MMUL
http://www.mmul.it

I seminari di Mia Mamma Usa Linux

Evoluzione dell'alta affidabilità su Linux

MMUL
http://www.mmul.it

Chi siamo

Qualcosa di più semplice...

Che cos'è MMUL?

- MMUL nasce con il portale tecnico ***http://www.miamammausalinux.org*** nel gennaio 2008.
- Nell'ottobre 2010 prende vita la società con l'obiettivo di fornire il meglio ai clienti in materia Linux e software OpenSource.

MMUL
http://www.mmul.it

Chi siete

Breve introduzione dei partecipanti

MMUL
http://www.mmul.it

Prima parte

Cluster

MMUL
http://www.mmul.it

Cluster

- Cos'è un Cluster?
- Un cluster può essere definito come un gruppo di macchine che lavorano insieme per raggiungere uno scopo.
- Tale scopo può essere l'erogazione di un servizio, l'elaborazione di un calcolo, il bilanciamento di un servizio...
- Il numero minimo di macchine presenti in un cluster è quindi due.

MMUL
http://www.mmul.it

Tipologie di cluster

- Le principali tipologie di cluster sono due:
- **High Performance Cluster**
  - Cluster per il calcolo parallelo (un processo viene elaborato da più macchine invece che da una sola)
- **High Availability Cluster**
  - Cluster per l'erogazione di servizi altamente disponibili.
  - Cluster per il bilanciamento di servizi (load balancing mediante LVS – Linux Virtual Server).

MMUL
http://www.mmul.it

Differenze tra le tipologie di cluster

- Principali differenze tra le due tipologie di cluster:
  - **High performance**: priorità alle performance. Si lavora per rendere l'elaborazione il più veloce possibile
  - **High available (HA)**: priorità ai servizi. Si lavora per garantire continuità a quanto erogato. Un disservizio deve essere nei limiti del possibile trasparente all'utilizzatore finale.
- Questo seminario tratta di cluster HA.

MMUL
http://www.mmul.it

Categorie di cluster HA

- I cluster HA si dividono in due macro categorie:
  - **shared-everything**: cluster che per operare si basano su un'area comune (Storage, SAN, etc.). Il motore del cluster si occupa di regolare gli accessi concorrenti ai dati nelle zone condivise;
  - **shared-nothing**: cluster in cui ciascun nodo è totalmente indipendente dagli altri (DRBD, replica MySQL, CEPH storage). Il motore del cluster si occupa di mantenere aggiornati i dati uniformemente e limitare i danni provocati da situazioni di “Split brain”;

MMUL
http://www.mmul.it

Cluster in Linux: prodotti disponibili

- Diverse soluzioni cluster sono disponibili per Linux sul mercato:
  - **Veritas Cluster**: licenza chiusa, prodotto e mantenuto da Symantec (http://www.symantec.com);
  - **Oracle RAC**: licenza chiusa, prodotto e mantenuto da Oracle (http://www.oracle.com);
  - **Red Hat Cluster**: licenza GPL, mantenuto da Red Hat (http://www.redhat.com);
  - **Linux-HA**: licenza GPL/LGPL, prodotto e mantenuto dalla comunity Linux-ha dal 1999 (http://www.linux-ha.org);

MMUL
http://www.mmul.it

Seconda parte

Heartbeat/Corosync

Esempio di un file ha.cf:

```
##      keepalive: how long between heartbeats?
keepalive 1
##      deadtime: how long-to-declare-host-dead?
deadtime 10
##      warntime: how long before issuing "late heartbeat" warning?
warntime 5
##      Very first dead time (initdead)
initdead 20
##      What interfaces to unicast heartbeats over?
ucast eth1 10.0.0.1
ucast eth1 10.0.0.2
ucast eth0 192.168.1.86
ucast eth0 192.168.1.66
auto_failback off
node    debian-squeeze-nodo1
node    debian-squeeze-nodo2
ping 192.168.1.50
respawn hacluster /usr/lib/heartbeat/ipfail
apiauth ipfail gid=haclient uid=hacluster
```

Esempio di file haresources:

```
debian-squeeze-nodo1 IPaddr::193.42.170.84/24/eth0 apache2
```

