Università di Bologna

Corso di Sistemi Middleware - CdL Informatica Magistrale

Realizzazione e deployment di un'applicazione distribuita JBoss su macchine virtuali con Opennebula

di Federico Montori e Margherita Lazzarini

8 maggio 2014

Sommario

Nello svolgimento di questo progetto abbiamo provveduto a installare e configurare Opennebula, un toolkit per la gestione distribuita di macchine virtuale in un ambiente di cloud computing, e istanziare un'applicazione stateful JBoss, un middleware basato su Java che include un cluster di application server. Lo scopo del progetto è la dimostrazione delle funzionalità di migrazione delle macchine virtuali fra diversi host fisici e le proprietà di fault tolerance dell'intero cluster.

1 Introduzione

Opennebula è un software open source che si occupa di gestire un'infrastruttura che ospita diverse tecnologie quali macchine, reti e memorie virtuali e di monitorarle fornendo servizi di cloud computing. Supporta una vasta gamma di interfacce cloud (Amazon EC2 Query, OGF Open Cloud Computing Interface, vCloud) e hypervisors (Xen, KVM, VMware).

JBoss AS, o semplicemente JBoss, è un application server open source multipiattaforma che implementa i servizi Java EE, dal 2006 propretà di Red Hat. In questo progetto è stata utilizzata la versione JBoss EAP 6 (Enterprise Application Platform).

Le macchine virtuali ospitanti i server JBoss sono il *member host*, su cui viene eseguita l'applicazione server, e il *domain controller*, che funge da gateway delle richieste HTTP ed effettua load balancing fra i vari member host. L'infrastruttura di virtualizzazione è istanziata su due tipi di host fisici: il front-end, che ha la funzione di gestire e monitorare le macchine virtuali e ospita il *domain controller*, e il worker node, che ospita uno dei member host del cluster. Nel nostro caso, come illustrato in Fig. 1, abbiamo utilizzato due macchine, una delle quali agisce semplicemente da worker node, una sia da worker node che da front-end. La funzionalità di migrazione permette il trasferimento a runtime di una macchina virtuale da un host fisico all'altro. La proprietà di fault tolerance garantisce il funzionamento dell'applicazione server anche dopo il crash di un worker node.

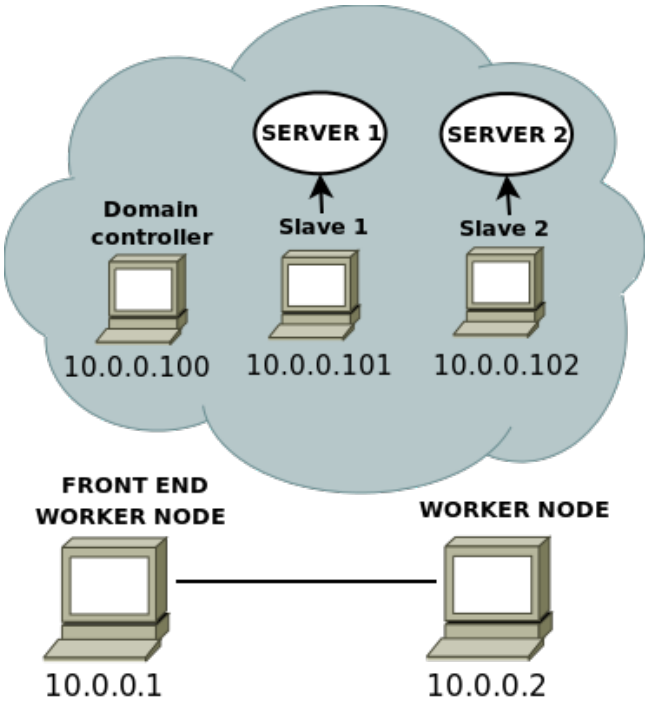

Figura 1: Scenario di sviluppo del progetto

2 Dettagli tecnici

Sugli entrambi gli host fisici è stato installato Debian 7.4 versione stable e, sul front-end, tutte le componenti necessarie per l'installazione di Opennebula, come illustrato nella documentazione di Opennebula (`http://docs.opennebula.org/4.6/design_and_installation/building_your_cloud/ignc.html#installing-on-debian-ubuntu`). In aggiunta, abbiamo installato il servizio NFS per la condivisione di aree di memoria:

```
$ sudo apt-get install nfs-common portmap nfs-kernel-server
```

