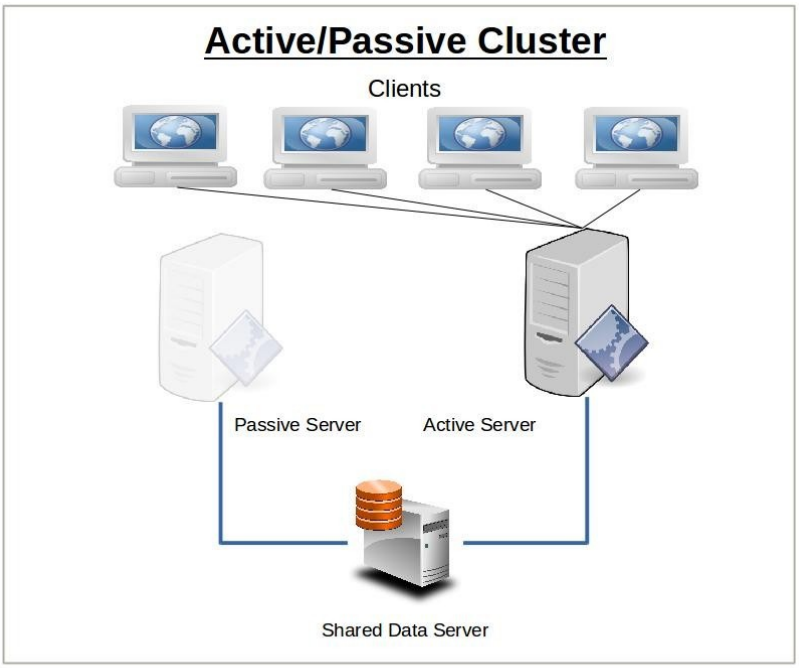
Corso di calcolo distribuito e sistemi ad alte prestazioni
Relazione: High Availability Cluster

De Angelis Fabio
Matricola: **318037**

Introduzione

In questa relazione viene presentato un progetto svolto nell'ambito di corso di Calcolo Distribuito e Sistemi ad Alte Prestazioni, che consiste nella realizzazione di High Availability Cluster.

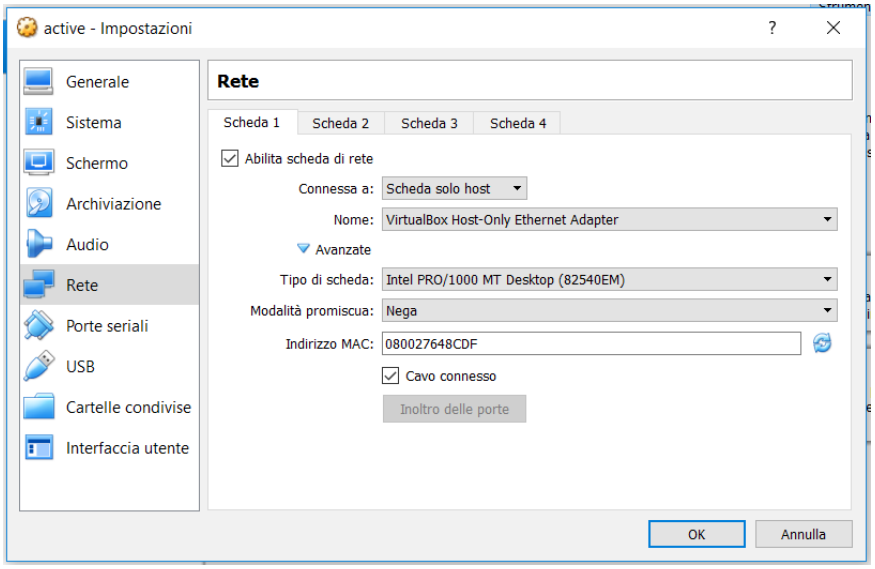
Un Cluster è un insieme di computer connessi tra loro tramite una rete, con lo scopo di distribuire un'elaborazione complessa tra le varie macchine che lo compongono, andando quindi ad implementare un unico sistema con una notevole potenza di calcolo.
Gli High Availability Cluster (HA) sono una particolare tipologia di cluster pensata per garantire la continuità dei servizi erogati.
Nello specifico in questo progetto, è stato realizzato un cluster virtuale composto da due macchine virtuali collegate tra loro in configurazione attiva/passiva.

