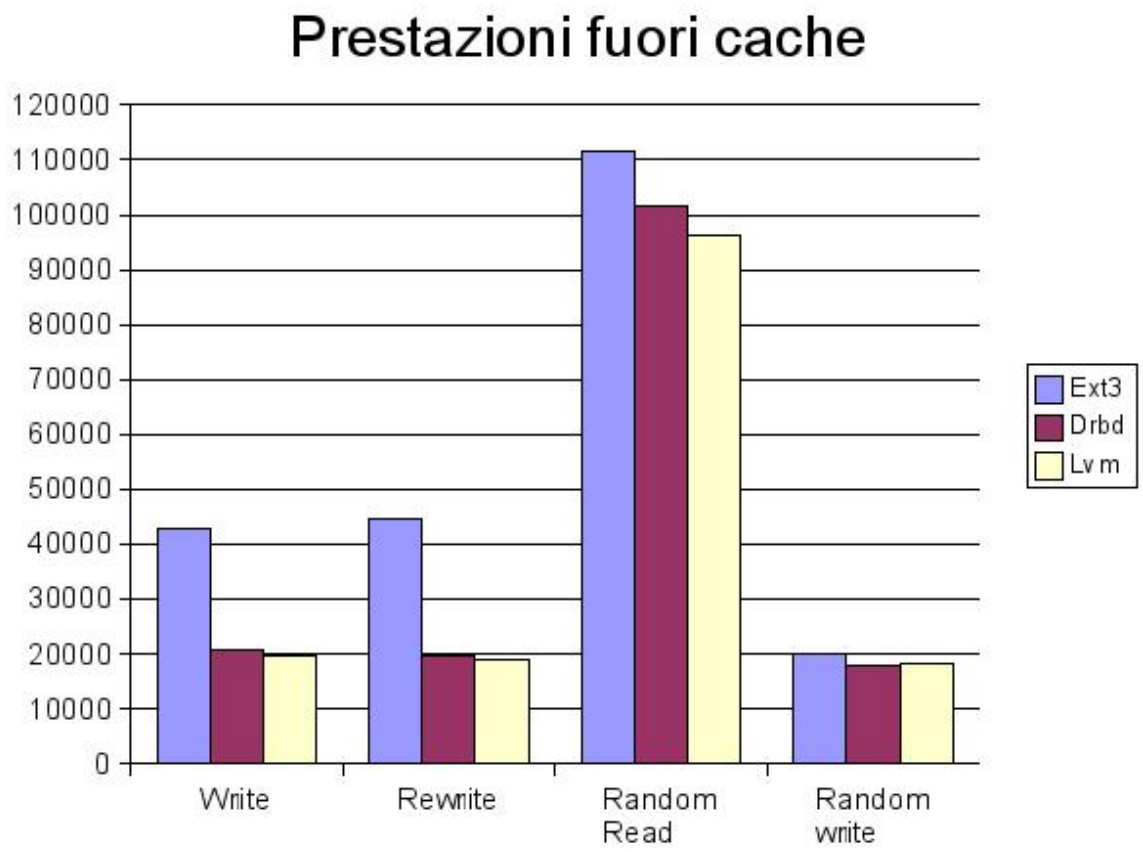

FIG. 3: Raffronto senza l'intervento del caching del sistema operativo.

In definitiva dai test, che naturalmente non vogliono essere esaustivi ma dare semplicemente una idea di quel che succede, sembra chiaro che fino a quando le operazioni di scrittura avvengono usando la cache del filesystem del kernel linux il calo di prestazioni in questa architettura è molto limitato, e praticamente non risente della introduzione di LVM. La creazione, sequenziale o casuale, di un gran numero di file di piccole dimensioni evidenzia un calo delle prestazioni intorno al 40%. Le operazioni su file di grosse dimensioni, che quindi non usano la cache del filesystem, mostrano il rallentamento del device sottostante, drammatico nel caso di scrittura sequenziale (superiore al 50%), ma molto inferiore (< 10%) nel caso di scrittura e lettura casuale.

I test successivi sono stati condotti con l'intenzione di confrontare tra loro i vari file system journaled da installare sul device LVM, in modo da avere una idea, seppure approssimativa, di quale di questi scegliere per avere le migliori prestazioni. Di seguito si riportano i risultati dei due test visti in precedenza:

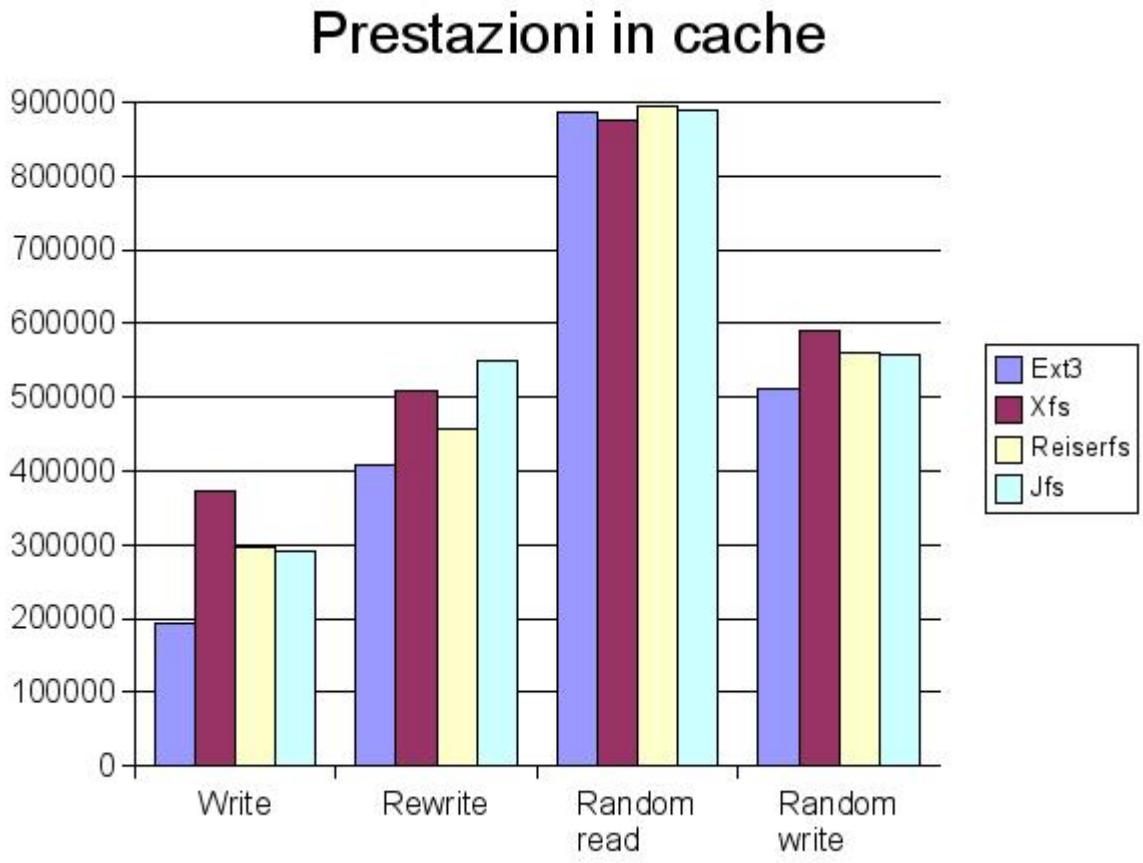

FIG. 4: Raffronto fra diversi file-system montati su LVM con l'intervento del caching del sistema operativo.