La configurazione base del front-end termina con la configurazione dell'utente *oneadmin*, impostato di default come gestore di Opennebula, documentata anch'essa in (`http://docs.opennebula.org/4.6/design_and_installation/building_your_cloud/ignc.html#step-3-starting-opennebula`). In aggiunta, la coppia oneadmin:[[md5_of_password]] deve essere aggiunta al file *sunstone_auth*.
Nel worker node è necessario installare il pacchetto *opennebula-node* e configurare alla stessa maniera l'utente *oneadmin*. È altresì indispensabile che l'utente *oneadmin* abbia lo stesso user ID, e group ID su ogni macchina del cluster.
Infine, per fare sì che Opennebula sia in grado di monitorare gli host del

cluster (i.e. possa effettuare una connessione ssh automatica ad essi) occorre che il front-end contenga nel file .ssh/authorized_keys le chiavi pubbliche id_rsa.pub, generate con il comando ssh-keygen, di tutti gli host fisici appartenenti al cluster.

Per abilitare la comunicazione fra le due macchine, abbiamo utilizzato una connessione cablata e configurato un bridge tra gli indirizzi 10.0.0.1 e 10.0.0.2 asseganti rispettivamente al front-end e al worker node:

```
# brctl addbr smbr0
# brctl addif smbr0 eth0
# ifconfig smbr0 10.0.0.1 netmask 255.0.0.0 // 10.0.0.2 per il
    worker node
```

Per istanziare le macchine virtuali sui diversi host del cluster è necessario condividere la cartella che ne contiene i dati aggiungendo al file /etc/exports di tutti i nodi la linea:

```
/var/lib/one/datastores          *(rw, sync, no_subtree_check)
```

In seguito è possibile avviare sia il servizio NFS che il demone di Opennebula:

```
$ one start
$ sunstone-server start
$ sudo /etc/init.d/nfs-common start
$ sudo /etc/init.d/nfs-kernel-server start
```

Il worker node deve montare a sua volta la directory datastores:

```
# mount -tnfs4 10.0.0.1:/var/lib/one/datastores /var/lib/one/
    datastores
```

2.1 Configurazione di un cluster con Opennebula

A questo punto sarà possibile connettersi con l'interfaccia grafica di Opennebula Sunstone da browser all'indirizzo localhost:9869 con l'utente oneadmin e la sua password. Lo scenario richiede che vengano settati alcuni componenti:

- Creare un nuovo datastore di tipo System shared.
- Creare una nuova fixed Virtual Networks selezionando il bridge appena creato e tre nuovi indirizzi IP che verranno destinati alle macchine virtuali del cluster (nel nostro caso 10.0.0.100, 10.0.0.101 e 10.0.0.102).
- Creare due nuovi host KVM relativi agli host fisici.

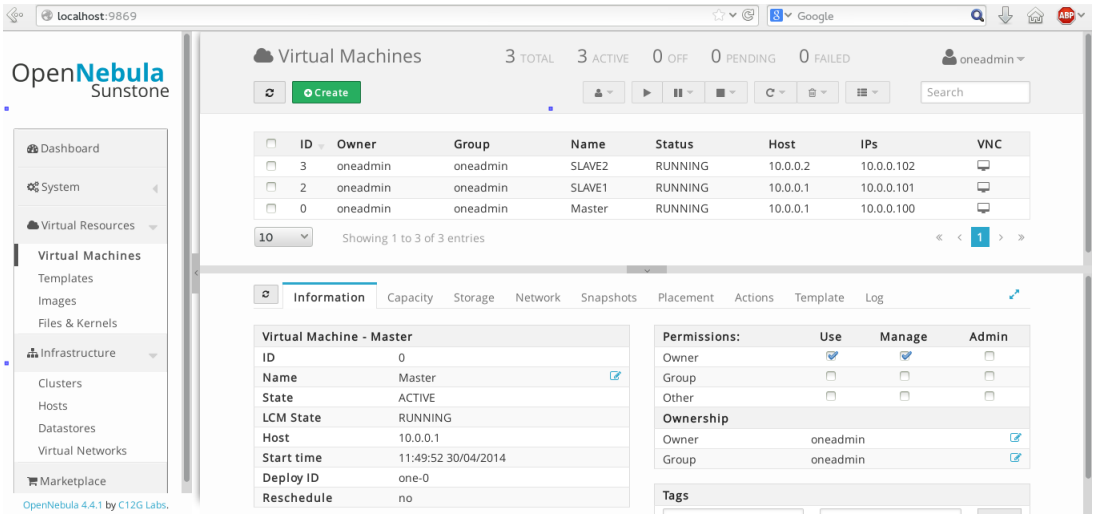

Figura 2: L'interfaccia Sunstone mostra le 3 macchine virtuali in stato RUNNING

- Creare un nuovo Cluster che includa gli host, i datastore e la Virtual Network.