Esempio di file cib.xml:

```
# cat /var/lib/heartbeat/crm/cib.xml
<cib validate-with="pacemaker-1.0" crm_feature_set="3.0.1" have-quorum="1" admin_epoch="0"
epoch="13" num_updates="0" cib-last-written="Tue Mar 29 22:13:05 2011" dc-uuid="crash-1">
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <nvpair id="cib-bootstrap-options-dc-version" name="dc-version" value="1.0.9-
da7075976b5ff0bee71074385f8fd02f296ec8a3"/>
        <nvpair id="cib-bootstrap-options-cluster-infrastructure" name="cluster-infrastructure"
value="openais"/>
        <nvpair id="cib-bootstrap-options-expected-quorum-votes" name="expected-quorum-votes"
value="2"/>
      </cluster_property_set>
    </crm_config>
    <nodes>
      <node id="crash-2" uname="crash-2" type="normal"/>
      <node id="crash-1" uname="crash-1" type="normal"/>
    </nodes>
    <resources/>
    <constraints/>
  </configuration>
```

Cluster Glue

La Cluster Glue è composta dalle seguenti componenti:

Local Resource Manager (LRM): questa componente rappresenta l'interfaccia tra Pacemaker ed i Resource Agents. Essa semplicemente processa i comandi ricevuti da Pacemaker, li passa ai Resource Agents e riporta il risultato ottenuto (success o failure).
Le operazioni gestite da LRM sono: start, stop, monitor, report di stato e lista delle risorse controllate.

STONITH: rappresenta un meccanismo di controllo dei nodi. Nel caso in cui un nodo è considerato "morto" esso viene forzatamente rimosso da STONITH ("Shoot The Other Node In The Head") in modo che vengano annullati tutti i rischi di scritture concorrenti anomale sui dati confivisi.

hb_report: uno strumento avanzato di report degli errori.

Cluster Plumbing Library: una libreria basso livello per le comunicazioni interne del cluster.

Resource Agents (prima parte)

I Resource Agents rappresentano gli strumenti attraverso i quali il cluster controlla le proprie risorse. Essi vengono definiti mediante:

Tipologia: LSB, OCF, Legacy (vedere slide successiva);
Fornitore: chi fornisce lo strumento (ad esempio heartbeat, pacemaker, linbit o "custom")
Nome: il nome effettivo del Resource Agent;

Una tipica definizione di risorsa in **Pacemaker**, come di vedrà avanti, assumerà questa forma:

```
primitive <Nome risorsa> <Tipologia>:<Fornitore>:<Nome> params <parametri> op <operations>
```

Resource Agents (seconda parte)

Pacemaker supporta tre tipi di Resource Agents:

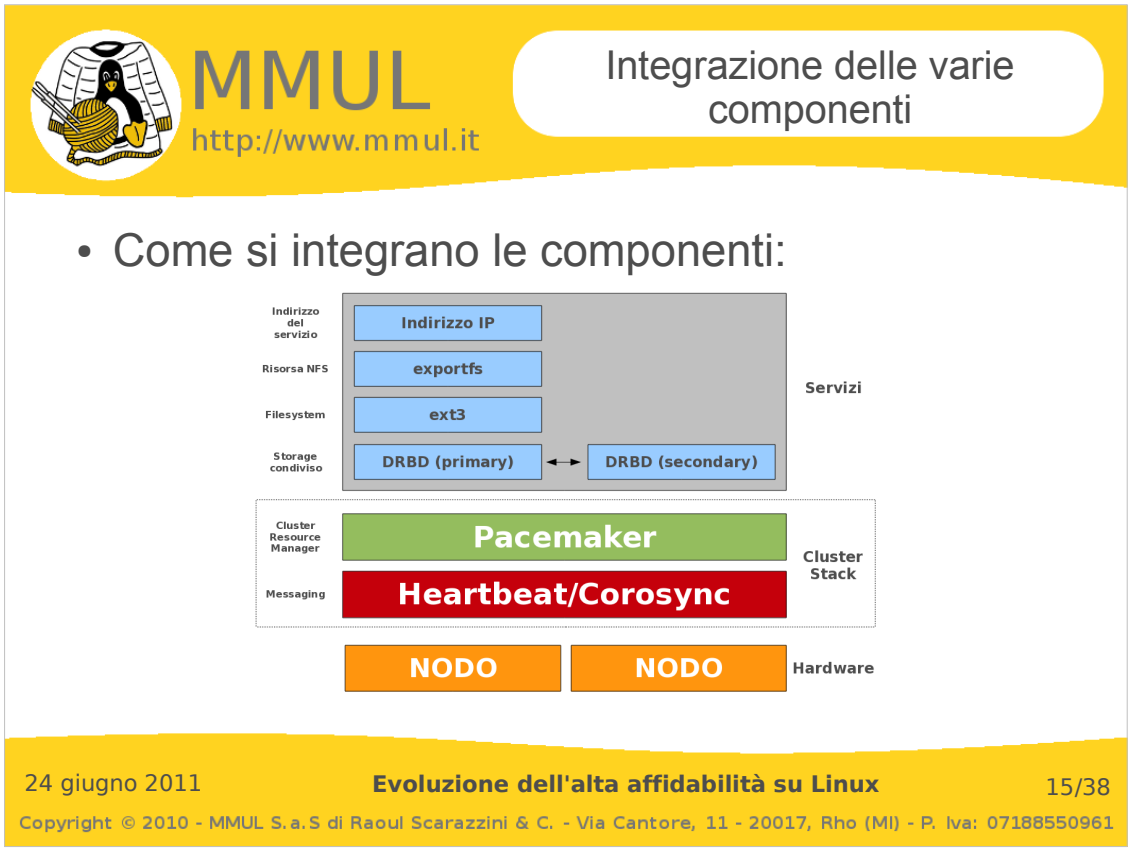
LSB Resource Agents: script che rispettano la Linux Standard Base, in sostanza gli script che avviano i demoni di sistema.
Tipicamente il posizionamento di tali script è all'interno del path **/etc/init.d/**.

