Realizzazione di un cluster con OpenNebula e JBoss: Fault Tollerance e Migration

Riccardo Contigiani
riccardo.contigiani2@studio.unibo.it
Mat: 0000784451

Stefano Traini
stefano.traini5@studio.unibo.it
Mat: 0000778487

April 30, 2016

1 Abstract

Un cluster è un insieme di elaboratori, connessi tra di loro tramite la rete, con lo scopo di distribuire l'elaborazione tra le varie macchine che compongono il cluster.
Questo documento mostra come configurare un cluster virtuale, attraverso l'utilizzo di un computer e dei software OpenNebula, Jboss e KVM, al fine di implementare le funzionalità di migrazione e di tolleranza ai guasti che permettono di mantenere lo stato di un'applicazione web anche se uno o più dei web server che compongono il cluster subisce un crash o viene spento.

2 Introduzione

OpenNebula è un software open-source che permette di gestire centri di elaborazione distribuiti ed eterogenei. Esso è adatto per creare una Infrastructure as a Service (IaaS) ossia gestisce le tecnologie di memoria, la rete, la monitorizzazione della virtualizzazione e la sicurezza per implementare servizi multi-livello come macchine virtuali su infrastrutture distribuite.
I requisiti per poterlo utilizzare sono:

- Sistema Operativo con kernel Linux;
- Supporto alla virtualizzazione da parte dell'hardware.

Per interagire con il software è possibile utilizzare l'interfaccia da linea di comando o in alternativa l'interfaccia grafica Sunstone. In questo progetto OpenNebula è servito per creare l'infrastruttura di rete. KVM (Kernel-based Virtual Machine) è un'infrastruttura di virtualizzazione del kernel linux che supportata dai computer con architettura Intel o AMD. Grazie all'utilizzo di KVM è stato possibile virtualizzare i server che avrebbero contenuto la web application. Jboss è un application server open-source realizzato in Java e distribuito da RedHat Corporation. È stato installato all'interno delle macchine virtuali per permettere il funzionamento dell'applicazione web e la creazione del cluster, grazie alla funzionalità mod-cluster nativa di Jboss.

3 Dettagli Tecnici

3.1 Hardware utilizzato

L'architettura utilizzata è composta da una sola macchina host che si occupa di gestire sia OpenNebula che le macchine virtuali.
La macchina Host ha le seguenti caratteristiche:

- CPU: Intel® i7-2670QM @ 2.20 GHz
- Memoria: 8GB DDR3
- Disco rigido: 750GB
- Sistema Operativo: GNU/Linux Ubuntu Gnome 14.04 LTS, 64 bit, kernel 3.19.0
- Indirizzo di rete: 10.0.0.1

La macchina virtuale Master ha le seguenti caratteristiche:

- CPU: 1
- Memoria: 1GB
- Disco rigido virtuale: 10GB
- Sistema Operativo: GNU/Linux Debian 8.2.0 64bit netinstall
- Interfaccia di virtualizzazione: KVM
- Indirizzo di rete: 10.0.0.100

La macchina virtuale Slave01 ha le seguenti caratteristiche:

- CPU: 1
- Memoria: 1GB
- Disco rigido virtuale: 10GB
- Sistema Operativo: GNU/Linux Debian 8.2.0 64bit netinstall
- Kernel Linux 3.16
- Interfaccia di virtualizzazione: KVM
- Indirizzo di rete: 10.0.0.101

La macchina virtuale Slave02 ha le seguenti caratteristiche:

- CPU: 1
- Memoria: 1GB
- Disco rigido virtuale: 10GB
- Sistema Operativo: GNU/Linux Debian 8.2.0 64bit netinstall
- Kernel Linux 3.16
- Interfaccia di virtualizzazione: KVM
- Indirizzo di rete: 10.0.0.102

3.2 Configurazione dell'Host

All'interno della macchina host è sono stati installati i pacchetti relativi al funzionamento di OpenNebula 4.6 tramite il seguente comando:

```
$sudo apt-get install opennebula
   opennebula-node opennebula-tools
   opennebula-commons opennebula-
   sunstone
```