Come primo approccio è stata importata l'immagine tty-linux-kvm e settato un nuovo template con parametri atti a istanziare una macchina virtuale associata a tutti gli elementi del cluster creato. Dopo aver istanziato il template, abbiamo tentato una migrazione live, ma dopo il fixing di vari errori inerenti a permessi, si è verificato un errore di tipo unexpected non documentato, probabilmente causato dall'eterongeneità hardware fra le due macchine.

2.2 Deploy di un'applicazione stateful JBoss

Una volta scaricata una ISO di Ubuntu 12.04 a 32 bit, abbiamo creato un'immagine di tipo CDROM e tre di tipo DATABLOCK nella sezione Images dell'interfaccia di Sunstone, caricando l'immagine di Ubuntu nella prima (che costituirà il CD di installazione per le macchine virtuali) e creando blocchi vuoti da 10GB per le altre (che costituiranno gli Hard Disk delle macchine virtuali). Abbiamo così creato 3 nuovi Template che hanno dato origine alle 3 macchine virtuali selezionando opportunamente le immagini desider-

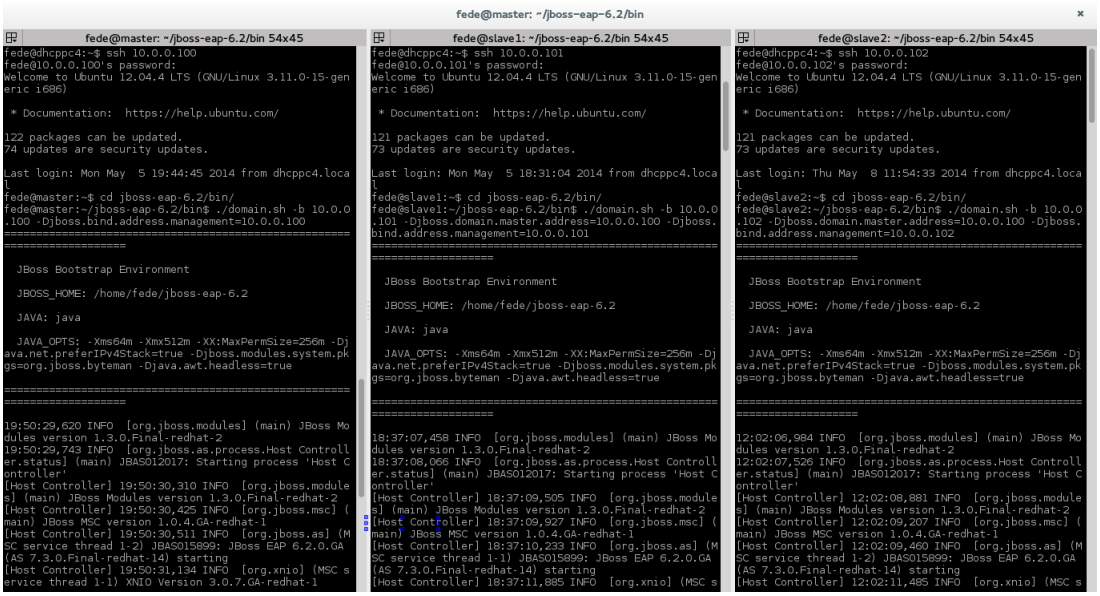

Figura 3: I 3 server JBoss in domain mode sulle 3 macchine virtuali

ate, gli elementi del cluster precedentemente creato e il boot da CD. In una delle macchine virtuali abbiamo installato Ubuntu tramite l'interfaccia VNC.

Per permettere alle macchine virtuali di connettersi a internet è stato necessario fornire le seguenti regole di iptables nella macchina fisica in modo che fornisse l'interfaccia wireless quale input e output per la macchina virtuale:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
# iptables -A FORWARD -i smbr0 -j ACCEPT
```

Nelle macchine virtuali, invece, abbiamo settato la macchina fisica come gateway mediante i seguenti comandi, inseriti in /etc/rc.local.