OCF Resource Agents: script che rispettano l'Open Cluster Framework, la cui differenza principale rispetto agli script LSB risiede nel fatto che supportano più funzioni per il controllo delle risorse, in particolare in merito al monitoring.
Tipicamente il posizionamento di tali script è all'interno del path **/usr/lib/ocf/resource.d/** al di sotto della cartella esistono tante cartelle quanti sono i fornitori.

Legacy Heartbeat Resource Agents: script creati ed utilizzati nelle prime versioni di Heartbeat (che utlizzavano il file haresources), generalmente per ciascun legacy script esiste uno script LSB od OCF più aggiornato.
Tipicamente il posizionamento di tali script è all'interno del path **/etc/ha.d/resource.d/**.

Elenco dei Resource Agents disponibili in una tipica installazione di Heartbeat/Corosync e Pacemaker:

anything	eDir88	iscsi	nfsserver	Route
AoEtarget	Evmsd	iSCSILogicalUnit	nginx	rsyncd
apache	EvmsSCC	iSCSITarget	oracle	SAPDatabase
AudibleAlarm	exportfs	jboss	oralsnr	SAPInstance
ClusterMon	Filesystem	LinuxSCSI	pgsql	scsi2reservation
conntrackd	fio	LVM	pingd	SendArp
CTDB	ICP	MailTo	portblock	ServeRAID
db2	ids	ManageRAID	postfix	sfex
Delay	IPaddr	ManageVE	proftpd	SphinxSearchDaemon
drbd	IPaddr2	mysql	Pure-FTPd	Squid



Installazione di Heartbeat o Corosync (si veda la slide successiva per le differenze):

Debian/Ubuntu

```
# apt-get install pacemaker corosync
```

o

```
# apt-get install pacemaker heartbeat
```

OpenSuSe

```
# zypper ar http://clusterlabs.org/rpm/opensuse-11.3/clusterlabs.repo
# zypper refresh
# zypper install --from clusterlabs pacemaker libpacemaker3 corosync heartbeat cluster-glue
libglue2
```

RedHat e derivate

```
# su -c 'rpm -Uvh http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm'
# wget -O /etc/yum.repos.d/pacemaker.repo http://clusterlabs.org/rpm/epel-5/clusterlabs.repo
# yum install -y pacemaker corosync heartbeat
```

/etc/corosync/corosync.conf:

```
compatibility: whitetank

totem {
        version: 2
        secauth: off
        threads: 0
        interface {
                ringnumber: 0
                bindnetaddr: 192.168.1.0
                mcastaddr: 226.94.1.1
                mcastport: 5405
        }
}

logging {
        fileline: off
        to_stderr: yes
        to_logfile: yes
        to_syslog: yes
        logfile: /var/log/corosync/corosync.log
        debug: off
        timestamp: on
        logger_subsys {
                subsys: AMF
                debug: off
        }
}

amf {
        mode: disabled
}

service {
        # Load the Pacemaker Cluster Resource Manager
        ver:       0
        name:      pacemaker
}

aisexec {
        user:   root
        group:  root
}
```

/etc/ha.d/ha.cf:

```
autojoin none
keepalive 1
deadtime 10
warntime 5
initdead 20
mcast eth0 239.0.0.43 694 1 0
bcast eth1
node    debian-lenny-nodo1
node    debian-lenny-nodo2
crm respawn

/etc/ha.d/authkeys:

auth 1
```

/etc/ha.d/authkeys:

```
auth 1
1 crc
```

MMUL
http://www.mmul.it

Terza parte

DRBD

Pacchetti da installare:

Nelle distribuzioni recenti (quelle che montano un kernel di versione uguale o superiore alla 2.6.32) qundi non è necessaria l'installazione di alcun pacchetto driver, ma solo delle utility di gestione dei device drbd.
Ad esempio, per Debian/Ubuntu:

```
# apt-get install drbd8-utils
```

Per OpenSuSe:

```
# zypper install drbd
```

RedHat e derivate:

```
# yum install drdb83
```

Se la distribuzione non è recente allora si dovranno installare i pacchetti precompilati dei moduli, disponibili generalmente per tutte le distribuzioni. In alternativa, i driver sono compilabili da sorgenti.