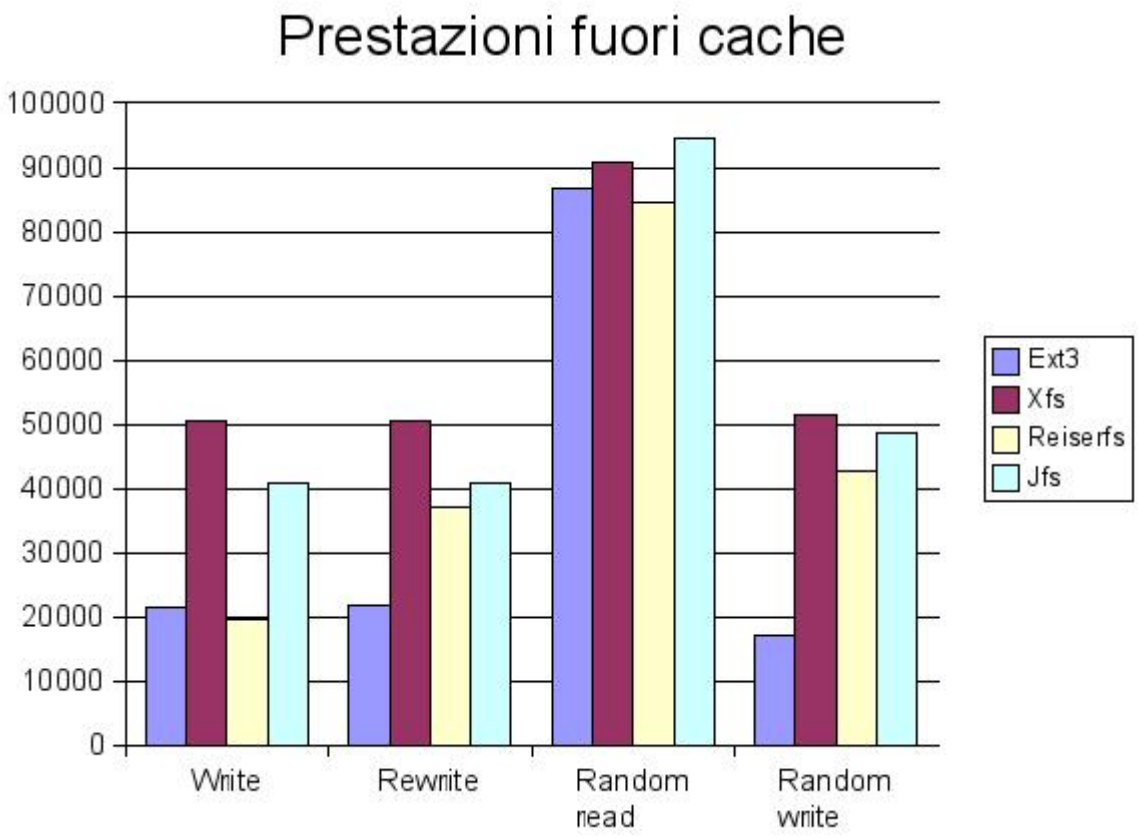

FIG. 5: Raffronto fra diversi file-system montati su LVM senza l'intervento del caching del sistema operativo.

TAB. 2: Test bonnie++

	Ext3			***XFS***			***Reiserfs***			***JFS***		
Sequential												
Create	7	16	3424	9	15	2796	9	24	19453	9	18	1000
Read	34	20	163	25	16	533	22	16	536	40	22	320
Delete	142	5	171	782	21	239	512	56	126	610	2	108
Random												
Create	7	16	4010	9	15	2521	9	23	18868	9	18	1000
Read	22	12	370	19	12	714	15	10	1206	21	11	1200
Delete	100	4	231	1141	33	90	128	14	1648	210	1	2200

Infine il test tiobench, con 10 thread contemporanei, ognuno con un file di 8 GByte, verifica la velocità di scrittura e lettura:

#tiotest -t 10 -f 8192 -r 4000 -b 4096 -d /mail0

EXT3

```
,-----------------------------------------------------------------.
| Item | Time | Rate | Usr CPU | Sys CPU |
+----------------------+----------+-------------+----------+---------+
| Write 81920 MBs | 7566.7 s | 10.826 MB/s | 0.3 % | 23.5 % |
| Random Write 156 MBs | 369.3 s | 0.423 MB/s | 0.0 % | 1.0 % |
| Read 81920 MBs | 1069.1 s | 76.622 MB/s | 1.7 % | 19.3 % |
| Random Read 156 MBs | 259.1 s | 0.603 MB/s | 0.0 % | 0.5 % |
`-----------------------------------------------------------------'
```

Tiotest latency results:

```
,-------------------------------------------------------------------------.
| Item | Average latency | Maximum latency | % >2 sec| % >10 sec |
+--------------+-----------------+-----------------+----------+-----------+
| Write | 3.505 ms | 153202.638 ms | 0.05518 | 0.00078 |
| Random Write | 20.716 ms | 4666.292 ms | 0.05000 | 0.00000 |
| Read | 0.497 ms | 1838.389 ms | 0.00000 | 0.00000 |
| Random Read | 63.474 ms | 1172.909 ms | 0.00000 | 0.00000 |
|--------------+-----------------+-----------------+----------+---------|
| Total | 2.078 ms | 153202.638 ms | 0.02759 | 0.00039 |
`--------------+-----------------+-----------------+----------+-----------'
```

real 155m5.484s
user 1m7.414s
sys 33m20.259s

XFS

```
,-----------------------------------------------------------------.
| Item | Time | Rate | Usr CPU | Sys CPU |
+----------------------+----------+-------------+----------+---------+
| Write 81920 MBs | 6127.3 s | 13.370 MB/s| 0.4 % | 20.0 % |
| Random Write 156 MBs | 331.8 s | 0.471 MB/s | 0.0 % | 1.2 % |
| Read 81920 MBs | 884.5 s | 92.615 MB/s| 2.1 % | 24.0 % |
| Random Read 156 MBs | 240.9 s | 0.649 MB/s | 0.0 % | 0.4 % |
`-----------------------------------------------------------------'
```

Tiotest latency results:

```
,-------------------------------------------------------------------------.
| Item | Average latency | Maximum latency | % >2 sec| % >10 sec |
+--------------+-----------------+-----------------+----------+-----------+
| Write | 2.867 ms | 86050.270 ms | 0.02256 | 0.01062 |
```