Successivamente è stato creato l'utente oneadmin tramite il comando:

```
#adduser oneadmin
```

e gli sono stati concessi i privilegi di sudo aggiungendolo al gruppo super-user con il comando:

```
#adduser oneadmin sudo
```

Successivamente è stato necessario entrare nell'utente oneadmin ed installare il gestore della virtualizzazione kvm con il seguente comando:

```
$sudo apt-get install qemu-kvm
   libvirt-bin ubuntu-vm-builder
   bridge-utils
```

ed aggiungere l'utente ai gruppi kvm e libvirtd:

```
#adduser oneadmin kvm
#adduser oneadmin libvirtd
```

È stato oltretutto modificato il file `/etc/libvirt/quemu.conf`

```
user="oneadmin"
gruop="oneadmin"
dynamic_ownership=0
```

Il passaggio successivo è stato installare il server SSH (Secure Shell):

```
$sudo apt-get install openssh-server
```

modificare il file `/etc/ssh/sshd_config` settando il seguente parametro

```
PermitRootLogin without-password
```

ed infine generare una chiave SSH per permettere la creazione di una sessione cifrata tra l'host e il cluster attraverso il protocollo SSH:

```
$ ssh-keygen -t rsa
```

La chiave è contenuta all'interno del file `/var/lib/one/.ssh/id_rsa.pub` ed è stata copiata all'interno del file `/var/lib/one/.ssh/authorized_keys`.
Inoltre, per il corretto funzionamento dell'SSH, è stato modificato il file `/var/lib/one/.ssh/config` nel seguente modo:

```
Host *
StrictHostKeyChecking no
UserKnownHostsFile .ssh/known_hosts
```

A questo punto sono stati creati due script che devono essere avviati ogni qualvolta si effettua il login dell'utente oneadmin: Il primo è startService.sh che contiene i comandi necessari per avviare i servizi:

```
#!/bin/bash

/etc/init.d/opennebula-sunstone
   restart
/etc/init.d/networking restart
service libvirt-bin restart
```

Il secondo è bridge.sh che contiene i comandi per permettere il riconoscimento dell'host da parte di OpenNebula creando un ponte tra i due :

```
#!/bin/bash

brctl addbr smbr0
brctl addif smbr0 eth0
ifconfig smbr0 10.0.0.1 netmask
   255.0.0.0
ip addr
iptables -t nat -A POSTROUTING -o
   wlan0 -j MASQUERADE
iptables -A FORWARD -i smbr0 -j
   ACCEPT
```

Ora è possibile accedere a Sunstone, ossia l'interfaccia grafica fornita da OpenNebula per la gestione dell'infrastruttura. Per fare ciò è bastato collegarsi all'indirizzo `http://loclalhost:9869` con un comune browser. L'utente amministratore è oneadmin e la password è contenuta all'interno del file `/var/lib/one/.one/one_auth`

3.3 Configurazione delle VM

3.3.1 Aggiunta dell'Host

Il primo passo compiuto è quello di aggiungere l'host all'interno di OpenNebula. Di conseguenza è stato necessario entrare all'interno di Sunstone sotto la voce "Infrastructure→Hosts" ed aggiungere un nuovo host con i seguenti parametri:

- Type: KVM
- Name: h1
- Cluster: Default (none)
- Networking: Default (dummy)

3.3.2 Aggiunta della Virtual Network

Il secondo passo è quello dell'aggiunta di una rete virtuale alla quale si connetteranno la macchina master e le macchine slave. Per fare ciò bisogna recarsi nel menù "Infrastructure→Virtual Networks" ed aggiungere una nuova rete con i seguenti parametri:

- Name: net0
- Bridge: smbr0
- Network model: default
- IP Start: 10.0.0.100
- Size: 3

Così facendo sono stati riservati gli indirizzi da 10.0.0.100 a 10.0.0.102 per il master e gli slave.