Verificare se drbd è installato:

```
# modprobe drbd
# cat /proc/drbd
version: 8.3.7 (api:88/proto:86-91)
srcversion: EE47D8BF18AC166BE219757
```

L'output segnala che non ci sono stati problemi al caricamento del modulo drbd e che è stato creato nel filesystem /proc il file di status dei device (inizialmente ed ovviamente vuoto).

/etc/drbd.conf (inclusione di tutti gli altri file):

```
include "drbd.d/global_common.conf";
include "drbd.d/*.res";
```

/etc/drbd.d/global_common.conf (valori di default assunti da tutti i device creati, a meno di effettuare override):

```
common {
        protocol C;

        handlers {
                pri-on-incon-degr "/usr/lib/drbd/notify-pri-on-incon-degr.sh; /usr/lib/drbd/notify-
emergency-reboot.sh; echo b > /proc/sysrq-trigger ; reboot -f";
                pri-lost-after-sb "/usr/lib/drbd/notify-pri-lost-after-sb.sh; /usr/lib/drbd/notify-
emergency-reboot.sh; echo b > /proc/sysrq-trigger ; reboot -f";
                local-io-error "/usr/lib/drbd/notify-io-error.sh; /usr/lib/drbd/notify-emergency-
shutdown.sh; echo o > /proc/sysrq-trigger ; halt -f";
                }

        startup {
                # wfc-timeout degr-wfc-timeout outdated-wfc-timeout wait-after-sb
        }

        disk {
                # on-io-error fencing use-bmbv no-disk-barrier no-disk-flushes
                # no-disk-drain no-md-flushes max-bio-bvecs
        }

        net {
                # sndbuf-size rcvbuf-size timeout connect-int ping-int ping-timeout max-buffers
                # max-epoch-size ko-count allow-two-primaries cram-hmac-alg shared-secret
                # after-sb-0pri after-sb-1pri after-sb-2pri data-integrity-alg no-tcp-cork
        }

        syncer {
                # rate after al-extents use-rle cpu-mask verify-alg csums-alg
                Rate 600M;
        }
}
```

/etc/drbd.d/r0_share-a.res (definizione specifica di una risorsa)**:**

```
resource r0 {
        device /dev/drbd0;
        meta-disk internal;
        disk /dev/vg_store/lv_share-a;

        handlers {
                split-brain "/usr/lib/drbd/notify-split-brain.sh root";
                out-of-sync "/usr/lib/drbd/notify-out-of-sync.sh root";
                fence-peer "/usr/lib/drbd/crm-fence-peer.sh";
                after-resync-target "/usr/lib/drbd/crm-unfence-peer.sh";
        }

        disk {
                fencing resource-only;
                on-io-error detach;
        }

        net {
                after-sb-0pri discard-zero-changes;
                after-sb-1pri discard-secondary;
                after-sb-2pri disconnect;
        }

        on crash-1 {
                address 10.0.0.1:7788;
        }

        on crash-2 {
                address 10.0.0.2:7788;
        }
}
```

Il file relativo alla risorsa r1, sarà nominato /etc/drbd.d/r1_share-b.conf e varierà dal presente solo per il nome della risorsa (r1 invece di r0), il device /dev/drbd1 invece di /dev/drbd0), il parametro disk (/dev/vg_store/lv_share-b invece di lv_share-a) e per la porta su cui avverrà la comunicazione (7789 invece di 7788).

Tutti gli handler relativi al fencing consentiranno l'iterazione diretta tra DRBD e Pacemaker.

Definizione delle partizioni che verranno utilizzate da drbd (da eseguire su entrambi i nodi):

```
# pvcreate /dev/sda3
# vgcreate vg_store /dev/sda3
# lvcreate -L 100M -n lv_share-a vg_store
# lvcreate -L 100M -n lv_share-b vg_store
```

Creazione dei device drbd e prima sincronizzazione (su entrambi i nodi):

```
# drbdadm create-md r0
Writing meta data...
initializing activity log
NOT initialized bitmap
New drbd meta data block successfully created.
Success
# drbdadm create-md r1
Writing meta data...
initializing activity log
NOT initialized bitmap
New drbd meta data block successfully created.
Success
# drbdadm attach r0
# drbdadm syncer r0
# drbdadm connect r0
# drbdadm attach r1
# drbdadm syncer r1
# drbdadm connect r1
```

Infine su uno dei due nodi:

```
# drbdadm -- --overwrite-data-of-peer primary r0
# drbdadm -- --overwrite-data-of-peer primary r1
```

L'ultimo comando forzerà la prima sincronizzazione rendendo, dopo alcuni minuti, i due device drbd disponibili al sistema.