Random Write	0.030 ms	125.162 ms	0.00000	0.00000
Read	0.412 ms	1268.152 ms	0.00000	0.00000
Random Read	55.891 ms	277.224 ms	0.00000	0.00000
--------------+----------------+----------------+----------+-----------
Total	1.689 ms	86050.270 ms	0.01126	0.00530
`--------------+----------------+----------------+----------+-----------'

real 126m32.667s
user 1m8.176s
sys 24m0.341s

REISERFS

,--.
| Item | Time | Rate | Usr CPU | Sys CPU |
+----------------------+----------+-------------+----------+---------+
Write 81920 MBs	10319.6 s	7.938 MB/s	0.3 %	19.1 %
Random Write 156 MBs	354.2 s	0.441 MB/s	0.0 %	1.1 %
Read 81920 MBs	2777.7 s	29.492 MB/s	0.7 %	13.9 %
Random Read 156 MBs	256.9 s	0.608 MB/s	0.0 %	0.5 %
`--'
Tiotest latency results:

,---.
| Item | Average latency | Maximum latency | % >2 sec| % >10 sec |
+--------------+-----------------+-----------------+----------+-----------+
Write	4.721 ms	102080.479 ms	0.02223	0.01695
Random Write	20.865 ms	2824.790 ms	0.01750	0.00000
Read	1.269 ms	5749.239 ms	0.00003	0.00000
Random Read	63.134 ms	6092.853 ms	0.00250	0.00000
--------------+----------------+----------------+----------+-----------
Total	3.069 ms	102080.479 ms	0.01113	0.00846
`--------------+----------------+----------------+----------+-----------'

real 229m17.756s
user 1m2.490s
sys 39m47.797s

JFS

,--.

| Item | Time | Rate | Usr CPU | Sys CPU |
+----------------------+----------+--------------+----------+---------+
Write 81920 MBs	5451.2 s	15.028 MB/s	0.5 %	29.0 %
Random Write 156 MBs	298.3 s	0.524 MB/s	0.0 %	1.2 %
Read 81920 MBs	926.3 s	88.437 MB/s	1.9 %	20.7 %
Random Read 156 MBs	166.6 s	0.938 MB/s	0.1 %	0.5 %
`--'

Tiotest latency results:

,---.
| Item | Average latency | Maximum latency | % >2 sec | % >10 sec |
+--------------+-----------------+-----------------+----------+-----------+
Write	2.537 ms	10347.422 ms	0.05332	0.00001
Random Write	0.025 ms	99.104 ms	0.00000	0.00000
Read	0.433 ms	1482.398 ms	0.00000	0.00000
Random Read	41.116 ms	226.977 ms	0.00000	0.00000
--------------+-----------------+-----------------+----------+-----------
Total	1.522 ms	10347.422 ms	0.02661	0.00000
`--------------+-----------------+-----------------+----------+-----------'

real 114m11.116s
user 1m10.974s
sys 29m35.797

In definitiva nei test generici si vede è che:

1. Ext3 è il filesystem più lento nelle operazioni di scrittura di file di dimensioni tali da essere contenuti nella cache del filesystem, che nel caso in esame saranno presumibilmente la maggior parte. Il filesystem più veloce per questo tipo di file è XFS. JFS è il più veloce in lettura sequenziale e random di file fino a 5 MB.
2. ReiserFS è caratterizzato da una alta latenza, che ne penalizza fortemente le prestazioni anche nel campo in cui dovrebbe difendersi meglio, quello dei piccoli file.
3. Tranne che per la cancellazione dei file, XFS e JFS hanno prestazioni confrontabili sia per velocità che per latenza che per occupazione di CPU.
4. JFS si è rivelato sempre il più rapido nel concludere l'esecuzione dei test.

I numeri devono essere letti con l'avvertenza che ext3 è l'unico filesystem che fa journaling anche dei dati oltre che dei metadati.

I risultati mostrati non hanno ovviamente valore generale, si riferiscono alla particolare configurazione hardware + software di sistema utilizzata, nè possono essere presi come relativi a quelle che saranno le prestazioni del sistema ultimato, in quanto dipendono anche

dai test utilizzati.

Quanto questo sia vero lo si è verificato con un test sulla piattaforma ultimata, descritta in queste pagine. Con una mailbox di prova l'apertura via web-mail della INBOX con 4000 mail necessita, dal login alla visualizzazione della prima pagina di 25 sec con Ext3, 2 sec con XFS, 4 sec con JFS e 3 sec con Reiserfs.

In definitiva la scelta di un filesystem moderno e performante può rendere il calo delle prestazioni dovuto all'uso di DRBD+LVM con ext3 del tutto trascurabile. A seconda della applicazione usata il calo assoluto delle prestazioni con XFS o JFS potrà rivelarsi accettabile o meno.

4 GESTIONE AUTOMATICA DEL CLUSTER: HEARTBEAT

Come si è già preannunciato Heartbeat è il software di gestione del cluster, che si occupa di spostare i servizi dal nodo bloccato a quello funzionante.