3.3.3 Creazione dell'immagine master

Per le macchine virtuali è stata utilizzata l'immagine ISO della distribuzione GNU/Linux Debian 8.2.0 64bit netinstall. Inizialmente sotto la voce "Virtual Resources→Images" è stato necessario creare un datablock vuoto e un datablock di tipo CDROM.

Il primo è stato creato con lo scopo di fungere da disco rigido del master con i seguenti parametri:

- Name: DBempty
- Type: OS
- Persistent: yes
- Size: 10 GB
- Image location: Empty datablock

Il secondo ha lo scopo di contenere l'immagine ISO per poter avviare l'installazione del sistema operativo Debian con le seguenti caratteristiche:

- Name: Debian-img
- Type: CDROM
- Persistent: no
- Image location: Upload

Ed infine si è selezionata l'immagine ISO precedentemente scaricata.

3.3.4 Creazione del template master

Il primo template che si è creato è servito per istanziare la VM del master. Selezionando la voce "VirtualResources→Templates" si è creato un template con i seguenti parametri:

- Name: Debian-tmp
- Hypervisor: KVM
- Memory: 1024MB
- CPU: 1
- Storage Disk0: Debian-img
- Storage Disk1: DBempty
- Network Interface0: net0
- OS Booting 1st Boot: CDROM
- OS Booting 2nd Boot: HD
- Listen IP: 0.0.0.0
- Context Add SSH contextualization: yes
- Context Public Key: copiare il contenuto del file `/var/lib/one/.ssh/id_rsa.pub`

Una volta creato il template è stato sufficiente selezionarlo ed istanziarlo con il nome "master" ed una volta avviata la macchina virtuale è bastato seguire i passi per l'istallazione di Debian selezionando come hard disk il datablock DBempty.
Conclusa l'installazione di Debian è stato modificato il template Debian-tmp con i seguenti parametri:

- Storage Disk0: DBempty
- OS Booting 1st Boot: HD

ed eliminate le voci Sorage Disk1 e OS Booting 2nd Boot.
Avendo settato il campo SSH per poter utilizzare il master dal terminare dell'host è bastato collegarsi tramite terminale all'indirizzo della macchina virtuale con il seguente comando:

```
$ ssh root@10.0.0.100
```

Una volta effettuato il login è stato scaricato e decompresso l'applicativo Jboss AS 7.1.1. all'interno della cartella `/opt/jboss`. A questo punto per non ripetere le stesse operazioni per gli slave, dato che fino a questo punto hanno una configurazione comune, si è deciso di clonare due volte l'immagine DBempty andando così a creare le immagini "slave01" e "slave02".

3.3.5 Configurazione del master

All'interno della macchina virtuale master sono stati modificati alcuni file di configurazione.

Listing 1: ∼/domain/configuration/host.xml

```
<host name="master" xmlns="urn:jboss
   :domain:1.2">
\dots
<interfaces>
        <interface name="management"
           >
```

```
                <inet-address value=
                    "${jboss.bind.
                    address.
                    management
                    :10.0.0.100}"/>
        </interface>
        <interface name="public">
                <inet-address value=
                    "${jboss.bind.
                    address
                    :10.0.0.100}"/>
        </interface>
        <interface name="unsecured">
                <inet-address value=
                    "${jboss.bind.
                    address .unsecure
                    :10.0.0.100}" />
        </interface>
</interfaces>
```

Inoltre, per poter far comunicare il master con i suoi slave è stato necessario creare degli account per l'autenticazione attraverso lo script `~/bin/add-user.sh`. Lanciando lo script si ottiene:

Listing 2: Configurazione della sicurezza

```
What type of user do you wish to add
    ?
a) Management User (mgmt-users.
    properties)
b) Application User (application-
    user.properties)
        (a):
Enter the details of the new user to
     add.
Realm (ManagementRealm) :
Username : slave01/slave02/master
Password : password
Re-enter Password : password
About to add user 'slave' for realm
    'ManagementRealm'
Is this correct yes/no? yes
Added user 'slave' to file '/home/
    weli/projs/jboss-as-7.1.0.CR1b/
    standalone/configuration/mgmt-
    users.properties'
Added user 'slave' to file '/home/
    weli/projs/jboss-as-7.1.0.CR1b/
    domain/configuration/mgmt-users.
    properties'
```