MMUL
http://www.mmul.it

Riepilogo mattutino

Di cosa abbiamo parlato

C'è qualcosa di poco chiaro?

Ci sono domande?

MMUL
http://www.mmul.it

Quarta parte

Il progetto

Gli share che verrano condivisi mediante NFS appoggeranno sui device DRBD, i quali però dovranno avere un file system in modo che il sistema possa leggervi e scrivervi all'interno:

```
# mke2fs -j /dev/drbd0
mke2fs 1.41.12 (17-May-2010)
Etichetta del filesystem=
Tipo SO: Linux
Dimensione blocco=1024 (log=0)
Dimensione frammento=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
25688 inode, 102360 blocchi
5118 blocchi (5.00%) riservati per l'utente root
Primo blocco dati=1
Maximum filesystem blocks=67371008
13 gruppi di blocchi
8192 blocchi per gruppo, 8192 frammenti per gruppo
1976 inode per gruppo
Backup del superblocco salvati nei blocchi:
        8193, 24577, 40961, 57345, 73729

Scrittura delle tavole degli inode: fatto
Creating journal (4096 blocks): fatto
Scrittura delle informazioni dei superblocchi e dell'accounting del filesystem: fatto

Questo filesystem verrà automaticamente controllato ogni 24 mount, o
180 giorni, a seconda di quale venga prima. Usare tune2fs -c o -i per cambiare.
```

La stessa operazione andrà eseguita per il device /dev/drbd1. Dal termine di queste operazioni, *che andranno eseguite OBBLIGATORIAMENTE dove drbd è in stato primary*, il sistema sarà in grado di montare le partizioni e scriverle di conseguenza.
Le directory su cui i device drbd verranno montati saranno /share-a e /share-b, che andranno di conseguenza create:

```
# mkdir /share-a
# mkdir /share-b
```

MMUL
http://www.mmul.it

Predisposizione del sistema: Installazione del demone NFS

- Il sistema necessiterà del demone NFS

```
# apt-get install nfs-kernel-daemon
```

- I demoni necessari all'erogazione del servizio sono tre:
  - `/etc/init.d/portmap`
  - `/etc/init.d/nfs-common`
  - `/etc/init.d/nfs-kernel-server`
- La configurazione del demone (/etc/exports) verrà gestita dal cluster

MMUL
http://www.mmul.it

Predisposizione sistema: disabilitazione avvio automatico servizi al boot

- I servizi che verranno gestiti dal cluster è essenziale vengano disabilitati al boot:

```
# update-rc.d -f portmap remove
# update-rc.d -f nfs-common remove
# update-rc.d -f nfs-kernel-server remove
# update-rc.d -f drbd remove
```

- Nelle distribuzioni Red Hat e derivate il comando sarà:

```
# chkconfig <servizio> off
```

Il comando crm può essere utilizzato anche per stampare a video la configurazione attuale (con tanto di syntax highlight) del cluster:

```
# crm configure show
node crash-1
node crash-2
property $id="cib-bootstrap-options" \
       dc-version="1.0.9-da7075976b5ff0bee71074385f8fd02f296ec8a3" \
       cluster-infrastructure="openais" \
       Expected-quorum-votes="2"
```

Similmente è possibile visualizzare anche la configurazione in versione XML:

```
# crm configure show xml
<?xml version="1.0" ?>
<cib admin_epoch="0" cib-last-written="Sat Apr  9 05:14:41 2011" crm_feature_set="3.0.1" dc-
uuid="crash-1" epoch="15" have-quorum="1" num_updates="5" validate-with="pacemaker-1.0">
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <nvpair id="cib-bootstrap-options-dc-version" name="dc-version" value="1.0.9-
da7075976b5ff0bee71074385f8fd02f296ec8a3"/>
        <nvpair id="cib-bootstrap-options-cluster-infrastructure" name="cluster-infrastructure"
value="openais"/>
        <nvpair id="cib-bootstrap-options-expected-quorum-votes" name="expected-quorum-votes"
value="2"/>
      </cluster_property_set>
    </crm_config>
    <rsc_defaults/>
    <op_defaults/>
    <nodes>
      <node id="crash-1" type="normal" uname="crash-1"/>
      <node id="crash-2" type="normal" uname="crash-2"/>
    </nodes>
    <resources/>
    <constraints/>
  </configuration>
</cib>
```