```
# ip addr add 10.0.0.100/8 dev eth0
# ip link set eth0 up
# route add default gw 10.0.0.1
# hostname master
# echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

Inoltre è necessario cambiare in master il nome dell'host in */etc/hostname* e configurare opportunamente il file /etc/hosts:

```
master 10.0.0.100
```

```
slave1 10.0.0.101
slave2 10.0.0.102
```

A questo punto abbiamo installato alcuni componenti necessari:

```
# apt-get install openssh-server unzip zip openjdk-7-jre openjdk
    -7-jdk nfs-kernel-server portmap nfs-common
```

e scaricato JBoss EAP 6.2 e il load balancer Mod Cluster dai seguenti link (`https://www.jboss.org/products/eap.html` e `http://downloads.jboss.org/mod_cluster//1.2.6.Final/linux-i686/mod_cluster-1.2.6.Final-linux2-x86.tar.gz`) seguendo la guida (`http://docs.jboss.org/mod_cluster/1.2.0/html/Quick_Start_Guide.html`). Ci riferiremo da ora in poi con JHOME a /home/fede/jboss-eap-6.2/.
A questo punto, nell'interfaccia Sunstone abbiamo clonato il disco fisso della macchina virtuale due volte e abbiamo cambiato il nome dell'hostname nei file opportuni in, rispettivamente, slave1 e slave2. Quindi lo scenario presenta in totale tre macchine virtuali: master e slave1, in stato running sul front-end, e slave2 in stato running sul member host.
In ognuna delle tre macchine virtuali abbiamo eseguito lo script JHOME-/bin/./adduser.sh, aggiungendo un utente di JBoss per ognuna delle tre macchine virtuali, rispettivamente chiamati admin, slave1 e slave2. Al termine, viene fornito in output una password in base 64, che viene utilizzata per modificare i files *JHOME/domain/configuration/host.xml* secondo il tutorial alla pagina `https://access.redhat.com/site/solutions/218053`.
Per avviare lo scenario è necessario modificare il file di configurazione di *httpd* nel master con la seguente sezione:

```
Listen 10.0.0.100:80

ServerName 10.0.0.100

<IfModule manager_module>
  Listen 10.0.0.100:6666
  ManagerBalancerName other_server_group
  <VirtualHost 10.0.0.100:6666>
    <Location />
     Order deny,allow
     Deny from all
     Allow from 10.0.0
    </Location>
    KeepAliveTimeout 300
    MaxKeepAliveRequests 0
```

```
    AdvertiseFrequency 5
    EnableMCPMReceive
    <Location /mod_cluster_manager>
        SetHandler mod_cluster-manager
        Order deny,allow
        Deny from all
        Allow from 10.0.0
    </Location>
  </VirtualHost>
</IfModule>
```

In tale modo sarà possibile al master, in ascolto su porta 80, di fungere da gateway di tutte le richieste e distribuirle agli slave effettuando così il load balancing e nascondendo all'utente il numero e gli indirizzi dei server. In seguito sarà necessario avviare il server Apache sul master e i server JBoss su tutte le VM in domain mode coi seguenti comandi:

```
$ /opt/jboss/httpd/sbin/apachectl start          // su master
$ JHOME/bin/./domain.sh -b 10.0.0.100 -Djboss.bind.address.
    management=10.0.0.100          // su master
$ JHOME/bin/domain.sh -b 10.0.0.101 -Djboss.domain.master.
    address=10.0.0.100 -Djboss.bind.address.management=10.0.0.101
     // su slave1
$ JHOME/bin/domain.sh -b 10.0.0.102 -Djboss.domain.master.
    address=10.0.0.100 -Djboss.bind.address.management=10.0.0.102
     // su slave2
```

Ora sarà possibile accedere al manager di JBoss via browser all'indirizzo `http://10.0.0.100:9990` con l'utente admin precedentemente creato ed eseguire le seguenti azioni:

- Assegnare server-one e server-two a other-server-group.
- Assegnare server-one esclusivamente a slave1 e server-two esclusivamente a slave2.
- Nella sezione Manage Deployments caricare il file .war contenente l'applicazione JBoss (spiegata tra poche righe) e asseganrla a other-server-group.
- Accedere all'applicazione tramite browser: `http://10.0.0.100/nomeapplicazione`.