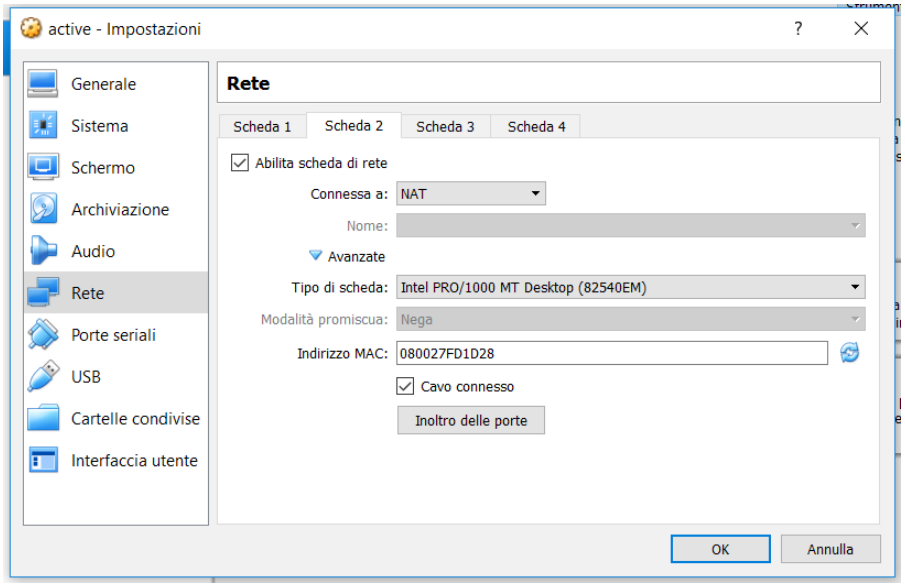


In una configurazione attiva / passiva (o asimmetrica), le applicazioni vengono eseguite su un server primario o master. Un server dedicato ridondante è presente per subentrare in caso di guasto, ma a parte questo non è configurato per eseguire altre funzioni. Quindi, in qualsiasi momento, uno dei nodi è attivo e l'altro è passivo. Questa configurazione fornisce un'istanza completamente ridondante di ciascun nodo, che viene portata in linea solo quando il relativo nodo primario non riesce.
Il cluster attivo / passivo contiene generalmente due nodi identici. Le istanze singole delle applicazioni di database sono installate su entrambi i nodi, ma il database si trova nell'archiviazione condivisa. Durante il normale funzionamento, l'istanza del database viene eseguita solo sul nodo attivo. In caso di guasto del sistema primario attualmente attivo, il software di clusterizzazione trasferirà il controllo del sottosistema del disco al sistema secondario. Come parte del processo di failover, viene avviata l'istanza del database sul nodo secondario, riprendendo quindi il servizio.

Primo Step: Creazione e configurazione della rete

Innanzitutto per procedere alla creazione della rete virtuale è stato scaricato e installato il software VirtualBox che permette la creazione e la gestione di macchine virtuali.
Sono state create due macchine virtuali: active e passive (rispettivamente attiva e passiva) con sistema operativo Fedora (distribuzione Linux) a cui sono stati assegnati 20gb di memoria e 2 di ram.
Una volta create le due macchine, si procede andando a configurare la rete andando innanzitutto a definire le interfacce di rete di entrambe le macchine dalle impostazioni di VirtualBox: si imposta come scheda di rete primaria (enp0s3) una scheda di rete host-only e come scheda secondaria (enp0s8) una scheda NAT.