STONITH (*Shoot The Other Node In The Head*) rappresenta la capacità del cluster di eliminare ogni dubbio su chi sta gestendo una risorsa. In situazioni di connettività precaria, in cui non è ben chiaro chi sta gestendo un'area comune (in particolare quando si utilizzano filesystem condivisi) il cluster effettua il fence (reset istantaneo) dei nodi interessati in modo da mantenere un unico sopravvissuto a scrivere nell'area condivisa.
In soluzioni produttive che prevedono condizioni di filesystem comuni non si può prescindere da STONITH.
Maggiori informazioni su fence e STONITH presso il capitolo Fencing and Stonith della documentazione Pacemaker:
http://www.clusterlabs.org/doc/crm_fencing.html

QUORUM: il quorum rappresenta un'area logica comune che definisce la validità o meno di un nodo per l'erogazione di una risorsa. Tale validità nei cluster con un numero di nodi superiori a due è rappresentata dalla votazione espressa dalla maggioranza dei nodi sopravvissuti ad una situazione anomala. Se due nodi su tre stabiliscono che un nodo possa erogare una risorsa, il cluster permette al nodo scelto di erogarla.
In situazioni in cui il quorum viene perso in seguito ad un malfunzionamento (ad esempio cluster a due nodi), va stabilito il comportamento del cluster.
Di default situazioni di no-quorum-policy comportano uno "stop" della risorsa. Se quindi un nodo erogante viene spento o per qualche ragione smette di funzionare, allora nessun altro nodo erogherà la risorsa che risulterà quindi in stato "stop". Ma tale proprietà del cluster (no-quorum-policy, appunto) è modificabile con altri valori consentiti:

ignore – continua a gestire le risorse
freeze – continua a gestire le risorse, ma non ripristinare risorse da nodi falliti
stop – ferma tutte le risorse
suicide – effettua un fence (utilizzando *STONITH*) dei nodi falliti

Nell'esempio trattato l'assenza di quorum viene semplicemente ignorata.
Maggiori informazioni nella documentazione Pacemaker:
http://www.clusterlabs.org/doc/en-US/Pacemaker/1.1/html/Pacemaker_Explained/s-cluster-options.html

I comandi in sequenza da lanciare sono i seguenti:

```
# crm configure primitive share-a_r0 ocf:linbit:drbd \
params drbd_resource="r0" \
op monitor interval="20s" timeout="40s" \
op start interval="0" timeout="240s" \
op stop interval="0" timeout="100s"

# crm configure primitive share-b_r1 ocf:linbit:drbd \
params drbd_resource="r1" \
op monitor interval="20s" timeout="40s" \
op start interval="0" timeout="240s" \
op stop interval="0" timeout="100s"
```

Ciascun comando oltre a definire la risorsa stabilisce anche le tempistiche di monitor, di start e di stop. La definizione delle risorse multi state è invece la seguente:

```
# crm configure ms share-a_ms-r0 share-a_r0 meta master-max="1" notify="true"
# crm configure ms share-b_ms-r1 share-b_r1 meta master-max="1" notify="true"
```

Viene indicato come ciascuna risorsa può avere un solo nodo master (master-max) il quale una volta assunto il proprio stato (primary o secondary che sia) deve notificare il successo agli altri nodi (notify), in modo che la risorsa sia in uno stato coerente su entrambi i nodi.
Al termine delle operazioni, lo stato del cluster sarà il seguente:

```
# crm status
…
Online: [ crash-2 crash-1 ]

 Master/Slave Set: share-a_ms-r0
     Masters: [ crash-1 ]
     Slaves: [ crash-2 ]
 Master/Slave Set: share-b_ms-r1
     Masters: [ crash-2 ]
     Slaves: [ crash-1 ]
…
```

E' possibile che il resoconto riporti alcune “Failed actions”, dovute alla creazione iniziale delle risorse drbd: ciò è dovuto al fatto che lo stato di tali risorse non era ancora definito (non esisteva ancora la risorsa multi state). Eliminare questi “falsi positivi” sarà possibile mediante il comando *crm resource cleanup share-a_ms-r0* (e *share-b_ms-r1*)

Situazione del cluster dopo queste modifiche:

```
============
Last updated: Mon May  9 23:44:03 2011
Stack: openais
Current DC: crash-1 - partition with quorum
Version: 1.0.9-da7075976b5ff0bee71074385f8fd02f296ec8a3
2 Nodes configured, 2 expected votes
3 Resources configured.
============

Online: [ crash-2 crash-1 ]

 Master/Slave Set: share-a_ms-r0
     Masters: [ crash-1 ]
     Slaves: [ crash-2 ]
 Master/Slave Set: share-b_ms-r1
     Masters: [ crash-2 ]
     Slaves: [ crash-1 ]
 Clone Set: nfs_clone
     Started: [ crash-2 crash-1 ]
```

Il gruppo nfs_clone risiede contemporaneamente su entrambi i nodi.