Per verificare la funzionalità di Heartbeat si può inizialmente considerare il solo DRBD+LVM come "servizio" da erogare in alta disponibilità.

Il file/etc/ha.d/haresources, che deve essere identico sui due nodi, contiene la lista delle risorse che possono migrare tra le macchine. In questo file quindi non vanno inclusi gli indirizzi IP fissi riferiti alle singole macchine, ma solo gli indirizzi IP relativi ai servizi offerti dal cluster. Ogni linea del file costituisce un gruppo logico di risorse, ossia una lista di risorse che si muovono, insieme, da un nodo all'altro, nell'ordine indicato. Per ogni linea cioé vengono avviate da sinistra a destra, e vengono fermate da destra a sinistra. Si assume inoltre che non vi sia alcuna relazione tra i differenti gruppi di risorse.

Nella configurazione minimale di test in cui l'unica risorsa è quella del disco, il file delle risorse conterrà solo:

hatest1 drbddisk::mail lvm Filesystem::/dev/drbd0::/mail0::ext3

dove lvm è lo script di avvio del Logical Volume Manager riportato in appendice, dato che quello presente in Heartbeat è appropriato solo per LVM1.

La configurazione dei parametri di funzionamento di heartbeat si trova nel file /etc/ha.d/ha.cf, del quale la distribuzione fornisce in genere un esempio; in particolare per Gentoo:

#cp /usr/share/doc/heartbeat-1.2.3/ha.cf /etc/ha.d/

In Appendice è riportato il file completo relativo al cluster in esame. Si noti che il logging viene indirizzato sul log di sistema, in modo che se swatch è attivo possa esaminare anche i log di heartbeat, e avvisare ad esempio l'amministratore quando c'è una transizione delle risorse. Il file deve essere presente identico sui due nodi, e deve avere permessi 600.

Di seguito alcuni commenti sui parametri di configurazione.

4.1 Dead-time

E' importante osservare che il deadtime del drbd deve essere impostato in modo tale che DRBD si accorga PRIMA di heartbeat che un nodo è morto. Se se ne accorge dopo allora nel caso ci sia un crash del nodo primario lo script di montaggio dello spazio disco DRBD, invocato da Heartbeat, non riuscirà a montare primario il vecchio secondario. Si ricordi infatti che DRBD viene fatto partire all'avvio della macchina, e non da Heartbeat, che invece si occupa solo di montare correttamente il primario. Ecco quel che succede:

- Muore improvvisamente il nodo primario
- Dopo un certo intervallo di tempo heartbeat si accorge che il nodo è morto e prova a migrare il servizio
- Lo script di Heartbeat prova a montare sul nodo secondario lo spazio disco, ma per DRBD il primario è ancora vivo e si rifiuta di farlo
- Di conseguenza i servizi non partono sul secondario e il cluster è stato inutile!!

I tre parametri di DRBD da controllare sono: timeout, connect-int e ping-int. Se la connessione tra le due macchine è idle per più di ping-int secondi DRBD genera un pacchetto di check per verificare che l'altro host sia vivo. Se non è possibile avere la connessione immediatamente, DRBD continua a provare la connessione, con un intervallo tra due tentativi pari a connect-int secondi. Infine se il nodo partner non risponde ai tentativi di connessione entro timeout decimi di secondo, viene considerato morto e la connessione viene abbandonata.

Le impostazioni correnti del setup descritto sono:

timeout 80; # unit: 0.1 seconds
connect-int 10; # unit: seconds
ping-int 10; # unit: seconds

Con questi numeri nel caso peggiore DRBD ha bisogno di 18 secondi per verificare la morte del suo partner.

Il parametro di Heartbeat da controllare è il deadtime. Per quanto detto prima il valore di 14 secondi che viene spesso impostato di default NON è sufficiente per scongiurare il pericolo di cui sopra. E' necessario avere un valore abbastanza più grande di 18 secondi, nel caso in esame si è scelto un

deadtime 40

Il prezzo da pagare è una minore velocità di reazione del cluster a crash hardware.

4.2 Watchdog

Il watchdog usato è di tipo software, anche se evidentemente una maggiore affidabilità

si ha usando un device hardware. Per attivarlo è necessario includere nella compilazione del kernel il supporto a watchdog e compilare il modulo softdog, con l'avvertenza di non abilitare tra le opzioni disponibili CONFIG_WATCHDOG_NOWAYOUT, e di creare il device opportuno con:

#mknod /dev/watchdog c 10 130

Per caricare all'avvio il modulo softdog in Gentoo si inserisce il nome del modulo nel file /etc/modules.autoload.d/kernel-2.6.