L'applicazione stateful JBoss è un semplice *Helloworld* che legge il numero contenuto di un file condiviso e, tramite la pressione di un bottone, consente

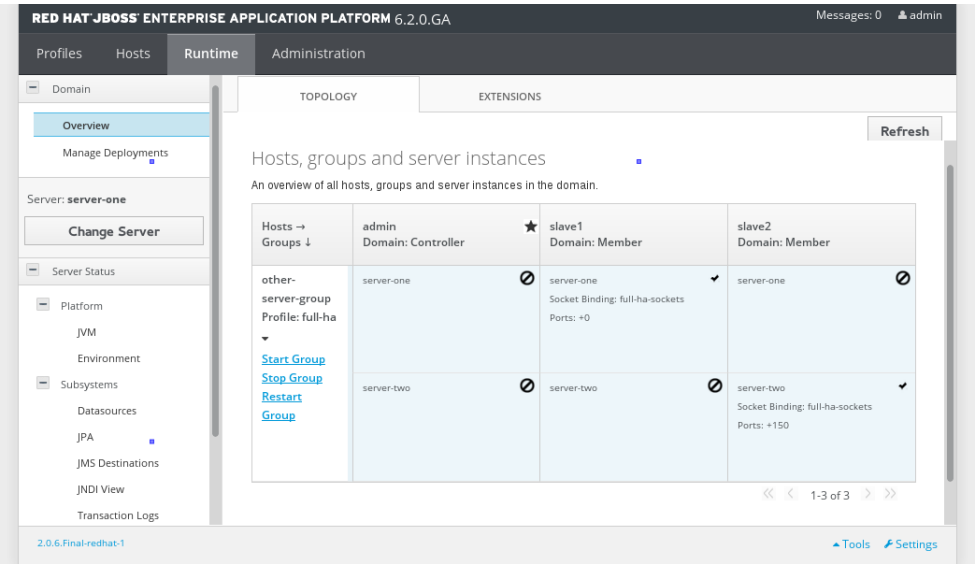

Figura 4: La console di amministrazione JBoss

di sommargli 1. Il **lato client** consiste in una pagina html vuota che richiama la servlet *HelloWorld*. La **servlet** effettua due tipi di richieste:

- *doGet* per la lettura del numero contenuto nel file condiviso *file.txt*
- *doPost*, che si attiva alla pressione del tasto Add 1 per l'incremento numero contenuto nel file condiviso *file.txt*

Anche in questo caso, è stato necessario aggiungere il path del file all'interno di */etc/exports* in tutte le macchine virtuali:

```
JHOME/bin/data/          *(rw,sync,no_subtree_check)
```

È inoltre necessario che gli slave effettuino una mount sulla directory per condividerla via NFS:

```
# mount -t nfs4 10.0.0.100:JHOME/bin/data JHOME/bin/data
```

L'applicazione è stata sviluppata con l'IDE Eclipse v. 3.7.2 ed esportata come *Helloworld.war*.

Figura 5: L'applicazione JBoss realizzata

3 Conclusioni

Opennebula e Sunstone sono tool che semplifica la gestione e il monitoring di macchine virtuali, attraverso un'interfaccia intuitiva e semplice da usare. JBoss e mod_cluster costituiscono un container portabile che permette in maniera relativamente semplice il deployment di applicazioni distribuite, occupandosi di gestire efficientemente load balancing e fault tolerance. Nonostante le potenzialità dei tool utlizzati, abbiamo riscontrato svariate difficoltà durante lo svolgimento del progetto:

- Una delle due macchine utilizzate inizialmente non consentiva l'attivazione delle istruzione per la virtualizzazione durante il boot del sistema. In particolare, non era possibile caricare il modulo *kvm-amd* necessario per l'utilizzo delle macchine virtuali. È stato necessario cambiare macchina fisica (ringraziamo Davide Berardi per averci concesso di utilizzare il suo laptop).

- Inizialmente abbiamo tentato l'installazione di pacchetti Debian dai repository unstable, ma alcune librerie generavano errori di dipendenze che rendevano impossibile completare l'installazione di tutti i pacchetti necessari

- L'interfaccia Sunstone è instabile in quanto non sempre è in grado di garantire il corretto funzionamento o la persistenza dell'attività delle delle macchine virtuali

- Affinchè gli host venissero correttamente monitorati da Opennebula, si è reso indispensabile modificare i permessi di alcune cartelle legate alla gestione delle chiavi SSH:
  - la home directory di oneadmin e la relativa sottocartella *.ssh* devono avere permessi 700 o 755
  - il file *id_rsa* contenente la chiave privata deve avere permessi 600
  - il file *authorized_keys* deve avere permessi 600