Dettaglio dei comandi lanciati:

```
# crm configure primitive share-a_fs ocf:heartbeat:Filesystem \
       params device="/dev/drbd0" directory="/share-a" fstype="ext3" \
       op monitor interval="20s" timeout="40s" \
       op start interval="0" timeout="60s" \
       op stop interval="0" timeout="60s"

# crm configure primitive share-a_ip ocf:heartbeat:IPaddr2 \
       params ip="192.168.1.200" nic="eth0" \
       op monitor interval="20s" timeout="40s"

# crm configure primitive share-b_fs ocf:heartbeat:Filesystem \
       params device="/dev/drbd1" directory="/share-b" fstype="ext3" \
       op monitor interval="20s" timeout="40s" \
       op start interval="0" timeout="60s" \
       op stop interval="0" timeout="60s"

# crm configure primitive share-b_ip ocf:heartbeat:IPaddr2 \
       params ip="192.168.1.201" nic="eth0" \
       op monitor interval="20s" timeout="40s"
```

Dettaglio dei comandi lanciati:

```
# crm configure primitive share-b_exportfs ocf:heartbeat:exportfs \
       params directory="/share-b" clientspec="192.168.1.0/24"
options="rw,async,no_subtree_check,no_root_squash" fsid="1" \
       op monitor interval="10s" timeout="30s" \
       op start interval="0" timeout="40s" \
       op stop interval="0" timeout="40s"

# crm configure primitive share-a_exportfs ocf:heartbeat:exportfs \
       params directory="/share-a" clientspec="192.168.1.0/24"
options="rw,async,no_subtree_check,no_root_squash" fsid="1" \
       op monitor interval="10s" timeout="30s" \
       op start interval="0" timeout="40s" \
       op stop interval="0" timeout="40s"

# crm configure group share-a share-a_ip share-a_fs share-a_exportfs

# crm configure group share-b share-b_ip share-b_fs share-b_exportfs
```

Lo status mostra i gruppi in questa forma:

```
…
 Resource Group: share-a
     share-a_ip (ocf::heartbeat:IPaddr2):       Started crash-1
     share-a_fs (ocf::heartbeat:Filesystem):    Started crash-1
     share-a_exportfs   (ocf::heartbeat:exportfs):      Started crash-1
 Resource Group: share-b
     share-b_ip (ocf::heartbeat:IPaddr2):       Started crash-2
     share-b_fs (ocf::heartbeat:Filesystem):    Started crash-2
     share-b_exportfs   (ocf::heartbeat:exportfs):      Started crash-2
…
```

Purtroppo però, esiste anche un nutrito numero di “Failed actions”:

```
    share-a_fs_monitor_0 (node=crash-2, call=26, rc=5, status=complete): not installed
    ...
    share-b_fs_monitor_0 (node=crash-1, call=25, rc=5, status=complete): not installed
```

Ciò è dovuto al fatto che la risorsa ha tentato di avviarsi dove drbd era secondary, fallendo. Il problema viene risolto collocando le risorse.

Dettaglio dei comandi lanciati:

Ciascun vincolo ha associato un peso. Tutto all'interno del cluster ha un peso e tale peso è essenziale per definire se un nodo è in grado o meno di erogare una risorsa.

In questo caso viene definita per entrambi i gruppi share un vincolo di tipo colocation di peso infinito (inf, quindi senza possibilità di variazione) per cui il gruppo si avvierà solo laddove la risorsa multi-state è in stato master:

```
# crm configure colocation share-a_on_share-a_ms-r0 inf: share-a share-a_ms-r0:Master
# crm configure colocation share-b_on_share-b_ms-r1 inf: share-b share-b_ms-r1:Master
```

L'altra condizione necessaria per aggiungere coerenza alla configurazione riguarda l'ordinamento nell'avvio delle risorse. Ciascun gruppo share non ha ragione di avviarsi se non dopo che il dispositivo drbd è attivo. Pacemaker gestisce la cosa attraverso le definizioni order:

```
# crm configure order share-a_AFTER_share-a_ms-r0 inf: share-a_ms-r0:promote share-a:start
# crm configure order share-b_AFTER_share-b_ms-r1 inf: share-b_ms-r1:promote share-b:start
```

Viene definito un vincolo order di peso infinito in cui prima viene promossa a master la risorsa multi-state e dopo viene avviato il gruppo share.