In seguito si avviano le due macchine e si va a completare la configurazione di rete assegnando indirizzi IP statici modificando il file di configurazione `/etc/sysconfig/network-scripts/ifcfg-enp0s3` tramite l'editor di sistema vi nel seguente modo:

Una volta terminata la configurazione si riavvia il network service.
Prima di passare alla successiva si definiscono gli hostname associati agli indirizzi modificando il file */etc/hosts/* come mostrato di seguito:

Secondo Step: Installazione Ambiente Software

La fase successiva consiste nell'installazione di Pacemaker, ovvero il software utilizzato per la creazione del cluster.
Per avviare l'installazione è sufficiente digitare il seguente comando da terminale:
sudo dnf -y install pacemaker pcs
Così facendo verrà installato pacemaker con le sue dipendenza aggiuntive ovvero il demone pcs e corosync.
Una volta installato va abilitato e avviato attraverso i seguenti comandi:
sudo systemctl enable pcs
sudo systemctl start pcs
E` possibile inoltre controllare lo stato del demone pcs tramite il comando:
sudo systemctl pcs status

Ora che pcs è installato e abilitato si può procedere alla creazione del cluster.
Innanzitutto è necessario creare una password per l'autenticazione del cluster, infatti durante l'installazione viene creato un user di default "hacluster"a cui viene assegnata la password tramite il comando:
sudo passwd hacluster
Successivamente dal server node1, che sarà il nostro server attivo, si autenticano entrambi i nodi digitando il seguente comando:

```
[fabdea@localhost ~]$ sudo pcs cluster auth -u hacluster -p Fabio1995
primary: Not authorized
secondary: Not authorized
Nodes to authorize: primary, secondary
primary: Authorized
secondary: Authorized
```

Una volta autenticati i nodi si procede al setup del cluster:

```
[root@localhost ~]# pcs cluster setup examplecluster node1 node2 --force
No addresses specified for host 'node1', using 'node1'
No addresses specified for host 'node2', using 'node2'
Warning: node2: Cluster configuration files found, the host seems to be in a cluster already
Warning: node1: Cluster configuration files found, the host seems to be in a cluster already
Destroying cluster on hosts: 'node1', 'node2'...
node1: Successfully destroyed cluster
node2: Successfully destroyed cluster
Requesting remove 'pcsd settings' from 'node1', 'node2'
node1: successful removal of the file 'pcsd settings'
node2: successful removal of the file 'pcsd settings'
Sending 'corosync authkey', 'pacemaker authkey' to 'node1', 'node2'
node1: successful distribution of the file 'corosync authkey'
node1: successful distribution of the file 'pacemaker authkey'
node2: successful distribution of the file 'corosync authkey'
node2: successful distribution of the file 'pacemaker authkey'
Synchronizing pcsd SSL certificates on nodes 'node1', 'node2'...
node2: Success
node1: Success
Sending 'corosync.conf' to 'node1', 'node2'
node1: successful distribution of the file 'corosync.conf'
node2: successful distribution of the file 'corosync.conf'
Cluster has been successfully set up.
```

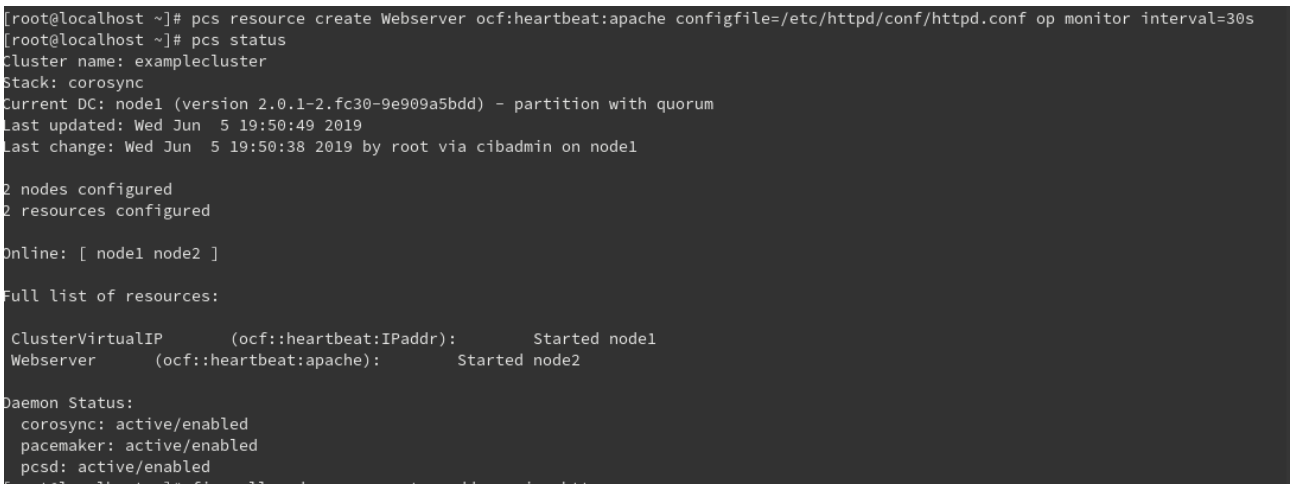
Ora che il cluster è stato effettivamente implementato non resta che abilitarlo e attivarlo, tramite i comandi:

```
sudo pcs enable --all
sudo pcs start --all
```