4.3 Ipfail

Il plugin ipfail si occupa di verificare l'effettiva raggiungibilità dei server dalla LAN, e necessita per poter operare di un utente cluster che deve essere presente (heartbeat gira invece con i privilegi di nobody). Per controllare la presenza di connettività verso l'esterno viene usato un ping ad un host che si suppone sempre presente, ad esempio il gateway della sottorete alla quale appartengono i due nodi. Se uno dei due nodi non riesce a raggiungere l'host esterno per un certo periodo di tempo, heartbeat assume che la connessione del nodo verso la LAN sia persa, e sposta i servizi sull'altro nodo, sempre che quest'ultimo sia correttamente connesso. Se ipfail segnala che anche il secondo nodo non ha connessione i servizi non vengono spostati, il che garantisce che una eventuale avaria dell'host di controllo non scateni migrazioni ingiustificate e caotiche. E' anche possibile usare più nodi esterni per il ping, per aumentare l'efficacia del controllo.

4.4 Autofailback

Infine si noti che l'opzione autofailback è disabilitata: in questo caso quando un nodo torna all'interno del cluster non si riappropria automaticamente del proprio gruppo di risorse: sarà compito dell'amministratore valutare caso per caso se e quando ripristinare la piena funzionalità.

4.5 Serial line

Con heartbeat 1.2.3 su un cluster con kernel 2.4.29 la linea seriale a 19200 baud è sufficiente a garantire la connettività. Su un secondo cluster con stessa versione di heartbeat ma kernel 2.6.12.6 la connessione seriale apparentemente funziona, ma dopo un certo intervallo di tempo su uno dei nodi compaiono dei messaggi del tipo:

heartbeat[26704]: WARN: TTY write timeout on [/dev/ttyS0] (no connection or bad cable? [see documentation])

e anche un semplice cat /dev/ttyS0 mostra l'assenza di connessione. Il problema si risolve impostando la velocità di 115200 baud sulla porta seriale, inserendo quindi la riga

stty 115200 < /dev/ttyS0

in local.start e modificando naturalmente il parametro corrispondente in ha.cf

4.6 Comunicazione criptata

L'ultimo file di configurazione di heartbeat è /etc/ha.d/authkeys, che consente di impostare gli algoritmi crittografici per la comunicazione tra i nodi, necessari a far si che sia possibile una comunicazione sicura anche attraverso la rete pubblica.

Ad esempio per impostare l'uso dell'algoritmo "sha1":

auth 1
1 sha1 passwdsha

4.7 Test del cluster

A questo punto si è ottenuto lo scheletro, funzionante, del cluster in alta disponibilità, che offre come servizio lo spazio disco DRBD. E' possibile quindi provare le funzionalità di migrazione automatica, di test della connessione verso l'esterno (ipfail), di alta disponibilità della linea interna in channel bonding.

Per eseguire delle prove si possono anche spostare i gruppi di risorse manualmente. I comandi si devono leggere pensando alla azione come riferita al nodo sul quale si stà eseguendo il comando, e alla risorsa come locale al nodo se in haresources appartiene allo stesso nodo. Ad esempio:

hatest1#/usr/lib/heartbeat/hb_standby local

ferma (standby) sul nodo hatest1 le risorse di proprietà del nodo stesso (local). Invece

hatest1#/usr/lib/heartbeat/hb_takeover foreign

prende (takeover) sul nodo hatest1 le risorse che a lui non appartengono (foreign).

Ambedue i comandi possono essere invocati con gli argomenti local, foreign o all.

5 INSTALLAZIONE E MONITOR DEL SERVIZIO

Il servizio che il cluster deve erogare può essere installato a questo punto sul device DRBD+LVM. Non esiste una procedura standard per l'installazione di un servizio in alta disponibilità. In genere però i passi da seguire sono i seguenti:

1. Installare gli eseguibili in ambedue i nodi, nelle directory previste dalla particolare distribuzione utilizzata. E' anche possibile installare nella partizione condivisa, ma in tal caso non si può seguire la procedura standard, nè usare eventuali meccanismi di aggiornamento automatico.

2. Creare dal nodo primario una directory nel device DRBD+LVM che contenga i file di configurazione del servizio. Nel caso in esame si tratta di un server Postfix, e quindi si è copiata /etc/postfix nella directory condivisa /mail0/etc/postfix.
3. Eliminare la directory di configurazione originale e sostituirla con un softlink alla directory condivisa.
4. Modificare i file di configurazione in modo che utilizzino la directory condivisa per i dati di funzionamento del server. Per postfix ad esempio è necessario indicare le alias_maps presenti in /mail0/etc/postfix.

Se tutto funziona si può aggiungere lo script di partenza del servizio al file /etc/ha.d/haresources che definisce le risorse gestite da Heartbeat.