A questo punto una cleanup dei gruppi share consentirà di avere una situazione pulita, senza gli errori in origine creati dall'assenza dei vincoli ora definiti:

```
# crm resource cleanup share-a
# crm resource cleanup share-b
```

Il comando crm status restituisce lo stato del cluster senza "Failed actions".

Dettaglio dei comandi lanciati:

```
# crm configure primitive ping ocf:pacemaker:ping \
params host_list="192.168.1.254" name="ping" \
op monitor interval="60s" timeout="60s" \
op start interval="0" timeout="60s"
```

Il comando crea una risorsa (primitive) denominata ping, il cui Resource Agent (RA) è di tipo OCF, fornito da pacemaker.
Il RA supporta diversi parametri, fra questi host_list deve contenere la lista degli host da controllare per verificare la connettività (in questo caso il gateway di rete) e name dovrà corrispondere al nome interno della risorsa (si veda la verifica della connettività illustrata nella prossima slide).
Infine le indicazioni che seguono le voci “op” (abbreviazione di operations) consentono di definire le politiche di monitoring della risorsa: come deve essere controllata (ad intervalli di 10 secondi e con 60 secondi di timeout) ed in quanto tempo si deve avviare e fermare (entro i 60 secondi di timeout).

La definizione della risorsa ping non è però sufficiente per fornire ad ogni nodo uno strumento per il controllo della connettività, poiché, definita nel modo indicato, la risorsa ping verrà eseguita su un unico nodo. Proprio per ovviare a questo problema e fare in modo che la stessa risorsa possa girare su più nodi è possibile clonarla:

```
# crm configure clone ping_clone ping meta globally-unique="false"
```

Il comando indica al cluster di creare la risorsa ping_clone che girerà su tutti i nodi, indistintamente (meta globally-unique=”false”).

Dettaglio dei comandi lanciati:

```
# crm configure location share-a_ON_connected_node share-a \
rule -inf: not_defined ping or ping lte 0

# crm configure location share-b_ON_connected_node share-b \
rule -inf: not_defined ping or ping lte 0
```

Vengono definiti due vincoli denominati *share-a_ON_connected_node* e *share-a_ON_connected_node* associati alle risorsa share che prevedono di dare il peso di meno infinito (-inf, rendere cioè incapace di erogare risorse) al nodo in cui la risorsa nominata ping non sia definita (not_defined) o il cui codice di uscita sia minore o uguale a zero (lte 0 e quindi il comando ping non ha avuto esito positivo, il ping node non risponde).

Stato finale del cluster:

```
# crm status
…
…
Online: [ crash-2 crash-1 ]

 Master/Slave Set: share-a_ms-r0
     Masters: [ crash-1 ]
     Slaves: [ crash-2 ]
 Master/Slave Set: share-b_ms-r1
     Masters: [ crash-2 ]
     Slaves: [ crash-1 ]
 Clone Set: nfs_clone
     Started: [ crash-2 crash-1 ]
 Resource Group: share-a
     share-a_ip (ocf::heartbeat:IPaddr2):       Started crash-1
     share-a_fs (ocf::heartbeat:Filesystem):    Started crash-1
     share-a_exportfs   (ocf::heartbeat:exportfs):      Started crash-1
 Resource Group: share-b
     share-b_ip (ocf::heartbeat:IPaddr2):       Started crash-2
     share-b_fs (ocf::heartbeat:Filesystem):    Started crash-2
     share-b_exportfs   (ocf::heartbeat:exportfs):      Started crash-2
 Clone Set: ping_clone
     Started: [ crash-2 crash-1 ]
```

Come osservare i comportamenti del cluster:

Mediante l'utilità del cluster *crm_mon*:

```
# crm_mon
```

Mediante l'utilità del cluster *crm_verify*:

```
# crm_verify -LV
```

Attraverso i log di sistema, su sistemi Debian/Ubuntu:

```
# tail -f /var/log/syslog
```

Su sistemi SuSe e RedHat:

```
# tail -f /var/log/messages
```

I log sono copiosissimi, pertanto è sempre meglio isolare mediante filtri l'output dei comandi (ad esempio *grep*).

MMUL
http://www.mmul.it

Riepilogo pomeridiano

Di cosa abbiamo parlato

C'è qualcosa di poco chiaro?

Ci sono domande?

MMUL
http://www.mmul.it

Conclusioni

Saluti!