Avviato il cluster, e verificato il suo funzionamento occorre però configurare alcune importanti impostazioni.
Infatti si procede disabilitando Stonith e ordinando di ignorare la quorum policy.
In seguito si definisce una nuova risorsa, ovvero il cluster virtual (o floating) ip address , ovvero un indirizzo ip, nel nostro caso associato al cluster, che non è strettamente legato a una singola interfaccia.

```
[fabdea@localhost ~]$ sudo pcs property set stonith-enabled=false
[fabdea@localhost ~]$ sudo pcs property set no-quorum-policy=ignore
[fabdea@localhost ~]$ sudo pcs resource create Cluster_VIP ocf:heartbeat:IPaddr2 ip=127.0.0.2 cidr_netmask=24 op monitor interval=30s
[sudo] password di fabdea:
[fabdea@localhost ~]$ sudo pcs status
Cluster name: examplecluster
Stack: corosync
Current DC: node1.example.com (version 2.0.1-2.fc30-9e909a5bdd) - partition with quorum
Last updated: Tue May 21 11:22:08 2019
Last change: Tue May 21 11:21:52 2019 by root via cibadmin on node1.example.com

2 nodes configured
1 resource configured

Online: [ node1.example.com node2.example.com ]

Full list of resources:

 Cluster_VIP    (ocf::heartbeat:IPaddr2):       Started node1.example.com

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

Dobbiamo configurare il cluster in modo tale che solo il nodo attivo del cluster "possegga" o risponda all'IP mobile in un dato momento.
Si configurano quindi due risorse: oltre al Cluster_VIP definiamo una risorsa per il Webserver http di apache.
Nel nostro caso apache era già installato in Fedora, se così non fosse ovviamente è necessario provvedere alla sua installazione prima di creare la relativa risorsa.



Terzo Step: Creazione e assegnazione risorse drbd al cluster

DRBD (Distribuited Software Block Device) è una soluzione software distribuita flessibile e versatile per ambienti Linux. Permette di replicare il contenuto di block devices (come ad esempio hard disk, partizioni e logical volumes) tra due o più server.
Nell'ambito degli High Availabilty Cluster risulta di fondamentale importanza per garantire la continuità del servizio in quanto se dei dati memorizzati su un device specifico smettono di essere disponibili o leggibili a causa di una qualche failure, questi rimangono comunque disponibili in quanto sarà possibile accedervi da una copia disponibile su un altro nodo del cluster.

Innanzitutto è necessario installare DRBD tramite il seguente comando da terminale:
sudo dnf install drbd
A installazione completata è necessario aggiornare la policy di Linux per esentare i processi DRBD dal controllo di SELinux:

```
semanage permissive -a drbd_t
```

oltre a questo, è necessario far abilitare drbd al firewall attraverso la porta 7789 per permettergli di effettuare la copia dei dati tra i vari server, che altrimenti non sarà autorizzata.
Per farlo bisogna aggiungere il servizio drbd al firewall di ogni nodo del cluster:
-nodo1:

```
firewall-cmd --permanent  --add-rich-rule='rule family="ipv4"  source
address="192.168.56.102" port port="7789" protocol="tcp" accept'

firewall-cmd --reload
```

-nodo2:

```
firewall-cmd --permanent  --add-rich-rule='rule family="ipv4"  source
address="192.168.56.101" port port="7789" protocol="tcp" accept'

firewall-cmd --reload
```

Ora che sono state configurate le policy di sicurezza è finalmente possibile creare la risorsa drbd.
In questo progetto è stata scelta come tipologia di device un hard disk aggiuntivo virtuale per macchina creato tramite virtual box.
Una volta creati e collegati gli hard disk virtuali, si procede andando a configurare la risorsa drbd.
Innanzitutto si definisce la configurazione globale di DRBD editando il file `/etc/drbd.d/global_common.conf` nel modo seguente:

```
global {
    usage-count  yes;
 }
common {
    net {
        protocol C;
    }
}
```

Si può notare come sia stato assegnato a drbd il protocollo “protocol C” ovvero il protocollo comunemente usato per nodi a breve distanza all’interno di un network nonché quello più comunemente usato per replicare le risorse drbd.
Una volta definita la configurazione si creano le risorse drbd da assegnare al cluster.
Per farlo è necessario creare un nuovo file .res e salvarlo nella directory /etc/drbd.d/.

```
resource clust_res {
    on node1 {
        device /dev/drbd0;
        disk /dev/sdb;
        meta-disk internal;
        address 192.168.56.101:7789;
    }
    on node2 {
        device /dev/drbd0;
        disk /dev/sdb;
        meta-disk internal;
        address 192.168.56.102:7789;
        }
}
```

Il file denominato risorsa clust_res è così articolato:

- device /dev/drbd0: specifica il nuovo virtual block device gestito da drbd
- disk /dev/sdb1: la partizione che costituisce il device fisico della risorsa drbd
- meta-disk: definisce dove drbd memorizza i suoi metadati. Usando “internal” saranno salvati all’interno del device fisico definito in disk
- address: specifica indirizzo IP e port number dei rispettivi nodi.

Dopo aver definito la risorsa deve essere creata effettivamente attraverso il seguente comando:

```
[fabdea@node1 ~]$ sudo drbdadm create-md clust_res
[sudo] password di fabdea:
initializing activity log
initializing bitmap (64 KB) to all zero
Writing meta data...
New drbd meta data block successfully created.
```

E’ possibile controllare lo stato della risorsa tramite il comando `drbadadm status`, che appena creata la risorsa sarà come disk status “Inconsistent”, questo perché è necessario comunicare a drbd tramite quale nodo del cluster si darà inizio alla sincronizzazione dei device.
Per farlo si esegue il comando `drbdadm primary --force clust_res` dal nodo in questione.
Una volta terminata la sincronizzazione lo stato di entrambi i dischi sarà “UpToDate” come si può notare nell’immagine seguente.

```
[fabdea@node1 ~]$ sudo drbdadm status clust_res
clust_res role:Secondary
  disk:Inconsistent
  peer role:Secondary
    replication:Established peer-disk:Inconsistent

[fabdea@node1 ~]$ sudo drbdadm primary --force clust_res
[fabdea@node1 ~]$ drbdadm status clust_res
clust_res role:Primary
  disk:UpToDate
  peer role:Secondary
    replication:SyncSource peer-disk:Inconsistent done:3.07
```

```
[fabdea@node2 ~]$ sudo drbdadm status clust_res
[sudo] password di fabdea:
clust_res role:Secondary
  disk:UpToDate
  peer role:Primary
    replication:Established peer-disk:UpToDate
```

Ultimo Step: Verifica efficienza e continuità del servizio

Una volta aggiunte le risorse drbd ad ogni nodo del cluster, l'implementazione del sistema può dirsi terminata e non resta che testarlo.
Innanzitutto è stata creata una pagina PHP che mostra le informazioni del cluster (come ad esempio user nome del nodo e indirizzo IP) collegandosi all'indirizzo https://localhost/server.php.
Per farlo è sufficiente creare il file e salvarlo in /var/www/html.
Di seguito sono mostrati il codice sorgente e la sua visualizzazione nel browser:

Apri — server.php — /var/www/html — Salva