Il servizio può però fermarsi anche per motivi diversi dal guasto hardware. Ad esempio un aggiornamento non riuscito o un errore dell'applicativo che in una particolare situazione blocchi il server. In casi di questo genere Heartbeat non rivela la situazione anomala, ed è per questo che viene utilizzato MON. Si tratta di un software che, disinteressandosi dello stato dell'hardware sottostante, interroga direttamente il servizio erogato alla ricerca di una risposta standard. Se tale risposta non arriva può essere configurato per avvisare l'amministratore e migrare sul nodo rimanente il servizio.

Tipicamente MON viene eseguito localmente su ogni nodo, e controlla i servizi che su tale nodo sono presenti. Se i servizi infatti vengono spostati c'è la possibilità che monitorarli sull'altro nodo sia ininfluente, perchè una successiva migrazione "indietro" porterebbe ad un nuovo blocco. Non si tratta però di una regola fissa, si può anche scegliere di far migrare MON insieme ai servizi, in modo da gestire ad esempio casi di blocco derivanti da situazioni transitorie e relative al particolare nodo.

Ad esempio nella pratica questo sistema è stato oggetto di un attacco DDoS di breve durata, che ha bloccato il servizio di posta. MON ha rilevato il blocco e spostato sul secondo nodo il servizio, dopo aver provato inutilmente a riavviarlo. La migrazione di MON sul secondo nodo in questo caso era corretta, perchè ultimato il DDoS era nuovamente necessario monitorare il servizio.

I file di configurazione di MON vengono installati, con Heartbeat, nella directory /etc/mon.d/. Nel file mon.cf vengono definite le politiche di monitoring, e gli script da richiamare per controllare il server. Ad esempio:

```
#
# mon.cf,v 1.1 2001/01/20 01:05:43 achim Exp
#
alertdir = /usr/lib/mon/alert.d
mondir = /mail0/mon/mon.d
logdir = /var/log
historicfile = /var/log/mon
statedir = /mail0/mon/state
```

```
authfile = /mail0/mon/auth.cf
maxprocs = 20
histlength = 100
randstart = 60s

#
# define groups of hosts to monitor
#
hostgroup smtp postfix.ba.infn.it

watch smtp
  service smtp_check
        interval 15s
        monitor smtp3.monitor -t 70 -T 30 -l /mail0/mon/smtp_200507.log
        period wd {Sun-Sat} hr {0-23}
                alertafter 2
                alert mail.alert -S "Attenzione! postfix non sta rispondendo" administrator@poweruser.ba.infn.it
                alert postfixError.alert
  service smtp_takeover
        interval 45s
        monitor smtp3.monitor -t 70 -T 30 -l /mail0/mon/smtp_200507.log
        period wd {Sun-Sat} hr {0-23}
                alertafter 3 30m
                numalerts 1
                alert mail.alert -S "Attenzione! postfix non risponde ancora: migrazione!" administrator@poweruser.ba.infn.it
                alert postfixTakeover.alert
```

E' chiaro che il server in questione svolge la funzione di mail relay, lo script non può avvisare l'amministratore, a meno che mail.alert non mandi il mail direttamente alla macchina dell'amministratore.

Si noti che la directory che contiene gli script di alert è locale ai nodi, in modo da far eseguire il comando corretto di takeover o standby a seconda del nodo sul quale in quel momento risiede il servizio.

La riga "watch" identifica un gruppo di test, in questo caso formato dai due "service" smtp_check e smtp_takeover. I due service vengono eseguiti continuamente. Il primo dopo due fallimenti rilevati dallo script smtp3.monitor manda un mail all'amministratore ed esegue lo script postfixError.alert, (che può a sua volta fare altre azioni, ad esempio far ripartire il servizio). Il secondo service invece entra in azione dopo che sono stati generati 3 errori nel

giro di 30 minuti, quindi sicuramente il tentativo di riavviare il server localmente non ha sortito il risultato voluto. In tal caso viene effettuata la migrazione del servizio.

Infine il file auth.cf nella directory /etc/mon.d stabilisce per ogni funzione di mon quali utenti possono eseguirla, dato che alcune sono critiche per il sistema.

6 STONITH

La presenza per la comunicazione tra i nodi di un canale ethernet in alta disponibilità, uno ethernet esterno e uno seriale sembra sufficiente a rendere estremamente improbabile una completa perdita di comunicazione tra i nodi.

Questa eventualità è la più pericolosa per un cluster di questo tipo. Il disastro ha queste fasi:

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'avere due IP uguali sulla rete, se i due nodi ad un certo punto si vedono nuovamente i dati vengono corrotti.

Per scongiurare una evenienza di questo tipo si moltiplicano i canali di comunicazione tra i server. Sul cluster di posta in produzione a Bari ci sono 6 canali di comunicazione inter-nodo.

La catena di eventi che potrebbe portare a questo disastro però potrebbe essere molto varia... Ad esempio potrebbe anche succedere che il nodo primario per qualche (oscura) ragione si blocchi completamente, dando il tempo al secondario di diventare primario, e poi si sblocchi riprendendo (a tradimento) il suo ruolo di primario....