E' stato necessario modificare anche:

Listing 3: ~/domain/configuration/domain.xml

```
\dots
<hornetq-server>
        <persistence-enabled>true</
            persistence-enabled>
        <cluster-user>jms-user</
            cluster-user>
        <cluster-password>simple-
            pass</cluster-password>
</hornetq-server>
\dots
<profile name="full-ha">
...
<subsystem xmlns="urn:jboss:domain:
    modcluster:1.0">
    <mod-cluster-config advertise-
        socket="modcluster" proxy-list
        ="10.0.0.100:10001">
          <dynamic-load-provider>
               <load-metric type="
                   busyness"/>
          </dynamic-load-provider>
     </mod-cluster-config>
</subsystem>
...
<profile />
```

Di conseguenza, per permettere il corretto funzionamento della funzionalità mod-cluster di Jboss è stato opportuno modificare il seguente file:

Listing 4: ~/opt/jboss/httpd/httpd/conf/httpd.conf

```
...
Listen 10.0.0.100:80
...
LoadModule slotmem_module modules/
    mod_slotmem.so
LoadModule manager_module modules/
    mod_manager.so
```

```
LoadModule proxy_cluster_module
   modules/mod_proxy_cluster.so
LoadModule advertise_module modules/
   mod_advertise.so
...

Listen 10.0.0.100:10001
MemManagerFile /var/cache/httpd

<VirtualHost 10.0.0.100:10001>

  <Directory />
    Order deny,allow
    Deny from all
    Allow from 10.0.0.
  </Directory>

  <Location /mod_cluster-manager>
   SetHandler mod_cluster-manager
   Order deny,allow
   Deny from all
   Allow from 10.0.0.
  </Location>

  KeepAliveTimeout 60
  MaxKeepAliveRequests 0

  ServerAdvertise On
 AdvertiseGroup 224.0.1.105:23364
AdvertiseFrequency 5

 ManagerBalancerName other-server-
    group
 EnableMCPMReceive

</VirtualHost>
```

Infine, è stato modificato anche il file:

Listing 5: ∼/domain/configuration/host.xml

```
<host name="master" xmlns="urn:jboss
   :domain:1.2">
...
<interfaces>
   <interface name="management">
      <inet-address value="${jboss.
         bind.address.management
         :10.0.0.100}"/>
   </interface>
   <interface name="public">
      <inet-address value="${jboss.
         bind.address:10.0.0.100}"/>
   </interface>
   <interface name="unsecure">
      <inet-address value="${jboss.
         bind.address.unsecure
         :10.0.0.100}"/>
   </interface>
</interfaces>
...
<server name="server-three" group="
   other-server-group" auto-start="
   true">
  <socket-bindings port-offset="250"
     />
</server>
</servers>
</host>
```

3.4 Configurazione delle macchine slave

Si è clonato due volte il template del master andando a creare “slave01-tmp” e “slave02-tmp” con l’unica eccezione che alla voce Storage Disk0 è stato selezionato dapprima il DataBlock slave01 e successivamente slave02 che sono stati precedentemente clonati. All’interno di ogni slave si è eliminato il file domain.xml perché queste macchine sono sotto il controllo del master.
In ogni slave sono state apportate le stesse modifiche, per comodità quindi i file modificati vengono mostrati solo per lo slave01.