```
<!DOCTYPE html>
<html>
<head></head>
<body>

<?php
$_SERVER=$GLOBALS['_SERVER'];
$ip=$_SERVER["REMOTE_HOST"] ?: gethostbyaddr($_SERVER["REMOTE_ADDR"]);
?>

<h2>NODE NAME:  </h2><?php print_r($_SERVER['HOSTNAME']); ?></br>
<h2>USER:       </h2><?php print_r($_SERVER['USERNAME']); ?></br>
<h2>IP ADDRESS: </h2><?php echo $ip; ?></br>
</body>
</html>
```

Attività — Firefox — mer 3 lug 19:30

Cluster info — localhost/server.php

NODE NAME:

node1

USER:

fabdea

IP ADDRESS:

192.168.56.101

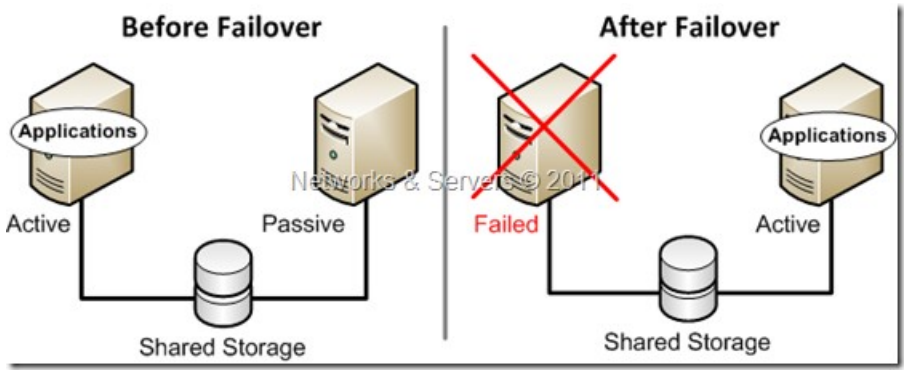
Infine, per verificare il cluster sia veramente un High Availability Cluster, ossia che garantisca sempre la continuità del servizio, è stato eliminato uno dei due nodi e verificato la stato del sistema, dimostrando come il servizio sia ancora attivo.
Quindi si procede eliminando un nodo, nel nostro caso node1:

```
[fabdea@node2 ~]$ sudo pcs cluster stop node1
node1: Stopping Cluster (pacemaker)...
node1: Stopping Cluster (corosync)...
```

per poi controllare lo stato del cluster:

```
[fabdea@node2 ~]$ sudo pcs status
Cluster name: mycluster
Stack: corosync
Current DC: node2 (version 2.0.1-2.fc30-9e909a5bdd) - partition with quorum
Last updated: Thu Jul  4 20:18:49 2019
Last change: Thu Jul  4 20:18:45 2019 by root via cibadmin on node2

2 nodes configured
2 resources configured

Online: [ node2 ]
OFFLINE: [ node1 ]

Full list of resources:

 ClusterVIP     (ocf::heartbeat:IPaddr2):       Started node2
 Webserver      (ocf::heartbeat:apache):        Starting node2

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
[fabdea@node2 ~]$
```

Possiamo notare come la risorsa ClusterVip che prima era in node1 ora sia “migrata” in node2, continuando a garantire la continuità del servizio.



Abbiamo quindi mostrato tutti i vantaggi dell’utilizzo degli High Availability Cluster, tra i quali la capacità di far fronte a failures che possono colpire i singoli nodi senza causare l’interruzione dei servizi offerti, oltre alle evidenti capacità di distribuzione di qualsivoglia carico di lavoro tra i vari nodi che lo compongono, caratteristiche che lo rendono un sistema distribuito ad alta potenza ed efficienza.