Per scongiurare questa eventualità alla radice viene usato un device STONITH. Si tratta di un interruttore comandato dal PC che permette di spegnere il nodo antagonista: il primo che riesce a spegnere l'altro nodo vince il duello e rimane l'unico primario.

Per sapere quali device STONITH vengono supportati da heartbeat:

#stonith -h

Nella versione 1.2.3 di heartbeat vengono supportati i device wti_nps e wti_tps. Nel cluster in esame si è usato il device WTI-IPS-400: è un interruttore comandato via rete che accetta una sola connessione telnet contemporanea, e di fatto, a meno di qualche messaggio di warning non influente, ha dimostrato di essere compatibile con il plugin wti_nps.

Nelle impostazioni generali del device bisogna aver cura di impostare:

Command echo ON

Command confirmation ON
Automated Mode OFF

Nelle impostazioni dei Plug Parameters bisogna dare una password diversa per ogni presa, e chiamare ogni presa con il nome dell'host collegato ad essa, come compare nel file di configurazione di Heartbeat. Nell'esempio le due prese si chiameranno hatest1 e hatest2.

A questo punto è possibile impostare l'uso dello stonith nel file /etc/ha.d/ha.cf

stonith wti_nps /etc/ha.d/conf/wti.conf

Nel file di configurazione deve essere presente l'indirizzo IP del device, e la password relativa alla presa dell'altro host. Dunque sull'host hatest1 in /etc/ha.d/conf/wti.conf deve esserci:

ip.device.di.stonith passwdpresahatest2

e viceversa sul nodo hatest2. Attenzione a non lasciare spazi vuoti dopo la password!

Per verificare che la configurazione sia corretta è possibile eseguire:

stonith -l -t wti_nps -F /etc/ha.d/conf/wti.conf

Nell'output deve comparire la presa che comanda l'altro nodo.

Una volta installato il device stonith la catena di eventi descritta sopra non può accadere, perchè ogni nodo quando perde completamente la comunicazione con l'altro nodo lo spegne! Uno solo dei due rimane acceso e prende il ruolo primario, il secondo nodo quando ripartirà troverà il primo nodo funzionante e si unirà al cluster.

APPENDICE

A. /etc/drbd.conf

```
resource mail {
  protocol C ;
   incon-degr-cmd "echo '!DRBD! primario su dati inconsistenti' | wall ; sleep 60 ; halt
-f";
  startup {
          degr-wfc-timeout 120;    # 2 minutes.
  }

  disk {
    on-io-error panic;
  }
```

```
  net {
    timeout     80;    # unit: 0.1 seconds
    connect-int  10;    # unit: seconds
    ping-int     10;    # unit: seconds
    #ko-count     4;     # if some block send times out this many times,
                        # the peer is considered dead, even if it still
                        # answeres ping requests
    max-buffers 4096;
    max-epoch-size 2048;
  }

  syncer {
    rate 100M;
    group 0;
    al-extents 521;
  }

  on hatest1 {
    device /dev/drbd0;
    disk   /dev/sda1;
    address 192.168.0.1:7788;
    meta-disk /dev/sdb13[0];
}

  on hatest2 {
    device   /dev/drbd0;
    disk     /dev/sda1;
    address  192.168.0.2:7788;
    meta-disk /dev/sdb13[0];
  }
}
```

B. /etc/init.d/ifenslave

```
#!/sbin/runscript
#Copyright 1999-2003 Gentoo Technologies, Inc.
#Distributed under the terms of the GNU General Public License, v2 or later
depend() {
        need net
        before logger
        before drbd
        before heartbeat
}

start() {
        ebegin "Enslaving ${ifenslave} to ${bond}"
        /sbin/ifenslave ${bond} ${ifenslave}
```

```
        eend $?
}

stop() {
        ebegin "Freeing ${ifenslave} from ${bond}"
        /sbin/ifconfig ${bond} down
        eend $?
}
```

C. /etc/ha.d/haresources

hatest1 drbddisk::mail lvm Filesystem::/dev/mail0::/mail0::ext3 IPaddr::192.168.185.4/24/eth0 postfix

D. /etc/ha.d/ha.cf

```
logfacility     daemon
keepalive 1
deadtime 40
warntime 7
initdead 120
udpport 694
baud    115200
serial  /dev/ttyS0      # Linux
ucast bond0 192.168.0.2
ucast eth0 192.168.185.1
auto_failback off
watchdog /dev/watchdog
stonith wti_nps /etc/ha.d/conf/wti.conf
node    hatest1
node    hatest2
ping 192.168.185.254
respawn cluster /usr/lib/heartbeat/ipfail
apiauth ipfail gid=cluster uid=cluster
deadping 30
```