Listing 6: ∼/domain/configuration/host.xml

```
<host name="slave01" xmlns="urn:
   jboss:domain:1.2">
...
<domain-controller>
        <remote host="10.0.0.100"
           port="9999" security-
```

```
            realm="ManagementRealm"/>
</domain-controller>

<security-realms>
        <security-realm name="
           ManagementRealm">
                <server-identities>
                <secret value="
                   cGFzc3dvcmQ="/>
        </server-identities>
        <authentication>
                <properties path="
                   mgmt-users.
                   properties"
                   relative-to="
                   jboss.domain.
                   config.dir"/>
        </authentication>
        </security-realm>
</security-realms>
...
<interfaces>
        <interface name="management"
           >
                <inet-address value=
                   "${jboss.bind.
                   address.
                   management
                   :10.0.0.101}"/>
        </interface>
        <interface name="public">
                <inet-address value=
                   "${jboss.bind.
                   address
                   :10.0.0.101}"/>
        </interface>
        <interface name="unsecured">
                <inet-address value=
                   "${jboss.bind.
                   address .unsecure
                   :10.0.0.101}" />
        </interface>
</interfaces>
...
<server name="server-three-slave01"
   group="other-server-group" auto-
   start="true">
    <socket-bindings port-offset="
       250"/>
</server>
        </servers>
</host>
```

Dove il valore del tag <secret value> è la trasformazione in base64 della password.

3.5 Deployment

Per poter dimostrare le funzionalità di migrazione e di tolleranza ai guasti è stato necessario sviluppare un'applicazione stateful con l'ambiente di sviluppo Jboss Studio. Il programma in questione [1] si occupa di mostrare il timestamp attuale e contiene al suo interno la servlet `put.jsp`, che si occupa di acquisire il valore del timestamp e di salvarlo in una variabile globale, e la servlet `get.jsp`, che permette di visualizzare il valore della variabile globale precedentemente salvata.
In conclusione attraverso il compilatore di Jboss studio si è ottenuto il file cluster-demo.war che è stato poi caricato sui server attraverso la console di Jboss manager ed in fine sono stati concessi i permessi al gruppo di host desiderato sull'esecuzione del file.

3.6 Testing dell'applicazione

Per eseguire l'applicazione il primo passo da compiere è quello di avviare httpd con il comando:

```
cd /opt/jboss/httpd/sbin
./apachectl start
```

ed avviare Jboss sul nodo master ed i nodi slave con il comando

```
cd /opt/jboss/jboss-as-7.1.1.Final/
   bin/
./domain.sh
```

Successivamente collegarsi all'indirizzo `http://10.0.0.100:9990/console/` per entrare all'interno del manager di Jboss e da qui verificare che tutte e tre le macchine virtuali siano in esecuzione.

Successivamente per verificare il valore del timestamp è bastato collegarsi all'indirizzo `http://10.0.0.100/cluster-demo/`. Per dimostrare che la funzione di migrazione opera correttamente è necessario recarsi all'indirizzo `http://10.0.0.100/cluster-demo/put.jsp` in modo tale che la variabile di sessione sia salvata, dopodiché si è tornati all'interno del manager di Jboss e si è simulato un guasto al master spegnendolo. Infine si è richiamata la sessione all'indirizzo `http://10.0.0.100/cluster-demo/get.jsp` dimostrando così la tolleranza ai guasti.

4 Conclusioni

In conclusione si è dimostrato come sia possibile configurare e gestire un cluster, garantendo le funzionalità di migrazione e di tolleranza ai guasti.
Le difficoltà riscontrate sono principalmente derivate dalla scarsa documentazione presente nel web dovuta probabilmente alla natura open-source del software che purtroppo non ha una community estesa. Altre difficoltà riscontrate sono state causate dalla scarsa flessibilità di alcuni programmi, infatti è stato difficoltoso trovare la giusta combinazione di versioni dei software da utilizzare e questo è stato possibile solo dopo un lungo e duro lavoro di ricerche nella rete e test sulla macchina fisica. A dimostrare ciò è sufficiente far notare che è stata utilizzata la versione Jboss AS 7 ormai obsoleta invece di usare la più recente Jboss EAP, poiché con quest'ultima non si è riusciti ad ottenere una configurazione del cluster stabile e priva di crash.

References

[1] As7 cluster howto. `https://docs.jboss.org/author/display/AS71/AS7+Cluster+Howto`.

[2] Jboss. `http://www.jboss.org`.

[3] Mod cluster 1.2.0 quick start guide. `https://docs.jboss.org/mod_cluster/1.2.0/html/Quick_Start_Guide.html`.

[4] Opennebula documentation. `http://docs.opennebula.org/`.